Savitribai Phule Pune University

Section-I

T. Y. B. C. A. (Science)

SEMESTER V

BCA - 505

Lab Course – I

Java Programming

Assignments

EDITORS:

Section-I: Java Programming

Mrs. Misbah Shaikh
Ms. Shaikh Fatima Rafique
Dr. Girija Dhumal
Prof. Rupali Nankar
Prof.Kamil Khan
Prof.Reshma Nagwade
Prof . Nadaf Farzana
Prof.Deepak Derle

Table of Contents for Section-II

Assignment Completion Sheet

Lab Course I			
Java Programming Assignments			
Sr. No.	Assignment Name	Marks (out of 5)	Teachers Sign
1	Java Tools and IDE, Simple Java programs		
2	Usage of Class and Object		
3	Inheritance and Interfaces		
4	Collections		
5	Exception Handling and I/O		
6	Swings		
7	Applet Programming		
8	JDBC		
9	Servlet		
10	JSP		
Total (Out of 50)			
Total (Out of 10)			

This is to certify that Mr./Ms.

Has successfully completed the Advanced Web Technology course work for Lab Course II and has scored _____ Marks out of 30.

Instructor H.O.D / Coordinator

Internal Examiner External Examiner

Section-I: Java Programming

Assignment 1: Java Tools and IDE, Simple Java programs

Objective:

- Introduction to the java environment
- Use of java tools like java, javac, jdb and javadoc
- Use of IDE – Eclipse (demo)
- Defining simple classes and creating objects.

Reading:

You should read the following topics before starting this exercise

1. Creating, compiling and running a java program.
2. The java virtual machine.
3. Java tools like javac, java, javadoc, javap and jdb.
4. Java keywords
5. Syntax of class.

Ready Reference :

Java Tools

(1) javac:-javac is the java compiler which compiles .java file into .class file(i.e. bytecode). If the program has syntax errors, javac reports them. If the program is error-free, the output of this command is one or more .class files.

Syntax:

javac fileName.java

(2) java:- This command starts Java runtime environment, loads the specified .class file and executes the main method.

Syntax:

javafileName

(3) javadoc:-javadoc is a utility for generating HTML documentation directly fromcomments written in Java source code.Javadoc comments have a special form but seems like an ordinary multiline comment to the compiler.

Syntax of the comment:

/**

A sample doc comment

*/

Syntax:

javadoc [options] [packagenames] [sourcefiles] [@files]

Where,

packagenames: A series of names of packages, separated by spaces

sourcefiles: A series of source file names, separated by spaces

@files: One or more files that contain packagenames and sourcefiles in any order, one name per line.

Javadoc creates the HTML documentation on the basis of the javadoc tags used in the source code files. These tags are described in the table below:

Tag	Syntax	Description
@see	@see reference	Allows you to refer to the documentation in other classes.
@author	@author authorinformation	Author-information contains author name, and / or authoremail address or any other appropriate information.
@version	@version	VersioninformationSpecifies the version of the program
@since	@since version	This tag allows you to indicate the version of this code that began using a particular feature.
@param	@param name description	This is used for method documentation. Here, name is the identifier in the method parameter list, and description is text that can describes the parameter.
@return	@return description	This describes the return type of a method.
@throws	@throws classname description	This is used when we handle Exceptions. It describes a particular type of exception that can be thrown from the method call.
@deprecated	@deprecated description	The deprecated tag suggests that this feature is no longer supported. A method that is marked @deprecated causes the compiler to issue a warning if it is used.

(4) jdb: -

jdb helps you find and fix bugs in Java language programs. This debugger has limited functionality.

Syntax:

jdb [options] [class] [arguments]

options : Command-line options.

class : Name of the class to begin debugging.

arguments : Arguments passed to the main() method of class.

After starting the debugger, the jdb commands can be executed. The important jdbcommands are:

i. help, or?: The most important jdb command, help displays the list of recognized commands with a brief description.

ii. run: After starting jdb, and setting any necessary breakpoints, you can use this command to start the execution the debugged application.

iii. cont: Continues execution of the debugged application after a breakpoint, exception, orstep.

iv. print: Displays Java objects and primitive values. For variables or fields of primitive types, the actual value is printed. For objects, a short description is printed.

Examples:

printMyClass.myStaticField
printmyObj.myInstanceField
printi + j + k
printmyObj.myMethod()//if myMethod returns non-null

v. dump: For primitive values, this command is identical to print. For objects, it printsthe current value of each field defined in the object. Static and instance fields areincluded.

vi. next: The next command advances execution to the next line in the current stack frame.

vii. step: The step commands advances execution to the next line whether it is in the current stack frame or a called method. Breakpoints can be set in jdb at line numbers, constructors, beginning of a method.

Example:

stop at MyClass:10 //sets breakpoint at instruction at line 10 of the source file containingMyClass
stop in MyClass.display // sets breakpoint at beginning of method display in MyClass
stop in MyClass.<init> //sets breakpoint at default constructor of MyClass
stop in MyClass.<init(int)> //sets breakpoint at parameterized constructor with int asparameter

(4) javap: -

The javap tool allows you to query any class and find out its list of methods and constants.

javap [options] class

Example: javapjava.lang.String

It is a disassembler which allows the bytecodes of a class file to be viewed when used with a classname and the –c option.

javap -c class

Setting CLASSPATH

The classpath is the path that the Java runtime environment searches for classes and other resource files. The class path can be set using either the –classpath option or by setting the CLASSPATH environment variable.

The -classpath option is preferred because you can set it individually for each application without affecting other applications and without other applications modifying its value. The default value of the class path is ".", meaning that only the current directory is searched. Specifying either the CLASSPATH variable or the -cp command line switch overrides this value.

javac –classpath \myProg\myPackage; \myProg\otherclasses

Or

CLASSPATH= classpath1;classpath2...
export CLASSPATH

Example

CLASSPATH=.:/usr/local/classes.jar:/home/user1/myclasses
export CLASSPATH

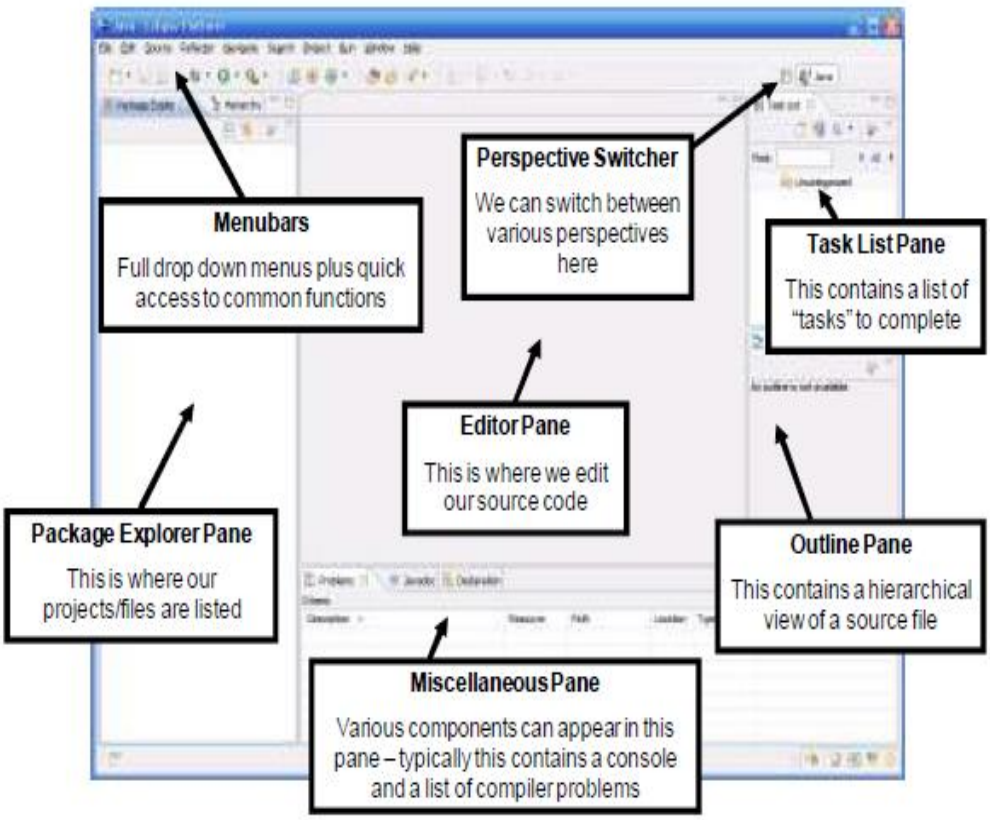
To retain the existing classpath setting, use $CLASSPATH in the new list.

export CLASSPATH=$CLASSPATH:/home/user1/myclasses

About Eclipse

Eclipse is a popular IDE (Integrated Development Environment) for java programming. Itcontains a base workspace and an extensible plug-in system for customizing the environment. Thelatest Eclipse version 4.5 was released in 2015. The Eclipse IDE is also available as an IDE forother languages, ranging from C, C++ to Lua, Python, Perl and PHP. It provides an editor,debugger, source control and other tools.

The GUI looks as shown:



Steps to run a java program using Eclipse:
1. Select workspace
2. Create a new project
3. Project name appears in package explorer, src folder contains source code files
4. Create class (New->Class)
5. Run your program (right click on the class and select Run As -> Java Application)

Self-Activity (using IDE and editor)

1. Sample program

```
/* Program to generate documentation*/
/**
This program demonstrates javadoc
*/
public class MyClass
{
        intnum;
        /**
        Default constructor*/
        publicMyClass()
         {
                num=0;
        }
        /**
                Member function
                @param x Represents the new value of num
```

```
        @return void No return value
    */
    public void assignValue(int x) {
    num = x;

}
```

Type the following command: javadoc MyClass.java. See the HTML documentation file MyClass.html

2. Sample program

```
/* Program to define a class and an object of the class* /
public class MyClass
{
    intnum;
    publicMyClass()
    {
        num=0;
    }
    publicMyClass(intnum)
    {
        this.num = num;
    }
    public static void main(String[] args)
    {
        MyClass m1 = new MyClass();
        if(args.length> 0)
        {
            int n = Integer.parseInt(args[0]);
            MyClass m2 = new MyClass(n);
            System.out.println(m1.num);
            System.out.println(m2.num);
        }
        else
        System.out.println("Insufficient arguments");
    }
}
```

Pass one command line argument to the above program and execute it.

Lab Assignments

SET A

1. Using javap, view the methods of the following classes from the lang package: java.lang.Object , java.lang.String and java.util.Scanner.
2. Compile sample program 2. Type the following command and view the bytecodes.

javap -c MyClass

SET B

1. Write a java program to display the system date and time in various formats shown below:

    Current date is : 31/07/2015
    Current date is : 07-31-2015
    Current date is : Friday July 31 2015
    Current date and time is : Fri July 31 16:25:56 IST 2015
    Current date and time is : 31/07/15 16:25:56 PM +0530
    Current time is : 16:25:56
    Current week of year is : 31
    Current week of month : 5
    Current day of the year is : 212

Note: Use java.util.Date and java.text.SimpleDateFormat class

2. Define a class MyNumber having one private int data member. Write a default constructor to initialize it to 0 and another constructor to initialize it to a value (Use this). Write methods isNegative, isPositive, isZero, isOdd, isEven. Create an object in main. Use command line arguments to pass a value to the object (Hint : convert string argument to integer) and perform the above tests. Provide javadoc comments for all constructors and methods and generate the html help file.

Signature of the instructor:---------- Date:----------

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: well done [ ]

Assignment 2: Usage of Class and Object

Objectives:

1. Defining a class.
2. Defining a Object
3. Creating an array of objects.
4. Writing constructor for class and overload the constructor
5. Using this keyword
6. Using prdefine classes
7. Defining nested classes ,Inner classes,Anonomous class and object.
8. Defining package and use it.
9. Using finalize() method.

Reading:

You should read the following topics before starting this exercise:

1. Structure of a class in java.
2. Declaring class reference.
3. Creating an object using new.
4. Declaring an array of references.
5. Creating an array of objects.
6. Constructor,overloadind of constructor and use of this keyword
7. Predefine classes-String class
   -String Buffer class
   -wrapper class
8. Inner class,nested class.local class,Anonamous class and object.
9. package creation,access and use.
10. Garbage collection finalize( ) method.

Ready Reference:

Functions	Syntax	Example

Class	**class** <class_name>
	{ //access specifiers
	Data member;
	//access specifiers
	Member Functions;
	}
		Class Student
		{
		private:
		int RollNo;
		String Name;
		Public:
		Void Accept();
		Void display();
		}

Object
1.By reference variable
	1.Class_name object_name=new class_name;
	Student s1=**new** Student();
		class Student{
		int id;
		String name;
		}
		class TestStudent2{
		public static void main(String args[])
		{
		Student s1=**new** Student(); //object creation by references variable
		s1.id=101;
		s1.name="Sonoo";
		System.out.println(s1.id+" "+s1.name);//printing members with a white space
		}
		}

		Output: 101 Sonoo

2.Initialization Through method	In this example, we are creating the two objects of Student class and initializing the value to these objects by invoking the insertRecord method. Here, we are displaying the state (data) of the objects by invoking the displayInformation() method.	class Student
		{
		int rollno;
		String name;
		void insertRecord(int r, String n)
		{
		rollno=r;
		name=n;
		}
		void displayInformation(){System.out.println(rollno+" "+name);}
		}
		class TestStudent4{
		public static void main(String args[]){
		Student s1=new Student();
		Student s2=new Student();
		s1.insertRecord(111,"Karan");
		s2.insertRecord(222,"Aryan");
		s1.displayInformation();
		s2.displayInformation();
		}
		}
		Output:
		111 Karan
		222 Aryan

Array of object	class_name obj[ ]
	=new class_name[max_size];
		class Student
		{
		int marks;
		}
		public class ArrayOfObjects
		{
		public static void main(String args[])
		{
		Student std[] = new Student[3];// array of reference variables of Student
		std[0] = new Student(); // convert each reference variable into Student object
		std[1] = new Student();
		std[2] = new Student();
		std[0].marks = 40; // assign marks to each Student element
		std[1].marks = 50;
		std[2].marks = 60;
		System.out.println("\n students average marks:"
		+ (std[0].marks+std[1].marks+std[2].marks)/3);
		}
		}
		Output:
		students average marks: 50

Default constructor: A constructor that has no parameter is known as default constructor.	Class class_name
	{
	//code
	Class_name()//default constructor
	{ //code
		import java.io.*;
		class Student
		{
		int id;

	}
	}
		String name;
		Student() //Default constructor
		{ System.out.println("Default constructor");
		id=0;
		name="";
		}
		void display()
		{
		System.out.println(id+" "+name);
		}
		public static void main(String args[])
		{
		Student s1=new Student();
		s1.display();
		}
		}
		Output:
		Default constructor
		0

Parameterized constructor	Class class_name
	{
	//code
	Class_name(datatype val)//default constructor
	{ //code
	}
	}
		import java.io.*;
		class Student
		{ int id;
		String name;
		Student(int i,String n) //parameterized constructor
		{ System.out.println("Parameterized constructor");
		id = i;
		name = n; }
		void display() { System.out.println(id+" "+name); }
		public static void main(String args[])
		{ Student s1 = new Student(111,"Karan");
		s1.display(); }
		}
		Output:
		Parameterized constructor
		111 karan

Copy constructor	There is no copy constructor in java. But, we can copy the values of one object to another like copy constructor in C++.	import java.io.*;
		class Student6{
		int id;
		String name;
		Student6(int i,String n){
		id = i;
		name = n;
		}
		Student6(Student6 s){
		id = s.id;
		name =s.name; }
		void display(){System.out.println(id+" "+name);}
		public static void main(String args[]){
		Student6 s1 = new Student6(111,"Karan");
		Student6 s2 = new Student6(s1); //copy the constructor
		s1.display();
		s2.display();

		} } Output: 111 Karan 111 Karan

Overloading of constructor	Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter lists. The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.	mport java.io.*;
		lass Student5{
		int id;
		String name;
		int age;
		Student5(int i,String n){
		id = i;
		name = n;
		}
		Student5(int i,String n,int a){
		id = i;
		name = n;
		age=a;
		}
		void display(){System.out.println(id+" "+name+" "+age);}
		public static void main(String args[]){
		Student5 s1 = new Student5(111,"Karan");
		Student5 s2 = new Student5(222,"Aryan",25);
		s1.display();
		s2.display();
		}
		
		)utput:
		11 Karan 0
		22 Aryan 25

Use of "this " keyword	This keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.	import java.io.*;
		class Student{
		int rollno;
		String name;
		float fee;
		Student(int rollno,String name,float fee){
		this.rollno=rollno;
		this.name=name;
		this.fee=fee;
		}
		void display(){System.out.println(rollno+" "+name+" "+fee);}
		}
		class TestThis2{
		public static void main(String args[]){
		Student s1=new Student(111,"ankit",5000f);
		s1.display();
		}}
		Output:
		111 ankit 5000.0

Predefine classes **String class:**	**Method**	**Description**

The java.lang.String class provides a lot of methods to work on string. By the help of these methods, we can perform operations on string such as trimming, concatenating, converting, comparing, replacing strings etc.

String toLowerCase()	The java string toLowerCase() method returns the string in lowercase letter. In other words, it converts all characters of the string into lower case letter.
String toUpperCase()	The java string toUpperCase() method returns the string in uppercase letter. In other words, it converts all characters of the string into upper case letter.
String trim()	The string trim() method eliminates white spaces before and after string.
String startsWith() and endsWith() method	The java string startsWith() method checks if this string starts with given prefix. It returns true if this string starts with given prefix else returns false. The java string endsWith() method checks if this string ends with given suffix. It returns true if this string ends with given suffix else returns false.
string charAt()	The string charAt() method returns a character at specified index.
string length()	The string length() method returns length of the string.
string valueOf()	The string valueOf() method coverts given type such as int, long, float, double, boolean, char and char array into string.
string replace()	The string replace() method replaces all occurrence of first sequence of character with second sequence of character.
string compareTo()	The java string compareTo() method compares the given string with current string lexicographically. It returns positive number, negative number or 0.It compares strings on the basis of Unicode value of each character in the strings.If first string is lexicographically greater than second string, it returns positive number (difference of character value). If first string is less than second string lexicographically, it returns negative number and if first string is lexicographically equal to second string, it returns 0.
String concat(String str)	The String concat() method concatenates the specified string to the end of current string.
String substring (int beginIndex) String substring(int beginIndex, int endIndex)	The **java string substring()** method returns a part of the string.We pass begin index and end index number position in the java substring method where start index is inclusive and end index is exclusive. In other words, start index starts from 0 whereas end index starts from 1.
java string split()	The java string split() method splits this string against given regular expression and returns a char array.

Sample Program:

```
import java.io.*;
class StringExample   {
public static void main(String args[])
   {
String s1="hello STRING";
String s1upper=s1.toUpperCase();
System.out.println(s1upper);
String s1lower=s1.toLowerCase();
System.out.println(s1lower);
char ch=s1.charAt(4);//returns the char value at the 4th index
System.out.println(s1);
String s2="  hello string   ";
System.out.println(s2+"javatpoint");//without trim()
System.out.println(s2.trim()+"javatpoint");//with trim()
String s3="java string split method";
System.out.println(s3.startsWith("ja"));  // it will print true
String s4="java endsWith";
System.out.println(s4.endsWith("t"));// it will print false
System.out.println("string length is: "+s1.length());//print the no. of char  in s1
int value=30;
String s5=String.valueOf(value);
System.out.println(s5);//it will print 30
tring s6="java program";
tring replaceString=s6.replace('a','e');
replaces all occurrences of 'a' to 'e'
ystem.out.println(replaceString);
tring s7="hello";
tring s8="hellow";
ystem.out.println(s7.compareTo(s8));
tring s9="java string";
9.concat("is immutable");
ystem.out.println(s9);
ystem.out.println(s1.substring(2,4));
ystem.out.println(s1.substring(2));
tring s10="java string split method ";
tring[] words=s10.split("\\s");
splits the string based on whitespace
using java foreach loop to print elements of string array
or(String w:words){
ystem.out.println(w);
```

```
Output:-
HELLO STRING
hello string
hello STRING
  hello string   javatpoint
hello stringjavatpoint
true
false
string length is: 12
30
jeve progrem
-1
java string
ll
llo STRING
java
string
split
method
```

String buffered class: Java StringBuffer class is used to create mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

Important Constructors of StringBuffer class:

Constructor	**Description**
StringBuffer()	creates an empty string buffer with the initial capacity of 16.
StringBuffer(String str)	creates a string buffer with the specified string.
StringBuffer(int capacity)	creates an empty string buffer with the specified capacity as length.

Method	**Description**
append(String s)	is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc.
insert(int offset, String s)	is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc.
replace(int startIndex, int endIndex, String str)	is used to replace the string from specified startIndex and endIndex.
delete(int startIndex, int endIndex)	is used to delete the string from specified startIndex and endIndex.
reverse()	is used to reverse the string.
capacity()	is used to return the current capacity.
ensureCapacity(int minimumCapacity)	is used to ensure the capacity at least equal to the given minimum.
charAt(int index)	is used to return the character at the specified position.
length()	is used to return the length of the string i.e. total number of characters.
substring(int beginIndex)	is used to return the substring from the specified beginIndex.
substring(int beginIndex, int endIndex)	is used to return the substring from the specified beginIndex and endIndex.

Sample program:

```
import java.io.*;
class StringBufferExample{
public static void main(String args[]){
StringBuffer sb=new StringBuffer("Hello ");
sb.append("Java");//now original string is changed
System.out.println(sb);//prints Hello Java
 sb.insert(1,"Java");//now original string is changed
System.out.println(sb);//prints HJavaello
sb.replace(1,3,"Java");
System.out.println(sb);//prints HJavalo
sb.delete(1,3);
System.out.println(sb);//prints Hlo
sb.reverse();
System.out.println(sb);//prints olleH
StringBuffer sb1=new StringBuffer();
System.out.println(sb1.capacity());//default 16
sb1.append("Hello");
System.out.println(sb1.capacity());//now 16
sb1.append("java is my favourite language");
System.out.println(sb1.capacity());//now (16*2)+2=34 i.e (oldcapacity*2)+2
sb1.ensureCapacity(10);//now no change
System.out.println(sb1.capacity());//now 34
sb1.ensureCapacity(50);//now (34*2)+2
System.out.println(sb1.capacity());//now 70
}
}
```

```
Output:
Hello Java
HJavaello Java
HJavavaello Java
Hvavaello Java
avaJ olleavavH
16
16
34
34
```

Wrapped class

Wrapper class in java provides the mechanism *to convert primitive into object and object into primitive*.Since J2SE 5.0, **autoboxing** and **unboxing** feature converts primitive into object and object into primitive automatically. The automatic conversion of primitive into object is known as autoboxing and vice-versa unboxing.

The eight classes of *java.lang* package are known as wrapper classes in java. The list of eight wrapper classes are given below:

Primitive Type	Wrapper class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

To convert the argument from String to any type, use Wrapper classes.

Method	Purpose
Byte.parseByte	Returns byte equivalent of a String
Short.parseShort	Returns the short equivalent of a String
Integer.parseInt	Returns the int equivalent of a String
Long.parseLong	Returns the long equivalent of a String
Float.parseFloat	Returns the float equivalent of a String
Double.parseDouble	Returns the double equivalent of a String

```
//Java program to demonstrate Wrapping
public class WrapperExample{
public static void main(String args[]){
//Converting int into Integer
int a=20;
Integer i=Integer.valueOf(a);//converting int into Integer
Integer j=a;//autoboxing, now compiler will write Integer.valueOf(a) internally
System.out.println(a+" "+i+" "+j);
}}
```

Output:

```
20 20 20
```

```
//program for unwrapping
public class WrapperExample2{
public static void main(String args[]){
//Converting Integer to int
Integer a=new Integer(3);
int i=a.intValue();//converting Integer to int
```

<table>
<tr><td></td><td colspan="2">int j=a;//unboxing, now compiler will write a.intValue() internally
System.out.println(a+" "+i+" "+j);
}}
Output:
3 3 3</td></tr>
<tr><td>Inner class</td><td>We use inner classes to logically group classes and interfaces in one place so that it can be more readable and maintainable. Additionally, it can access all the members of outer class including private data members and methods.
Syntax of Inner class
class Java_Outer_class{
//code
class Java_Inner_class{
//code
} }</td><td>class TestMemberOuter1{
private int data=30;
class Inner{
void msg(){System.out.println("data is "+data);}
}
public static void main(String args[]){
TestMemberOuter1 obj=new TestMemberOuter1();
TestMemberOuter1.Inner in=obj.new Inner();
in.msg();
}
}
Output:
data is 30</td></tr>
<tr><td>Nested class</td><td>Inner class is a part of nested class. Non-static nested classes are known as inner classes.</td><td>There are two types of nested classes non-static and static nested classes.The non-static nested classes are also known as inner classes.
○ Non-static nested class (inner class)
1. Member inner class
2. Anonymous inner class
3. Local inner class
○ Static nested class</td></tr>
<tr><td>Local class</td><td>A class i.e. created inside a method is called local inner class in java. If you want to invoke the methods of local inner class, you must instantiate this class inside the method.</td><td>public class localInner1{
private int data=30;//instance variable
void display(){
class Local{
void msg(){System.out.println(data);}
}
Local l=new Local();
l.msg();
}
public static void main(String args[]){
localInner1 obj=new localInner1();
obj.display();
}
}
Output: 30</td></tr>
<tr><td>Anonymous Inner Class: An inner class declared without a class name is known as an anonymous inner class. In case of anonymous inner classes, we declare and instantiate them at the same time. Generally, they are used whenever you need to override the method of a class or an interface. The syntax of an anonymous inner class is as follows Syntax
AnonymousInner an_inner = new AnonymousInner() {
public void my_method() {
……..
……..
} };</td><td colspan="2">abstract class AnonymousInner {
public abstract void mymethod();
}
public class Outer_class {
public static void main(String args[]) {
AnonymousInner inner = new AnonymousInner() {
public void mymethod() {
System.out.println("This is an example of anonymous inner class");
}
};
inner.mymethod();
}
}
Output:</td></tr>
</table>

This is an example of anonymous inner class

Anonymous Object

Anonymous simply means nameless. An object which has no reference is known as anonymous object. It can be used at the time of object creation only.

```
New Calculation();//anonymous object
```

Calling method through anonymous object

```
new Calculation().fact(5);
```

```
import java.io.*;
class Calculation{
 void fact(int  n){
  int fact=1;
  for(int i=1;i<=n;i++){
   fact=fact*I;
  }
 System.out.println("factorial is "+fact);
}
public static void main(String args[]){
 new Calculation().fact(5);//calling method with anonymous object
}
}
```

Output:

Factorial is 120

Package creation,access,use

Creating a package

To create a user defined package, the package statement should be written in the source code file. This statement should be written as the first line of the program. Save the program in a directory of the same name as the package.

```
package packageName;
```

Accessing a package

To access classes from a package, use the import statement.

```
import packageName.*; //imports all classes
import packageName.className; //imports specified class
```

Note that the package can have a hierarchy of subpackages. In that case, the package name should be qualified using its parent packages. Example: project.sourcecode.java

Here, the package named project contains one subpackage named sourcecode which contains a subpackage named java.

Access Rules

The access rules for members of a class are given in the table below.

Accessible to	public	protected	none	private
Same class	Yes	Yes	Yes	Yes
Class in same package	Yes	Yes	Yes	No
Subclass (in other package)	Yes	Yes	No	No
Non subclass in Other package	Yes	No	No	No

Sample program for package:

```
//save by A.java
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
//save by B.java
package mypack;
import pack.*;

class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

Compile the package by javac -d . A.java

Output: Hello

Garbage collection finalized method

In java, garbage means unreferenced objects.Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.

finalize() method:The finalize() method is invoked each time before the object is garbage collected. This method can be used to perform cleanup processing. This method is defined in Object class as:

```
protected void finalize(){}
import java.util.*;
```

```
public class ObjectDemo extends GregorianCalendar {

  public static void main(String[] args) {
    try {
      // create a new ObjectDemo object
      ObjectDemo cal = new ObjectDemo();

      // print current time
      System.out.println("" + cal.getTime());

      // finalize cal
      System.out.println("Finalizing...");
      cal.finalize();
      System.out.println("Finalized.");

    } catch (Throwable ex) {
      ex.printStackTrace();
    }
  }
}
```

Output:

```
Sat dec 22 00:27:21 EEST 2017
Finalizing...
Finalized.
```

Lab Assignments

SET A

Run all the sample program.

1. Write a java program which accepts 3int values and prints the maximum and minimum.
2. Write a java program which accepts an integer arrary and print the data and sort the data in descending order.
3. Write a java program which define class Employee with data member as name and salary.program store the information of 5 Employees.display the name who earn maximum salary.(Use arrary of object)
4. Define a Student class (roll number, name, percentage). Define a default and parameterized constructor. Keep a count of objects created. Create objects using parameterized constructor and display the object count after each object is created. (Use static member and method). Also display the contents of each object.
5. Write a java program which accepts a string and a characters to be search from the user the program should display the totalno of character in string.
6. Define a class person(pid,pname,age,gender). Define the default and parametrized constructor.Overlaod the constructor.Accept the 5 person details and display it.(use this keyword.)

SET B

1. Create a package named Series having three different classes to print series:

    a. Prime numbers b. Fibonacci series c. Squares of numbers Write a program to generate ‘n’ terms of the above series.

2. Write a Java program to create a Package “SY” which has a class SYMarks (members – ComputerTotal, MathsTotal, and ElectronicsTotal). Create another package TY which has a class TYMarks (members – Theory, Practicals). Create n objects of Student class (having rollNumber, name, SYMarks and TYMarks). Add the marks of SY and TY computer subjects and calculate the Grade (‘A’ for >= 70, ‘B’ for >= 60 ‘C’ for >= 50 , Pass Class for > =40 else ‘FAIL’) and display the result of the student in proper format.
3. Writa a java program that take input as aperson name in the format of first middle last name and then print it in the form last first and middle,where in the middle name first character is capital letter.
4. Write a package game which will have 2 classes Indoor and Outdoor.Use a function display to generate the list of player for the specific game.use default and parametrized constructor.
5. Define class student(rno,name,marks1,marks2) .Define result class(total,percentage) inside the student class.accept the student details and display the Marksheet with Rno,name,marks1,Marks2,toatl,percentage.(use inner class concept.)

SET C:

1. Define a class Employee(id,name,department,salary).define default and parametrized constuctor also overlaod the constructor.Create the inner class manager(bonus).define the accept and display in both the class.create the n object of manager and display the details of manager having maximum salary.(sal=sal+bouns)

Signature of the instructor:---------- Date:----------

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: well done [ ]

Assignment 3 – Inheritance and Interfaces

Objectives:

- To implement inheritance in java.
- To define abstract classes.
- To define and use interfaces.
- Use predefined interfaces like Cloneable

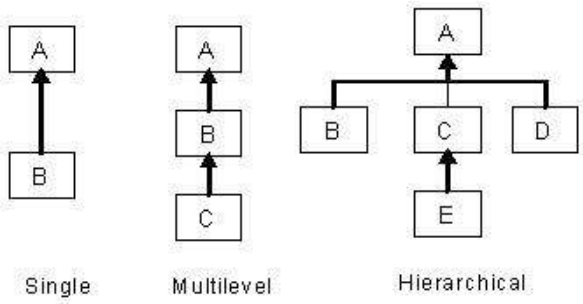
Reading :

You should read the following topics before starting this exercise:
1. Concept of inheritance.
2. Use of extends keyword.
3. Concept of abstract class.
4. Defining an interface.
5. Use of implements keyword.

Ready Reference :

Inheriting a class : The syntax to create a subclass is :
classSubClassName extends SuperClassName
{
//class body
}
Example:
class Manager extends Employee
{ //code }

Types of Inheritance



Access in subclass
The following members can be accessed in a subclass:
i) public or protected superclass members.
ii) Members with no specifier if subclass is in same package.

The "super" keyword
It is used for three purposes:

i) Invoking superclass constructor - super(arguments)
ii) Accessing superclass members – super.member
iii) Invoking superclass methods – super.method(arguments)

Example:

```
class A
{
        protectedintnum;
        A(intnum) { this.num = num; }
}
class B extends A
{
        intnum;
        B(int a, int b)
        {
                super(a); //should be the first line in the subclass constructor
                this.num = b;
        }
        void display()
        {
                System.out.println("In A, num = " + super.num);
                System.out.println("In B, num = " + num);
        }
}
```

Overriding methods

Redefining superclass methods in a subclass is called overriding. The signature of the subclass method should be the same as the superclass method.

```
class A
{
        void method1(intnum) {//code}
}
class B extends A
{
        void method1(int x) { //code }
}
```

Dynamic binding

When over-riding is used, the method call is resolved during run-time i.e. depending on the object type, the corresponding method will be invoked.

Example:

```
A ref;
ref = new A();
ref.method1(10); //calls method of class A
ref = new B();
ref.method1(20); //calls method of class B
```

Abstract class

An abstract class is a class which cannot be instantiated. It is only used to create subclasses. A class which has abstract methods must be declared abstract. An abstract class can have data members, constructors, method definitions and method declarations.

```
abstract class ClassName
{
...
}
```

Abstract method

An abstract method is a method which has no definition. The definition is provided by the subclass.

```
abstractreturnType method(arguments);
```

Interface

An interface is a pure abstract class i.e. it has only abstract methods and final variables. An interface can be implemented by multiple classes.

```
interfaceInterfaceName
{
        //abstract methods
        //final variables
}
```

Example:

```
interfaceMyInterface
{
        void method1();
        void method2();
        int size= 10; //final and static
}
classMyClass implements MyInterface
{
        //define method1 and method2
}
```

Self Activity

1. Sample program to demonstrate inheritance and interfaces

```
interface Shape
{
        double area();
}
class Circle implements Shape
{
        double radius;
        Circle(double radius)
        {
                this.radius=radius;
        }
        public double area()
        {
                returnjava.util.Math.PI * radius* radius;
        }
}
class Cylinder extends Circle
{
        double height;
        Cylinder(double radius, double height)
        {
                super(radius);
                this.height=height;
        }
        public double area() //overriding
        {
                returnjava.util.Math.PI * radius* radius *height;
        }     }

public class Test
{
        public static void main(String[] args)
```

```
{
        Shape s;
        s = new Circle(5.2);
        System.out.println("Area of circle = " + s.area());
        s = new Cylinder(5, 2.5);
        System.out.println("Area of cylinder = " + s.area());
}       }
```

Lab Assignments

SET A

1. Define a class Employee having private members – id, name, department, salary. Define default and parameterized constructors. Create a subclass called "Manager" with private member bonus. Define methods accept and display in both the classes. Create n objects of the Manager class and display the details of the manager having the maximum total salary (salary+bonus)

2. Create an abstract class Shape with methods calc_area and calc_volume. Derive three classesSphere(radius) , Cone(radius, height) and Cylinder(radius, height), Box(length, breadth, height)from it. Calculate area and volume of all. (Use Method overriding).

SET B

1. Define an abstract class "Staff" with members name and address. Define two sub-classes of this class – "FullTimeStaff" (department, salary) and "PartTimeStaff" (number-of-hours, rate-perhour). Define appropriate constructors. Create n objects which could be of either FullTimeStaffor PartTimeStaff class by asking the user's choice. Display details of all "FullTimeStaff" objects and all "PartTimeStaff" objects.

2. Write a Java program to create a super class Vehicle having members Company and price.Derive 2 different classes LightMotorVehicle (members – mileage) and HeavyMotorVehicle(members – capacity-in-tons). Accept the information for n vehicles and display the informationin appropriate form. While taking data, ask the user about the type of vehicle first.

SET C

1. Create an interface "CreditCardInterface" with methods :viewCreditAmount(), useCard(),payCredit() and increaseLimit(). Create a class SilverCardCustomer (name, cardnumber (16digits), creditAmount – initialized to 0, creditLimit - set to 50,000) which implements the aboveinterface. Inherit class GoldCardCustomer from SilverCardCustomer having the same methodsbut creditLimit of 1,00,000. Create an object of each class and perform operations. Displayappropriate messages for success or failure of transactions. (Use method overriding)
    i. useCard() method increases the creditAmount by a specific amount uptocreditLimit
    ii. payCredit() reduces the creditAmount by a specific amount.
    iii. increaseLimit() increases the creditLimit for GoldCardCustomers (only 3 times, not more than 5000Rs. each time)

Signature of the instructor:---------- Date:----------

Assignment Evaluation

0: Not Done [ ] 1: Incomplete [ ] 2: Late Complete [ ]

3: Needs Improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Assignment 4 – Collections

Objectives

- Study the Collections framework in java
- Use various collections

Collection:

A collection is an object that represents a group of objects. A collection — sometimes called a container — is simply an object that groups multiple elements into a single unit. Collections are used to store, retrieve, manipulate, and communicate aggregate data.

Interface Name	Description
Collection	The most general collection interface type. A collection represents a group of objects known as its elements. The Java platform doesn't provide any direct implementations of this interface.
List	Represents an ordered collection. Lists can contain duplicate elements.
Set	Represents an unordered collection that does not permit duplicate elements.
SortedSet	Represents a set whose elements are maintained in a sorted order.
Queue	A collection used to hold multiple elements prior to processing. A Queue provides additional insertion, extraction and inspection operations.
Map	Represents key-value pairs. A Map cannot contain duplicate keys; each key can map to at most one value. Does not extend collection.
SortedMap	Represents a Map that maintains its mappings in ascending key order.

The Collection interface:

Method	Explanation
boolean add(Object element)	Method adds objects in the collection.
boolean remove(Object element)	Method removes objects in the collection.
The Collection interface also supports query operations:	
int size()	Returns the size of the collection.
boolean isEmpty()	Returns true if the collection is empty or false.
Boolean contains(Object element)	Returns true if the collection contains the element passed in argument.
Other operations are tasks done on groups of elements or the entire collection at once:	
boolean containsAll(Collection collection)	The containsAll() method allows you to discover if the current collection contains all the elements of another collection, a subset.
boolean	addAll(Collection collection)
void	removeAll(Collection collection)

List interface:

Method	Description
void add(int index, Object element)	Inserts the specified element at the specified position in this list (optional operation).
Boolean addAll(int index, Collection c)	Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
Object get(int index)	Returns the element at the specified position in this list.
int indexOf(Object o)	Returns the index in this list of the first occurrence of the specified element, or -1 if this list does not contain this element.
int lastIndexOf(Object o)	Returns the index in this list of the last occurrence of the specified element, or -1 if this list does not contain this element.
ListIterator listIterator()	Returns a list iterator of the elements in this list (in proper sequence). ListIterator
listIterator(int index)	Returns a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list.
Object remove(int index)	Removes the element at the specified position in this list (optional operation).
Object set(int index, Object element)	Replaces the element at the specified position in this list with the specified element (optional operation).
int size()	Returns the number of elements in this list.
List subList(int fromIndex, int toIndex)	Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive.

Set Interface and SortedSet Interface:

Method	Description
`Comparator comparator()`	Returns the comparator associated with this sorted set, or null if it uses its elements' natural ordering.
`Object first()`	Returns the first (lowest) element currently in this sorted set.
`SortedSet headset(Object toElement)`	Returns a view of the portion of this sorted set whose elements are strictly less than toElement.
`Object last()`	Returns the last (highest) element currently in this sorted set.
`SortedSet subset(Object fromElement, Object toElement)`	Returns a view of the portion of this sorted set whose elements range from fromElement, inclusive, to toElement, exclusive.
`SortedSet tailSet(Object fromElement`	Returns a view of the portion of this sorted set whose elements are greater than or equal to fromElement.

Map Interface:

Method	Description
`void clear()`	Removes all mappings from this map
`boolean containsKey(Object key)`	Returns true if this map contains a mapping for the specified key.
`boolean containsValue(Object value)`	Returns true if this map maps one or more keys to the specified value.
`Set entrySet()`	Returns a set view of the mappings contained in this map.
`boolean equals(Object o)`	Compares the specified object with this map for equality.
`Object get(Object key)`	Returns the value to which this map maps the specified key.
`int hashCode()`	Returns the hash code value for this map.
`boolean isEmpty()`	Returns true if this map contains no key-value mappings
`Set keySet()`	Returns a set view of the keys contained in this map.
`Object put(Object key, Object value)`	Associates the specified value with the specified key in this map
`void putAll(Map t)`	Copies all of the mappings from the specified map to this map
`Object remove(Object key)`	Removes the mapping for this key from this map if it is present (optional operation).
`int size()`	Returns the number of key-value mappings in this map.
`Collection values()`	Returns a collection view of the values contained in this map

List Implementations:

There are two general-purpose List implementations — ArrayList and LinkedList.

1. ArrayList: Resizable-array implementation of the List interface. Implements all optional list operations, and permits all elements, including null. Each ArrayList instance has a capacity. The capacity is the size of the array used to store the elements in the list. It is always at least as large as the list size.
2. LinkedList: The LinkedList class implements the List interface. All of the operations perform as could be expected for a doubly-linked list. Operations that index into the list will traverse the list from the beginning or the end, whichever is closer to the specified index.

ArrayList Method name	Description
addAll(int index, Collection c)	Inserts all of the elements in the specified Collection into this list, starting at the specified position.
ensureCapacity(int minCapacity)	Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.
removeRange(int fromIndex, int toIndex)	Removes from this List all of the elements whose index is between fromIndex, inclusive and toIndex, exclusive.
trimToSize()	Trims the capacity of this ArrayList instance to be the list's current size.
LinkedList Method name	**Description**
`addFirst(Object o)`	Inserts the given element at the beginning of this list.
`addLast(Object o)`	Appends the given element to the end of this list.
`Object getFirst()`	Returns the first element in the list
`Object getLast()`	Returns the last element in the list
`Object removeFirst()`	Removes and returns the first element from this list.
`Object removeLast()`	Removes and returns the last element from this list.

`ListIterator listIterator(int index)`	Returns a list-iterator of the elements in this list (in proper sequence), starting at the specified position in the list.

Set Implementations:

There are three general-purpose Set implementations — HashSet, TreeSet, and LinkedHashSet.

1. TreeSet: The elements are internally stored in a search tree. It is useful when you need to extract elements from a collection in a sorted manner.

2. HashSet: It creates a collection the uses a hash table for storage. The advantage of HashSet is that it performs basic operations (add, remove, contains and size) in constant time and is faster than TreeSet

3. LinkedHashSet: The only difference is that the LinkedHashSet maintains the order of the items added to the Set, The elements are stored in a doubly linked list.

Iterator:

The Iterator interface provides methods using which we can traverse any collection. This interface is implemented by all collection classes.

`hasNext``()`	`true` if there is a next element in the collection.
`next``()`	Returns the next object.
`remove()`	Removes the most recent element that was returned by next()

ListIterator

Forward iteration	
`hasNext``()`	`true` if there is a next element in the collection.
`next``()`	Returns the next object.
Backward iteration	
`hasPrevious``()`	`true` if there is a previous element.
`previous``()`	Returns the previous element.
Getting the index of an element	
`nextIndex``()`	Returns index of element that would be returned by subsequent call to next().
`previousIndex``()`	Returns index of element that would be returned by subsequent call to previous().
Optional modification methods.	
`add(`*`obj`*`)`	Inserts obj in collection before the next element to be returned by `next()` and after an element that would be returned by `previous()`.
`set()`	Replaces the most recent element that was returned by `next` or `previous()`.
`remove()`	Removes the most recent element that was returned by `next()` or `previous()`.

HashTable

The important methods of the HashTable class are

Method name	Description
`void clear()`	Clears this hashtable so that it contains no keys.
`boolean contains(Object value)`	Tests if some key maps into the specified value in this hashtable.
`boolean containsKey(Object key)`	Tests if the specified object is a key in this hashtable.
`containsValue(Object value)`	Returns true if this Hashtable maps one or more keys to this value.

`Enumeration elements()`	Returns an enumeration of the values in this hashtable.
`Set entrySet()`	Returns a Set view of the entries contained in this Hashtable.
`Object get(Object key)`	Returns the value to which the specified key is mapped in this hashtable.
`int hashCode()`	Returns the hash code value for this Map as per the definition in the Map interface.
`Boolean isEmpty()`	Tests if this hashtable maps no keys to values.
`Enumeration keys()`	Returns an enumeration of the keys in this hashtable.
`Set keySet()`	Returns a Set view of the keys contained in this Hashtable.
`Object put(Object key, Object value)`	Maps the specified key to the specified value in this hashtable.
`void putAll(Map m)`	Copies all of the mappings from the specified Map to this Hashtable These mappings will replace any mappings that this Hashtable had for any of the keys currently in the specified Map.
`void rehash()`	Increases the capacity of and internally reorganizes this hashtable, in order to accommodate and access its entries more efficiently.
`Object remove(Object key)`	Removes the key (and its corresponding value) from this hashtable.
`int size()`	Returns the number of keys in this hashtable.
`Collection values()`	Returns a Collection view of the values contained in this Hashtable.

Sample Program

```
/* Program to demonstrate ArrayList and LinkedList */
import java.util.*;
class ArrayLinkedListDemo {
public static void main(String args[])
{
ArrayList al = new ArrayList(); LinkedList l1 = new LinkedList();
System.out.println("Initial size of al: " + al.size());
// add elements to the array list al.add("A");
al.add("B");
al.add("C");
al.add(2, "AA"); System.out.println("Contents of al: " + al);

al.remove("B"); al.remove(1);
System.out.println("Contents of al: " + al); l1.add("A");
l1.add("B");
l1.add(new Integer(10));
System.out.println("The contents of list is " + l1); l1.addFirst(“AA”);
l1.addLast("c");
l1.add(2,"D");
l1.add(1,"E");
l1.remove(3);
System.out.println("The contents of list is " + l1);
}
}
```

2.Sample program

```
/* Program to demonstrate iterator */
import java.util.*; public class IteratorDemo
{
public static void main(String[] args)
{
ArrayList a1 = new ArrayList();
al.add("C"); al.add("A"); al.add("E"); al.add("B"); al.add("D"); al.add("F");
Iterator itr = a1.iterator();       //obtain iterator while(itr.hasNext())
```

```
{
String elt = (String)itr.next(); System.out.println("Element = " + elt);
}
LinkedList l = new LinkedList();
l.add("A");  l.add("B");  l.add("C");  l.add("D"); ListIterator litr = l.listIterator();
while(litr.hasNext())
{
String elt = (String)litr.next(); System.out.println(elt);
}
System.out.println("Traversing Backwards : "); while(litr.hasPrevious())
System.out.println(litr.previous());
}
}
```

3.Sample program

```
/* Program to demonstrate HashTable*/
import  java.util.*; public class HashtableDemo
{
public static void main(String[] args)
{
Hashtable hashtable = new Hashtable(); String str, name = null;
hashtable.put( "A", 75.2 ); // adding value into hashtable hashtable.put( "B", 65.9 );
hashtable.put( "C", 95.1 );
hashtable.put( "D", 85.7 );

System.out.println("Retriving all keys from the Hashtable"); Enumeration keys = hashtable.keys();
while( keys. hasMoreElements() )
System.out.println( keys.nextElement() );

System.out.println("Retriving all values from the table"); Enumeration values = hashtable.elements();
while( values. hasMoreElements() )
System.out.println( values.nextElement() );
}
}
```

Lab Assignments

SET A

1. Accept ‘n’ integers from the user and store them in a collection. Display them in the sorted order. The collection should not accept duplicate elements. (Use a suitable collection). Search for an particular element using predefined search method in the Collection framework.

2. Create a Hash table containing Employee name and Salary. Display the details of the hash table. Also search for a specific Employee and display Salary of that Employee.

SET B

1. Construct a linked List containing names of colors: red, blue, yellow and orange. Then extend your program to do the following:
   i. Display the contents of the List using an Iterator;
   ii. Display the contents of the List in reverse order using a ListIterator;

iii. Create another list containing pink and green. Insert the elements of this list between blue and yellow.

2 Create a java application to store city names and their STD codes using an appropriate collection. The GUI ahould allow the following operations:

i. Add a new city and its code (No duplicates)
ii. Remove a city from the collection
iii. Search for a cityname and display the code

SET C

1. Read a text file, specified by the first command line argument, into a list. The program should then display a menu which performs the following operations on the list:

1. Insert line 2. Delete line 3. Append line 4. Modify line 5. Exit

When the user selects Exit, save the contents of the list to the file and end the program.

Signature of the instructor:---------- Date:----------

Assignment Evaluation

0: Not Done [ ] 1: Incomplete [ ] 2: Late Complete [ ]

3: Needs Improvement [ ] 4: Complete [ ] 5: Well Done [ ]

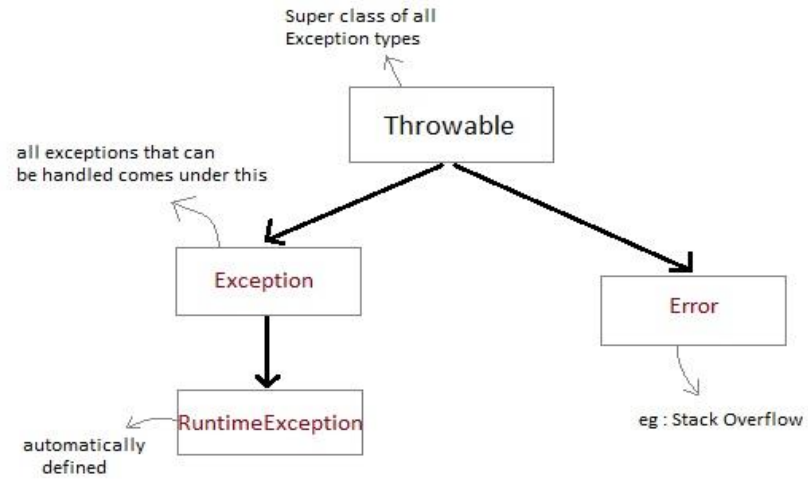
Assignment 5 – Exception Handling and I/O

Exception Handling:

Exception Handling is the mechanism to handle runtime malfunctions. We need to handle such exceptions to prevent abrupt termination of program. The term exception means exceptional condition, it is a problem that may arise during the execution of program. A bunch of things can lead to exceptions, including programmer error, hardware failures, files that need to be opened cannot be found, resource exhaustion etc.

Exception:
A Java Exception is an object that describes the exception that occurs in a program. When an exceptional events occurs in java, an exception is said to be thrown. The code that's responsible for doing something about the exception is called an **exception handler**.

Exception class Hierarchy:
All exception types are subclasses of class **Throwable**, which is at the top of exception class hierarchy.



- **Exception** class is for exceptional conditions that program should catch. This class is extended to create user specific exception classes.
- **RuntimeException** is a subclass of Exception. Exceptions under this class are automatically defined for programs.
- **Exceptions** of type **Error** are used by the Java run-time system to indicate errors having to do with the run-time environment, itself. Stack overflow is an example of such an error.

Exception are categorized into 3 category.

- **Checked Exception**

  The exception that can be predicted by the programmer at the compile time.*Example :* File that need to be opened is not found. These type of exceptions must be checked at compile time.

Checked Exceptions:

1.Exception
2.IOException
3.FileNotFoundException
4.ParseException
5.ClassNotFoundException
6.CloneNotSupportedException
7.InstantiationException
8.InterruptedException
9.NoSuchMethodException
10.NoSuchFieldException

- **Unchecked Exception**

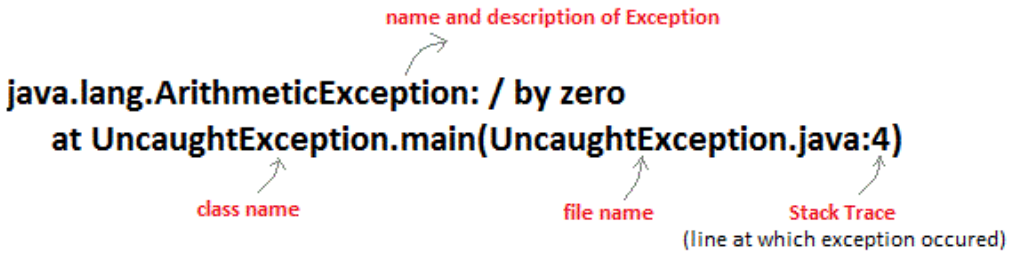
Unchecked exceptions are the class that extends Runtime Exception. Unchecked exception are ignored at compile time. Example : Arithmetic Exception, Null Pointer Exception, Array Index out of Bound exception. Unchecked exceptions are checked at runtime.

Unchecked Exceptions:

1.ArrayIndexOutOfBoundsException
2.ClassCastException
3.IllegalArgumentException
4.IllegalStateException
5.NullPointerException
6.NumberFormatException
7.AssertionError
8.ExceptionInInitializerError
9.StackOverflowError
10.NoClassDefFoundError

Uncaught Exceptions:
When we don't handle the exceptions, they lead to unexpected program termination. Lets take an example for better understanding.

```
classUncaughtException
{
publicstaticvoidmain(String args[])
{
int a =0;
int b =7/a;// Divide by zero, will lead to exception
}
}
```



Exception Handling Mechanism

In java, exception handling is done using five keywords,

1. **try**
2. **catch**
3. **throw**
4. **throws**
5. **finally**

Exception handling is done by transferring the execution of a program to an appropriate exception handler when exception occurs.

Using try and catch:
Try is used to guard a block of code in which exception may occur.
Example using Try and catch

```
classExcp
{
public static void main(String args[])
 {
inta,b,c;
try
  {
   a=0;
   b=10;
   c=b/a;
System.out.println("This line will not be executed");
  }
catch(ArithmeticException e)
  {
System.out.println("Divided by zero");
  }
System.out.println("After exception is handled");
 }
}
```

Output:
Divided by zero
After exception is handled

An exception will thrown by this program as we are trying to divide a number by zero inside **try** block. The program control is transferred outside **try** block. Thus the line "*This line will not be executed*" is never parsed by the compiler. The exception thrown is handled in **catch** block.

Once the exception is handled, the program control is continue with the next line in the program i.e after catch block. Thus the line "*After exception is handled*" is printed.

Multiple catch blocks:
A try block can be followed by multiple catch blocks..
Example for Multiple Catch blocks:

```
classExcep
{
public static void main(String[] args)
 {
try
  {
intarr[]={1,2};
arr[2]=3/0;
  }
catch(ArithmeticExceptionae)
  {
System.out.println("divide by zero");
  }
catch(ArrayIndexOutOfBoundsException e)
  {
System.out.println("array index out of bound exception");
  }
 }
}
```

Output:
divide by zero

Note: Although both ArrayIndexOutOfBoundsException and ArithmeticExceptionoccured, but since first catch is of Arithmetic Exception, It will be caught there and program control will be continued after the catch block.

Note: At a time, only one exception is processed and only one respective catch block is executed.

Example for Unreachable Catch block:

```
classExcep
{
public static void main(String[] args)
 {
try
  {
intarr[]={1,2};
arr[2]=3/0;
  }
catch(Exception e)    //This block handles all Exception
  {
System.out.println("Generic exception");
  }
catch(ArrayIndexOutOfBoundsException e)    //This block is unreachable
```

```
  {
System.out.println("array index out of bound exception");
  }
 }
}
```
Output:
Generic exception

Nested try statement:
try statement can be **nested** inside another block of **try**.

```
classExcep
{
public static void main(String[] args)
 {
try
  {
intarr[]={5,0,1,2};
try
   {
int x=arr[3]/arr[1];
   }
catch(ArithmeticExceptionae)
   {
System.out.println("divide by zero");
   }
arr[4]=3;
  }
catch(ArrayIndexOutOfBoundsException e)
  {
System.out.println("array index out of bound exception");
  }
 }
}
```
Output:
divide by zero
array index out of bound exception
throw Keyword:

throw keyword is used to throw an exception explicitly. Only object of Throwable class or its sub classes can be thrown. Program execution stops on encountering throw statement, and the closest catch statement is checked for matching type of exception.

Syntax :
throwThrowableInstance
Creating Instance of Throwable class:

There are two possible ways to create an instance of class Throwable,

1. Using a parameter in catch block.
2. Creating instance with **new** operator.

newNullPointerException("test"); //This constructs an instance of NullPointerException with name test.
Example demonstrating throw Keyword:
```
class Test
{
static void avg()
```

```
 {
try
  {
throw new ArithmeticException("demo");
  }

catch(ArithmeticException e)
  {
System.out.println("Exception caught");
  }
 }

public static void main(String args[])
 {
avg();
 }
}
```

In the above example the avg() method throw an instance of ArithmeticException, which is successfully handled using the catch statement and thus, the program outputs "Exception caught".

throws Keyword:

Any method that is capable of causing exceptions must list all the exceptions possible during its execution, so that anyone calling that method gets a prior knowledge about which exceptions are to be handled. A method can do so by using the throws keyword.

Syntax :

```
Typemethod_name(parameter_list)throwsexception_list

{

 //definition of method

}
```

Example demonstrating throws Keyword:

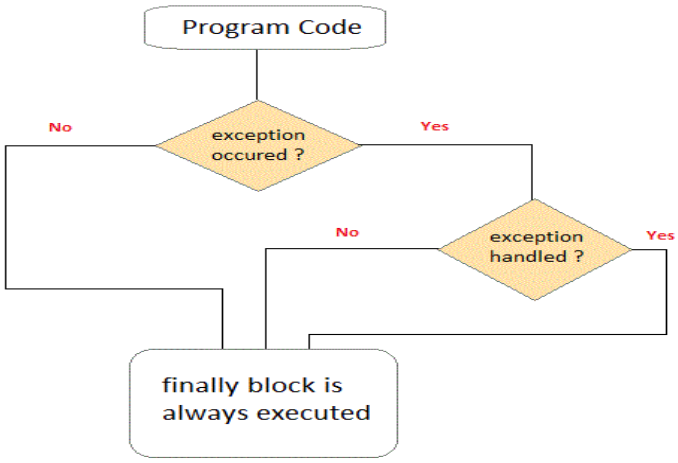
```
class Test
{
static void check() throws ArithmeticException
 {
System.out.println("Inside check function");
throw new ArithmeticException("demo");
 }

public static void main(String args[])
 {
try
  {
check();
  }
catch(ArithmeticException e)
  {
System.out.println("caught" + e);
  }
```

```
 }
}
```

Output:Inside check function
caughtjava.lang.ArithmeticException: demo

finally clause

A finally keyword is used to create a block of code that follows a try block. A finally block of code is always executed whether an exception has occurred or not.



Example demonstrating finally Clause:

```
Class ExceptionTest
{
public static void main(String[] args)
 {
int a[]= new int[2];
System.out.println("out of try");
try
  {
System.out.println("Access invalid element"+ a[3]);
/* the above statement will throw ArrayIndexOutOfBoundException */
  }
finally
  {
System.out.println("finally is always executed.");
  }
 }
}
```

Output:
Out of try
finally is always executed.
Exception in thread main java. Lang. exception array Index out of bound exception.

You can see in above example even if exception is thrown by the program, which is not handled by catch block, still finally block will get executed.

User defined Exception subclass:

You can also create your own exception sub class simply by extending java **Exception** class. You can define a constructor for your Exception sub class (not compulsory) and you can override the **toString()**function to display your customized message on catch.

```
classMyException extends Exception
  {
privateint ex;
MyException(int a)
   {
ex=a;
   }
public String toString()
   {
return "MyException[" + ex +"] is less than zero";
   }
  }

class Test
  {
static void sum(inta,int b) throws MyException
   {
if(a<0)
    {
throw new MyException(a); //calling constructor of user-defined exception class
```

```
    }
else
    {
System.out.println(a+b);
    }
   }
public static void main(String[] args)
   {
try
    {
sum(-10, 10);
    }
catch(MyException me)
    {
System.out.println(me); //it calls the toString() method of user-defined Exception
    }
   }
 }
```

Output:MyException[-10] is less than zero

Assertion:
Assertion is a statement in java. It can be used to test your assumptions about the program. While executing assertion, it is believed to be true. If it fails, JVM will throw an error named AssertionError. It is mainly used for testing purpose.

Advantage of Assertion:
It provides an effective way to detect and correct programming errors.

Syntax of using Assertion:
There are two ways to use assertion. First way is:
assert expression;

and second way is:

assert expression1 : expression2;

Simple Example of Assertion in java:

```
importjava.util.Scanner;

classAssertionExample{
public static void main( String args[] ){



Scanner scanner = new Scanner( System.in );
System.out.print("Enter ur age ");

int value = scanner.nextInt();
  assert value>=18:" Not valid";

System.out.println("value is "+value);
 }
}
```

Output: Enter ur age 11

Exception in thread "main" java.lang.AssertionError: Not valid

Lab Assignments:

Set A:

1.Write a java program to accept a number from the user, if number is zero then throw user defined exception “Number is 0” otherwise check whether no is prime or not (Use static keyword).

2. Define a class MyDate(Day,Month,year) with method to accept and display a MyDate object. Accept date as dd, mm, yyyy. Throw user defined exception “InvalidDateExeption” if the date is invalid.

3.Write a Java program to accept user name and password, If the user name and password are not same,then generate user –defined Exception “Incorrect PasswordException”.

4.Write a java program to accept Employee name from the user and check whether it is validor not. If it is not valid then throw user defined Exception “Name is Invalid” otherwise display it.

5.Define Exception VowelException,BlankException,ExitException.Write another class Test which reads a character from command line. If it is a vowel, throw VowelException,if it is blank throw BlankException and for a character ‘X’ throw an ExitException and terminate program.For any other character , display “Valid Character”.

Set B:

1.Write a class Student with attributes roll no, name, age and course. Initialize values through parameterized constructor. If age of student is not in between 15 and 21 then generate user-defined exception “Age Not Within The Range”.If name contains numbers or special symbols raise exception “Name not valid”.

2.Define a class saving account (acno, name, balance) .define appropriate and operation withdraw(), deposit(), and viewbalance(). The minimum balance must be 500. Create an object and perform operation. Raise user defined “InsufficientFundException” when balance is not sufficient for withdraw operation.

3.Define a class cricket player(name, no_ofinings, no_of_times_notout, total_runs, bat_avg).create an array of ‘n’ player objects.Calculate the batting average for each player using a static method avg(). Handle appropriate exception while calculating average. Define static method ‘sortPlayer’ which sorts the array on the basis of average. Display the player details in sorted order**.**

4.Write a program which accept two integers and an arithmetic operator from the command line and performs the operation.Fire the following user defined exceptions:

i.If the no of arguments are less than 3 then,fire “IllegalNumberOfArgument”.
ii.If the operator is nor an Arithmetic operator,throw “InvalidOperatorExceptio”.
iii.If result is –Ve ,then throw “NegativeResultException”

Assignment 5.1: I/O and File Handling

Introduction to java.io package:

Java I/O (Input and Output) is used *to process the input* and *produce the output*.Java uses the concept of stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.We can perform **file handling in java** by Java I/O API.

Stream

A stream is a sequence of data.In Java a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.In java, 3 streams are created for us automatically. All these streams are attached with console.

1) System.out: standard output stream

2) System.in: standard input stream

3) System.err: standard error stream

Let's see the code to print **output and error** message to the console.

```
1.  System.out.println("simple message");
2.  System.err.println("error message");
```

Let's see the code to get **input** from console.

```
1.  int i=System.in.read();//returns ASCII code of 1st character
2.  System.out.println((char)i);//will print the character
```

IO Stream

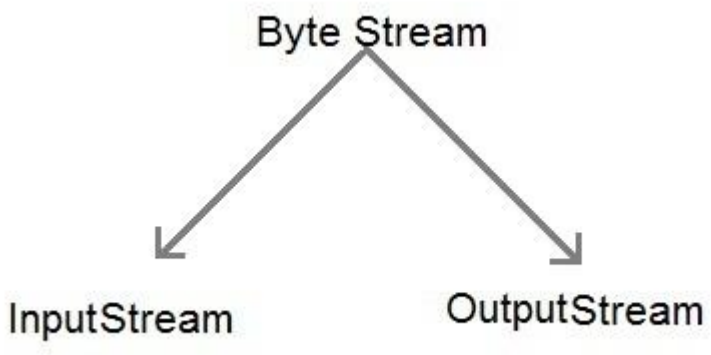
Java performs I/O through **Streams**. A Stream is linked to a physical layer by java I/O system to make input and output operation in java. In general, a stream means continuous flow of data. Streams are clean way to deal with input/output without having every part of your code understand the physical.

Java encapsulates Stream under **java.io** package. Java defines two types of streams. They are,

1. **Byte Stream :** It provides a convenient means for handling input and output of byte.
2. **Character Stream :** It provides a convenient means for handling input and output of characters. Character stream uses Unicode and therefore can be internationalized.

Byte Stream Classes

Byte stream is defined by using two abstract class at the top of hierarchy, they are InputStream and OutputStream.



These two abstract classes have several concrete classes that handle various devices such as disk files, network connection etc.

Some important Byte stream classes.

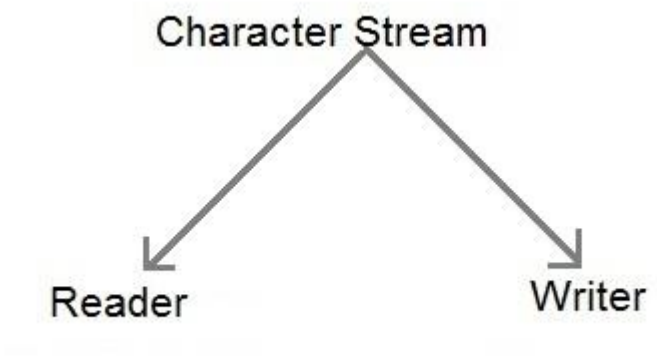
Stream class	Description
BufferedInputStream	Used for Buffered Input Stream.
BufferedOutputStream	Used for Buffered Output Stream.
DataInputStream	Contains method for reading java standard datatype
DataOutputStream	An output stream that contain method for writing java standard data type
FileInputStream	Input stream that reads from a file
FileOutputStream	Output stream that write to a file.
InputStream	Abstract class that describe stream input.
OutputStream	Abstract class that describe stream output.
PrintStream	Output Stream that contain print() and println() method

These classes define several key methods. Two most important are

1. read() : reads byte of data.
2. write() : Writes byte of data.

Character Stream Classes

Character stream is also defined by using two abstract class at the top of hierarchy, they are Reader and Writer.



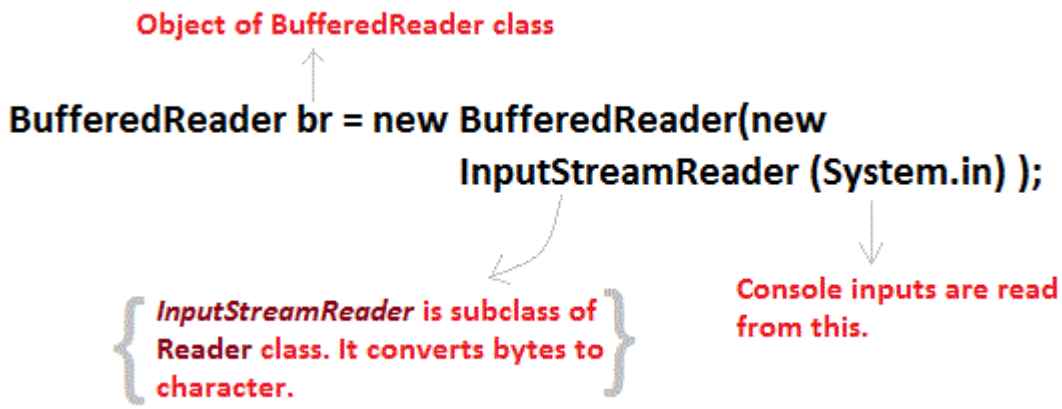
These two abstract classes have several concrete classes that handle unicode character.

Some important Charcter stream classes.

Stream class	Description
BufferedReader	Handles buffered input stream.
BufferedWriter	Handles buffered output stream.
FileReader	Input stream that reads from file.
FileWriter	Output stream that writes to file.
InputStreamReader	Input stream that translate byte to character
OutputStreamReader	Output stream that translate character to byte.
PrintWriter	Output Stream that contain print() and println() method.
Reader	Abstract class that define character stream input
Writer	Abstract class that define character stream output

Reading Console Input

We use the object of BufferedReader class to take inputs from the keyboard.



Reading Characters

`read()` method is used with BufferedReader object to read characters. As this function returns integer type value has we need to use typecasting to convert it into **char** type.

intread()throwsIOException

Below is a simple example explaining character input.

classCharRead

{

publicstaticvoidmain(String args[])

{

BufferedReaderbr=newBufferedreader(newInputstreamReader(System.in));

char c =(char)br.read();//Reading character

}

}

Reading Strings

To read string we have to use `readLine()` function with BufferedReader class's object.

String readLine()throwsIOException

Program to take String input from Keyboard in Java

```
import java.io.*;
classMyInput
{
publicstaticvoidmain(String[]args)
{
  String text;
InputStreamReaderisr=newInputStreamReader(System.in);
BufferedReaderbr=newBufferedReader(isr);
text=br.readLine();//Reading String
System.out.println(text);
}
}
```

Program to read from a file using BufferedReader class

```
import java. Io *;
classReadTest
{
publicstaticvoidmain(String[]args)
{
try
{
   File fl=newFile("myfile.txt");
BufferedReaderbr=newBufferedReader(newFileReader(fl));
   String str;
while((str=br.readLine())!=null)
{
System.out.println(str);
}
br.close();
fl.close();
}
catch(IOException  e)
{e.printStackTrace();}
}
}
```

Program to write to a File using FileWriter class

```
import java. Io *;
classWriteTest
{
publicstaticvoidmain(String[]args)
{
try
{
   File fl=newFile("myfile.txt");
   String str="Write this string to my file";
FileWriterfw=newFileWriter(fl);
fw.write(str);
```

```
fw.close();
fl.close();
}
catch(IOException  e)
{e.printStackTrace();}
}
}
```

Object Serialization Reader and Writer:

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called **deserialization**.

A class must implement **Serializable** interface present in `java.io` package in order to serialize its object successfully. **Serializable** is a **marker interface** that adds serializablebehaviour to the class implementing it.

Java provides **Serializable** API encapsulated under `java.io` package for serializing and deserializing objects which include,

- java.io.serializable
- java.io.Externalizable
- ObjectInputStream
- andObjectOutputStream etc.

Marker interface

Marker Interface is a special interface in Java without any field and method. Marker interface is used to inform compiler that the class implementing it has some special behaviour or meaning. Some example of Marker interface are,

- java.io.serializable
- java.lang.Cloneable
- java.rmi.Remote
- java.util.RandomAccess

All these interfaces does not have any method and field. They only add special behavior to the classes implementing them. However marker interfaces have been deprecated since Java 5, they were replaced by **Annotations**. Annotations are used in place of Marker Interface that play the exact same role as marker interfaces did before.

Signature of writeObject() and readObject()

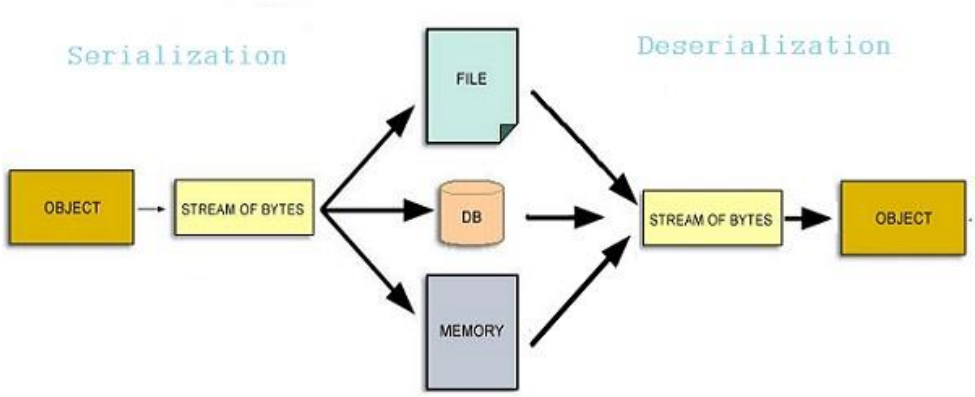
`writeObject()` method of `ObjectOutputStream` class serializes an object and send it to the output stream.

publicfinalvoidwriteObject(object x)throwsIOException

`readObject()` method of `ObjectInputStream` class references object out of stream and deserialize it.

publicfinal Object readObject()throwsIOException,ClassNotFoundException

while serializing if you do not want any field to be part of object state then declare it either static or transient based on your need and it will not be included during java serialization process.



Serializing an Object

```
import java.io.*;
classstudentinfoimplementsSerializable
{
   String name;
int rid;
static String contact;
studentinfo(string n,int r, string c)
{
this.name = n;
this.rid= r;
this.contact= c;
}
}

classTest
{
publicstaticvoidmain(String[]args)
{
try
{
Studentinfosi=newstudentinfo("Abhi",104,"110044");
FileOutputStreamfos=newFileOutputStream("student.ser");
Objectoutputstreamoos=newObjectOutputStream(fos);
oos.writeObject(si);
oos.close();
fos.close();
}
catch(Exception e)
{
e.printStackTrace();
```

```
}
}
}
```
Object of Studentinfo class is serialized using writeObject() method and written to student.serfile.
Deserialization of Object

```
import java.io *;
classDeserializationTest
{
publicstaticvoidmain(String[]args)
{
studentinfosi=null ;
try
{
FileInputStreamfis=newFileInputStream("student.ser");
ObjectOutputStreamois=newObjectOutputStream(fis);
si=(studentinfo)ois.readObject();
}
catch(Exception e)
{
e.printStackTrace();}
System.out.println(si.name);
System.out.println(si.rid);
System.out.println(si.contact);
}
}
```
Output:
Abhi
104
null
Contact field is null because,it was marked as static and as we have discussed earlier static fields does not get serialized.
NOTE: Static members are never serialized because they are connected to class not object of class.

Lab Assignments:
Set A:

1.Write a java program that displays the number of characters, lines & words from a file.

2.Write a java program to read the characters from a file, if a character is alphabet then reverse its case, if not then display its category on the Screen. (whether it is Digit or Space)

3.Write a java program to display the contents of a file in reverse order

Set B:

1.Write a Java program to create 2 files F1 and F2.Copy the contents of file F1 by changing the case into file F2.
F2 = F1
Also copy the contents of F1 and F2 in F3.
F3 = F1+F2
Display the contents of F3.

Use Command Line Arguments.

2. Write a Java program to accept list of files as command line arguments. Display the name and size of all the files. Delete all files with extension as '.html' from the current directory. Appropriate error messages should be printed.(Use vector Array)

3.Write a java program to copy the contents of one file into the another file, while copying change the case of alphabets and replace all the digits by '*' in target file.

Signature of the instructor:---------- Date:----------

Assignment Evaluation

0: Not Done [ ] 1: Incomplete [ ] 2: Late Complete [ ]

3: Needs Improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Assignment 6 - Swings

MVC(Model View Controller) Architecture :-

Swing API architecture follows loosely based MVC architecture in the following manner.

- Model - Model represents component's data.
- View represents visual representation of the component's data.
- Controller takes the input from the user on the view and reflects the changes in Component's data.
- Swing component has Model as a seperate element, while the View and Controller part are clubbed in the User Interface elements. Because of which, Swing has a pluggable look-and-feel architecture.

Swing Classes:

The following table lists some important Swing classes and their description.

Class		Description
1.	Box	Container that uses a BoxLayout
2.	JApplet	Base class for Swing applets
3.	JButton	Selectable component that supports text/image display
4.	JCheckBox	Selectable component that displays state to user
5.	JCheckBoxMenuItem	Selectable component for a menu; displays state to user
6.	JColorChooser	For selecting colors
7.	JComboBox	For selecting from a drop-down list of choices
8.	JComponent	Base class for Swing components
9.	JDialog	Base class for pop-up subwindows
10.	JEditorPane	For editing and display of formatted content
11.	JFileChooser	For selecting files and directories
12.	JFormattedTextField	For editing and display of a single line of formatted text
13.	JFrame	Base class for top-level windows
14.	JInternalFrame	Base class for top-level internal windows
15.	JLabel	For displaying text/images
16.	JList	For selecting from a scrollable list of choices
17.	JMenu	Selectable component for holding menu items; supports text/image display.
18.	JMenuBar	For holding menus.
19.	JMenuItem	Selectable component that supports text/image is Display.
20.	JOptionPane	For creating pop-up messages
21.	JPanel	Basic component container
22.	JPasswordField	For editing and display of a password
23.	JPopupMenu	For holding menu items and popping up over Components.
24.	JProgressBar	For showing the progress of an operation to the user
25.	JRadioButton	Selectable component that displays state to user; included in ButtonGroup to ensure that only one button is selected.
26.	JRadioButtonMenuItem	Selectable component for menus; displays state to user; a. inlcuded in ButtonGroup to ensure that only one button is b. selected
27.		JTableFor display of tabular data
28.	JTextArea	For editing and display of single-attributed textual content
29.	JTextField	For editing and display of single-attributed textual content on a single line
30.	JTextPane	For editing and display of multi-attributed textual content
31.	JToggleButton	Selectable component that supports text/image display; selection triggers component to stay “in” JToolBar Draggable container

Important Containers:
1. JFrame – This is a top-level container which can hold components and containers like panels.

Constructors-
JFrame()
JFrame(String title)

Methods-

setSize(int width, int height) -Specifies size of the frame in pixels
setLocation(int x, int y) -Specifies upper left corner
setVisible(boolean visible) -Set true to display the frame
setTitle(String title) -Sets the frame title
setDefaultCloseOperation(int mode) -Specifies the operation when frame is closed.

The modes are:
JFrame.EXIT_ON_CLOSE JFrame.DO_NOTHING_ON_CLOSE
JFrame.HIDE_ON_CLOSE JFrame.DISPOSE_ON_CLOSE

pack() -Sets frame size to minimum size required to hold components

2. JPanel –
This is a middle-level container which can hold components and can be added to other containers like frame and panels.
Constructors-
public javax.swing.JPanel(java.awt.LayoutManager, boolean);
public javax.swing.JPanel(java.awt.LayoutManager);
public javax.swing.JPanel(boolean);
public javax.swing.JPanel();

Some Important Components :

1.JLabel
With the JLabel class, you can display unselectable text and images.

Constructors-
JLabel(Icon i) JLabel(Icon I , int n)
JLabel(String s) JLabel(String s, Icon i, int n)
JLabel(String s, int n) JLabel()
The int argument specifies the horizontal alignment of the label's contents within its drawing area; defined in the SwingConstants interface (which JLabel implements): LEFT (default), CENTER, RIGHT, LEADING, or TRAILING.

Methods-
1. Set or get the text displayed by the label.
void setText(String) String getText()
2. Set or get the image displayed by the label.
void setIcon (Icon) Icon getIcon()
3. Set or get the image displayed by the label when it's disabled. If you don't specify a disabled image, then the look-and-feel creates one by manipulating the default image.
void setDisabledIcon(Icon) Icon getDisabledIcon()
4. Set or get where in the label its contents should be placed. For vertical alignment: TOP, CENTER (the default), and BOTTOM.

void setHorizontalAlignment(int) void setVerticalAlignment(int)
int getHorizontalAlignment() int getVerticalAlignment()

2. JButton
A Swing button can display both text and an image. The underlined letter in each button's text shows the mnemonic which is the keyboard alternative.
Constructors-

JButton(Icon I)
JButton(String s)
JButton(String s, Icon I)

Methods
void setDisabledIcon(Icon) void setPressedIcon(Icon)
void setSelectedIcon(Icon) void setRolloverIcon(Icon)
String getText() void setText(String)

3.JCheckBox
Class- JCheckBox

Constructors-
JCheckBox(Icon i) JCheckBox(Icon i,booean state)
JCheckBox(String s) JCheckBox(String s, boolean state)
JCheckBox(String s, Icon i) JCheckBox(String s, Icon I, boolean state)

Methods-
void setSelected(boolean state) String getText()
void setText(String s)

4. JRadioButton
Class- JRadioButton
Constructors-
JRadioButton (String s) JRadioButton(String s, boolean state)
JRadioButton(Icon i) JRadioButton(Icon i, boolean state)
JRadioButton(String s, Icon i)
JRadioButton(String s, Icon i, Boolean state)
JRadioButton()
To create a button group- ButtonGroup()

Adds a button to the group, or removes a button from the group-
void add(AbstractButton)
void remove(AbstractButton)

5. JComboBox
Class- JComboBox
Constructors- JComboBox()
Methods -
Void addItem(Object) Object getItemAt(int)
Object getSelectedItem() int getItemCount()

6. JList
Constructor- JList(ListModel)

Methods
boolean isSelectedIndex(int) void setSelectedIndex(int)

void setSelectedIndices(int[]) void setSelectedValue(Object, boolean)
void setSelectedInterval(int, int) int getSelectedIndex()
int getMinSelectionIndex() int getMaxSelectionIndex()
int[] getSelectedIndices() Object getSelectedValue()
Object[] getSelectedValues()

7.JFileChooser-
represents a dialog window from which the user can select file.

Constructor-
JFileChooser() - Constructs a JFileChooser pointing to the user's default directory.
JFileChooser(File currentDirectory) - Constructs a JFileChooser using the given File as thepath.
JFileChooser(String currentDirectoryPath) - Constructs a JFileChooser using the given path.

Methods –
boolean accept (File f) – Returns true if file should be displayed.
String getDialogTitle() – Gets the string that goes in the JFileChooser titlebar.
SetMultiSelectionEnables(true) – allow multiple selection.

8. JColorChooser –
create a color chooser dialog box so that user can select any color

Constructor-
JColorChooser() - create a color chooser panel with white color initially.
JColorChooser(color initialcolor) – create a color chooser panel with the specified color initially.

Methods-
void addChooserPanel(AbstractColorChooserPanel panel) - It is used to add a color chooser panel to the color chooser.

Static Color ShowDialog(Component c, String title,Color initialColor) – show the color chooser dialog box.

9. JMenu –

Constructors –
JMenu() - Creates a menu
JMenu(String) - Creates a menu. The string specifies the text to display for the menu.

Methods –
Inserts a menu item or separator into the menu at the specified position. –
JMenuItem insert(JMenuItem, int)
JMenuItem insert(Action, int)
void insert(String, int)
void insertSeparator(int)

Removes the specified item(s) from the menu.
void remove(JMenuItem)

void remove(int)
void removeAll()

10. Text classes
All text related classes are inherited from JTextComponent class
a. JTextField
Creates a text field. The int argument specifies the desired width in columns. The String argument contains the field's initial text. The Document argument provides a custom document for the field.

Constructors
JTextField() JTextField(String)
JTextField(String, int) JTextField(int)
JTextField(Document, String, int)

b. JPasswordField

Constructors-
JPasswordField() JPasswordField(String)
JPasswordField(String, int) JPasswordField(int)
JPasswordField(Document, String, int)

Methods-
1. Set or get the text displayed by the text field.
void setText(String) String getText()
2. Set or get the text displayed by the text field.
char[] getPassword()
3. Set or get whether the user can edit the text in the text field.
void setEditable(boolean) boolean isEditable()
4. Set or get the number of columns displayed by the text field. This is really just a hint for computing the field's preferred width.
void setColumns(int);
int getColumns()
5. Get the width of the text field's columns. This value is established implicitly by the font. int getColumnWidth()
6. Set or get the echo character i.e. the character displayed instead of the actual characters typed by the user.
void setEchoChar(char) char getEchoChar()

c. JTextArea
Represents a text area which can hold multiple lines of text
Constructors-
JTextArea (int row, int cols)
JTextArea (String s, int row, int cols)
Methods -
void setColumns (int cols) void setRows (int rows)
void append(String s) void setLineWrap (boolean)

8. Dialog Boxes
Swing has a JOptionPane class, that lets you put a simple dialog box.
Methods in JOption Class
1. static void showMessageDialog()- Shows a message with ok button.
2. static int showConfirmDialog()- shows a message & gets users options from set of options.
3. static int showOptionDialog- shows a message & get users options from set of options.

4. String showInputDialog()- shows a message with one line of user input.

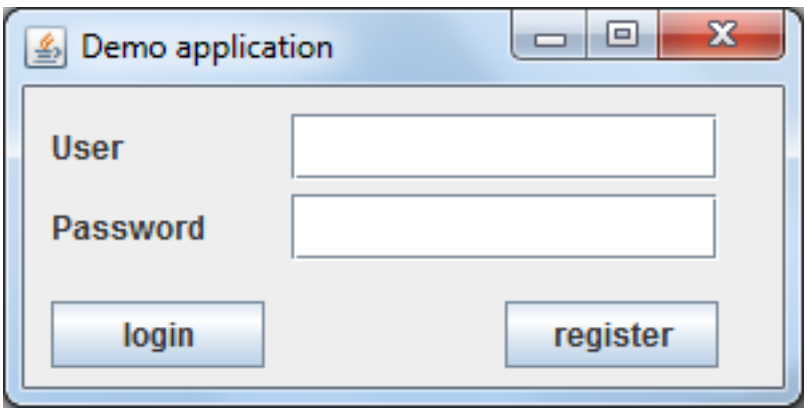
Sample Example of swing using JButton

```
/*1.Java JButton Example */

import javax.swing.*;

public class ButtonExample {
public static void main(String[] args) {
    JFrame f=new JFrame("Button Example");
    JButton b=new JButton("Click Here");
    b.setBounds(50,100,95,30);
    f.add(b);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

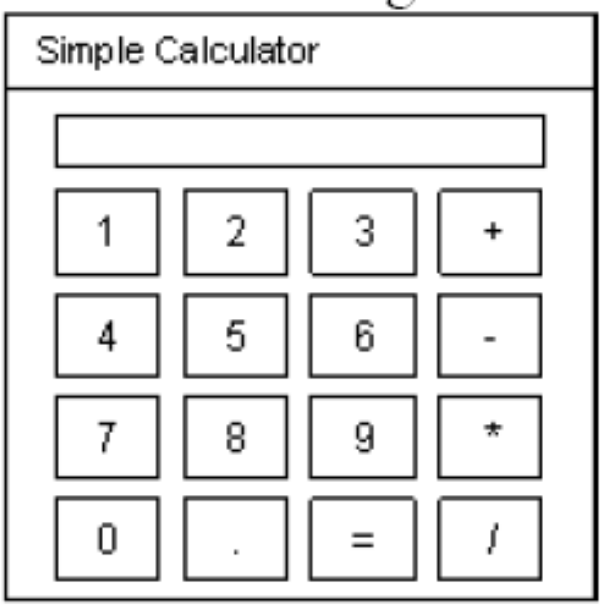
```
/*2.  java JPasswordField Example*/

import javax.swing.*;
public class PasswordFieldExample {
    public static void main(String[] args) {
    JFrame f=new JFrame("Password Field Example");
     JPasswordField value = new JPasswordField();
     JLabel l1=new JLabel("Password:");
       l1.setBounds(20,100, 80,30);
        value.setBounds(100,100,100,30);
          f.add(value);  f.add(l1);
          f.setSize(300,300);
          f.setLayout(null);
          f.setVisible(true);
1.        }
2.        }
```

Set A:

1. Write a program to create following GUI using Swing.

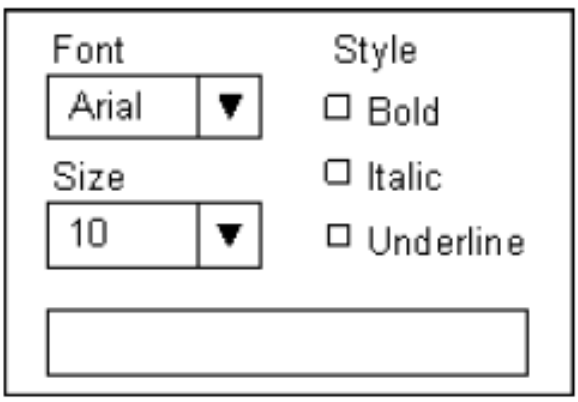


b)Write a java program to create the following GUI using Swing components.

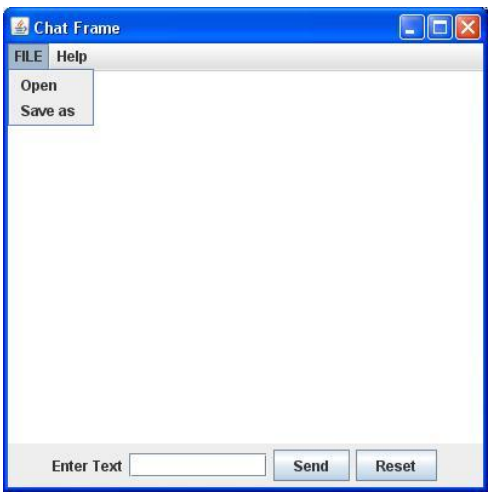


Set B

1. Write a program to create the following GUI.



2. Write a java program to design a following GUI (Use Swing).



Set C

Write a java program to design a following GUI (Use Swing).

Signature of the instructor:---------- Date:----------

Assignment Evaluation

0: Not Done [ ] 1: Incomplete [ ] 2: Late Complete [ ]

3: Needs Improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Assignment No. 7: Applet Programming

Objectives :

Applets are small java programs which are executed and displayed in a java compatible web browser.

Reading :

You should read the following topics before starting this exercise:

1. Applet tag,Applet class,Applet methods
2. Applet fundamentals, Applet lifecycle execution, Creating and running applets
3. Applets: Event handling using applets.

Ready Reference :

Creating an applet

All applets are subclasses of the **java.applet.Applet** class. You can also create an applet by extending the **javax.swing.JApplet** class.

The syntax is:

```
class MyApplet extends Applet
{
//applet methods
}
```

Applet methods:

Method	Purpose
init( )	Automatically called to perform initialization of the applet. Executed only once.
start( )	Called every time the applet moves into sight on the Web browser to allow the applet to start up its normal operations
stop( )	Called every time the applet moves out of sight on the Web browser to allow the applet to shut off expensive operations.
destroy( )	Called when the applet is being unloaded from the page to perform final release of resources when the applet is no longer used
paint()	Called each time the applets output needs to be redrawn

Running an applet

1. Compile the applet code using javac.
2. Use the java tool – appletviewer to view the applet (embed the APPLET tag in comments in the code)
3. Use the APPLET tag in an HTML page and load the applet in a browser

Using appletviewer:

1. Write the HTML APPLET tag in comments in the source file.
2. Compile the applet source code using javac.
3. Use appletviewer ClassName.class to view the applet.

Using browser:
1. Create an HTML file containing the APPLET tag.
2. Compile the applet source code using javac.
3. In the web browser, open the HTML file.

The APPLET tag
< APPLET
[CODEBASE = *appletURL*]
CODE = appletClassFile
[ALT = alternateText]
[ARCHIVE = archiveFile]
[NAME = appletInstanceName]
WIDTH = pixels
HEIGHT = pixels
[ALIGN = alignment]
[VSPACE = pixels]
[HSPACE = pixels]
>
[< **PARAM NAME** = *AttributeName* **VALUE** = *AttributeValue* />]
</**APPLET**>

Attribute	Value	Description
align	URL	*Deprecated* − Defines the text alignment around the applet
alt	URL	Alternate text to be displayed in case browser does not support applet
archive	URL	Applet path when it is stored in a Java Archive ie. jar file
code	URL	A URL that points to the class of the applet
codebase	URL	Indicates the base URL of the applet if the code attribute is relative
height	pixels	Height to display the applet
hspace	pixels	*Deprecated* − Defines the left and right spacing around the applet
name	name	Defines a unique name for the applet

object	name	Specifies the resource that contains a serialized representation of the applet's state.
title	test	Additional information to be displayed in tool tip of the mouse
vspace	pixels	*Deprecated* – Amount of white space to be inserted above and below the object.
width	pixels	Width to display the applet.

The mandatory attributes are CODE, HEIGHT and WIDTH.

Here is an example of a simple APPLET tag:

1. <applet code = MyApplet width=200 height=200 archive=”files.jar”></applet>
2. <applet code = “simple.class” width=200 height=200 codebase=”example/”> </applet>
3. <applet code="MyApplet.class" width=100 height=140></applet>

Passing parameters to applets
Param Tag :

The PARAM <param> tag is a sub tag of the <applet> tag. The <param> tag contains two attributes : *name* and *value* which are used to specify the name of the parameter and the value of the parameter respectively. For example, the param tags for passing name and age parameters looks as shown below:

<param name=”name” value=”Ramesh” />
<param name=”age” value=”25″ />

Now, these two parameters can be accessed in the applet program using the *getParameter()* method of the *Applet* class.

String getParameter(String param-name)
Example:
<APPLET NAME = “MyApplet.class” WIDTH = 100 HEIGHT = 100>
<PARAM NAME = “ImageSource” VALUE = “project/images/”>
<PARAM NAME = “BackgroundColor” VALUE = “0xc0c0c0”>
<PARAM NAME = “FontColor” VALUE = “Red”>
</APPLET>

Event Handling

Any program that uses GUI (graphical user interface) such as Java application written for windows, is event driven. Event describes the change in state of any object. **For Example :** Pressing a button, Entering a character in Textbox, Clicking or Dragging a mouse, etc.

Event handling has three main components,

- **Events :** An event is a change in state of an object.

- **Events Source :** Event source is an object that generates an event.
- **Listeners :** A listener is an object that listens to the event. A listener gets notified when an event occurs.

Events are supported by a number of Java packages, like **java.util**, **java.awt** and **java.awt.event**.

Important Event Classes and Interface

Event Classes	Description	Listener Interface
ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released and also when it enters or exit a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener
ItemEvent	generated when check-box or list item is clicked	ItemListener
TextEvent	generated when value of textarea or textfield is changed	TextListener
MouseWheelEvent	generated when mouse wheel is moved	MouseWheelListener
WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener
ComponentEvent	generated when component is hidden, moved, resized or set visible	ComponentEventListener
ContainerEvent	generated when component is added or removed from container	ContainerListener
AdjustmentEvent	generated when scroll bar is manipulated	AdjustmentListener
FocusEvent	generated when component gains or loses keyboard focus	FocusListener

Self Activity :

Steps to handle events:

1. Implement appropriate interface in the class.
2. Register the component with the listener.

1.Demo Example:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class EventApplet extends Applet implements ActionListener{
Button b;
TextField tf;

public void init(){
tf=new TextField();
tf.setBounds(30,40,150,20);

b=new Button("Click");
b.setBounds(80,150,60,50);

add(b);add(tf);
b.addActionListener(this);

setLayout(null);
}

 public void actionPerformed(ActionEvent e){
  tf.setText("Welcome");
 }
}
```

MyApplet.html

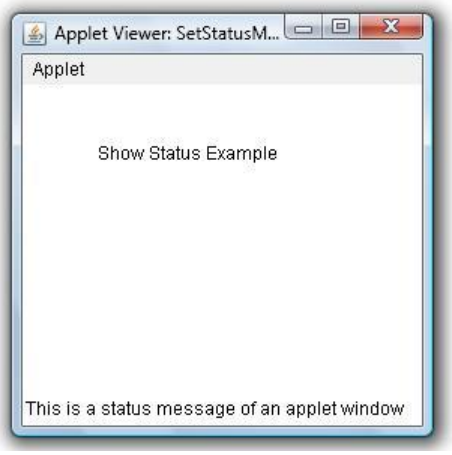
```
<html>
<body>
<applet code="EventApplet.class" width="300" height="300">
</applet>
</body>
</html>
```

Lab Assignments

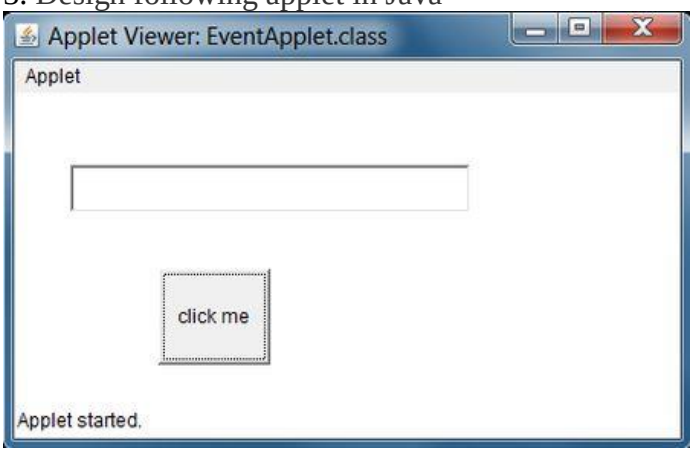
SET A

1. Create an applet to display a message at the center of the applet. The message is passed as a parameter to the applet.

2.Create an applet to set Status Message in Applet Window like

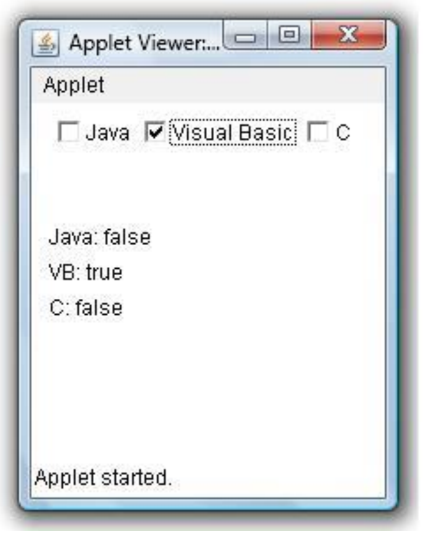


3. Design following applet in Java



SET B

1. Design following applet in Java and add checkbox event and display appropriate messages using label

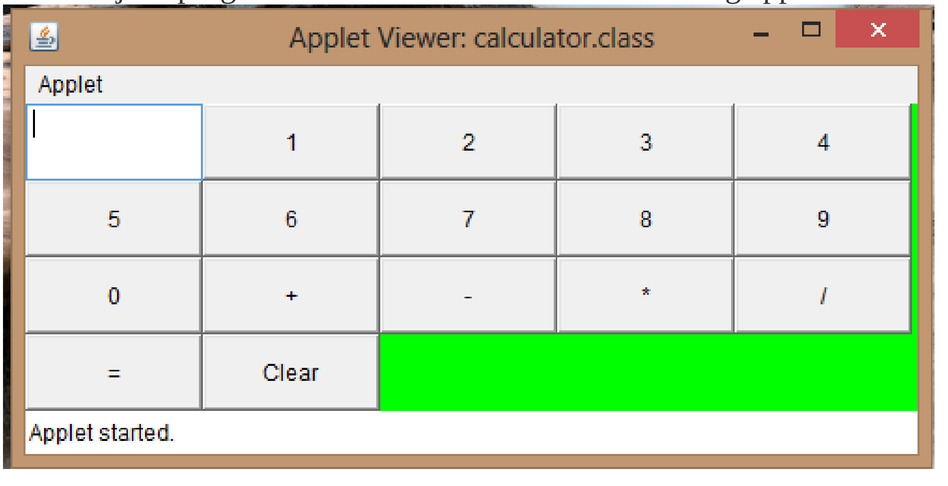


2.Design applet in Java to handle the Mouse Events such as MOUSE_MOVED and MOUSE_CLICK and display the position of the Mouse_Click in a TextField.

SET C

1 Write Java program to implement a shape selector from a given set of shapes(triangle, circle and line)

2. Write java program to make a Calculator in Java Using applet. Handle appropriate event.



Signature of the instructor:____ **Date:_________________________________**

Assignment Evaluation

0:Not Done [ ] **2:Late Complete [ ]** **4:Complete [ ]**

1:Incomplete[ ] **3:Needs Improvement [ ]** **5:Well Done[ ]**

Assignment 8: JDBC

JDBC

Java Database Connectivity (JDBC) API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database. JDBC works with Java on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

-JDBC Driver

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver-Type 1
2. Native-API driver (partially java driver)-Type 2
3. Network Protocol driver (fully java driver)-Type 3
4. Thin driver (fully java driver)-Type 4

   For databases Java uses **java.sql** package. This package contains following set of classes and interfaces.

Classes/interface	Description
java.sql.BLOB	Provide support for BLOB(Binary Large Object) SQL type.
java.sql.Connection	creates a connection with specific database
java.sql.CallableStatement	Execute stored procedures
java.sql.CLOB	Provide support for CLOB(Character Large Object) SQL type.
java.sql.Date	Provide support for Date SQL type.
java.sql.Driver	create an instance of a driver with the DriverManager.
java.sql.DriverManager	This class manages database drivers.
java.sql.PreparedStatement	Used to create and execute parameterized query.
java.sql.ResultSet	It is an interface that provide methods to access the result row-by-row.
java.sql.Savepoint	Specify savepoint in transaction.
java.sql.SQLException	Encapsulate all JDBC related exception.
java.sql.Statement	This interface is used to execute SQL statements.

For postgresql, use the driver:

org.postgresql.Driver

Steps to connect to the database in java

There are 5 steps to connect any java application with the database in java using JDBC. They are as follows:

- Register the driver class
- Creating connection
- Creating statement
- Executing queries
- Closing connection

1) Register the driver Class:

The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

Syntax of forName() method

public static void forName(String className)**throws** ClassNotFoundException

2) Create the connection object

The getConnection() method of DriverManager class is used to establish connection with the database.

Syntax of getConnection() method

1) public static Connection getConnection(String url)throws SQLException
2) public static Connection getConnection(String url,String name,String password) throws SQLException

Commonly used methods of Connection interface:

Methods	Description
createStatement()	Creates a statement object that can be used to execute SQL queries.
createStatement(int resultSetType, int resultSetConcurrency)	Creates a Statement object that will generate ResultSet objects with the given type and concurrency.
setAutoCommit(boolean status)	is used to set the commit status.By default it is true.
commit()	saves the changes made since the previous commit/rollback permanent
Rollback()	Drops all changes made since the previous commit/rollback.
Close()	closes the connection and Releases a JDBC resources immediately.

3) Create the Statement object

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax of createStatement() method

public Statement createStatement()**throws** SQLException

eg. Statement stmt=con.createStatement();

4) Execute the query

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax of executeQuery() method

public ResultSet executeQuery(String sql)**throws** SQLException

```
eg.  ResultSet rs=stmt.executeQuery("select * from tblemp");
 while(rs.next())
{
  System.out.println(rs.getInt(1)+" "+rs.getString(2));
}
```

5) Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax of close() method

public void close()throws SQLException

Major Classes and Interfaces of JDBC:
DriverManager Class:
This class will attempt to load the driver classes referenced in the jdbc.drivers system property.
Statement Interface:
Object of this interface is used for firing a static SQL or PL/SQL queries returning the results it has produced.
CallableStatement interface:
This interface is used to execute SQL stored procedures.
The stored procedures are a group of queries with variables in it.
Defined as:

CallableStatement cs=con.prepareCall();

The developer can prepare stored procedure for multiple uses and hence it allows variable declaration also.

Executing a SQL statement with the Statement object, and returning a jdbc resultSet.

To execute a query, call an execute method from Statement such as the following:

- **execute:** Use this method if the query could return one or more ResultSet objects.
- **executeQuery:** Returns one ResultSet object.

- **executeUpdate:** Returns an integer representing the number of rows affected by the SQL statement. Use this method if you are using INSERT, DELETE, or UPDATE SQL statements.

ResultSet interface:
In case of select query, it returns set of selected records, a ReultSet object maintains cursor pointing to its current row of data.
Defined as:

public interface ResultSet extends Wrapper, AutoCloseable

Methods of this interface:

Method	Description
boolean absolute(int row)	Mopves the cursor to the specified row number. Returns false if no such row is found.
void afterLast()	Moves the cursor to the end of these ResultSet object, i.e. just after the last row.
void beforeFirst()	Moves the cursor to the front of this Resultset object, i.e. just before the first row.
void close()	Immediately releases invoking ResultSet object's database and JDBC resources.
void deleteRow()	Deletes the current row from invoking Resultset object and also from the database.

MetaData

MetaData is data of data, i.e. we can get further information from the data.
(1) In case of ResultSet, we have metadata of retrieved ResultSet called as **ResultSetMetaData**
(2) In case of Database, we have metadata of database called as **DatabaseMetaData.**

1) ResultsetMetaData Interface:
This interface provides an object that can be used to get information about the types and properties of the columns in a ResultSet object.
Defined as:

public interface ResultSetMetaData extends Wrapper

Methods of this interface:

Method	Description
int getColumnCount()	Retuirns the number of columns available in ResultSet object.
String getColumnType(int col_num)	Returns the SQL type of the column.
getPrecision(int col_num)	Returns the max size for the column.
Boolean isAutoIncrement(int col_num)	Checks & returns whether the specified

	column is set for auto increment or not.
String getTableName(int col_num)	Returns the table name to which the specified column belongs.
boolean isNullable**()**	Checks and returns whether the specified column is allowed to keep null or not.
boolean isReadOnly()	Checks and returns whether specified column is read only or not.

2) DatabaseMetaData Interface:
An instance of this interface provides comprehensive information about the database.
Defined as:

public interface DatabaseMetaData extends Wrapper

Methods of this interface:

Method	**Description**
boolean allProceduresAreCallable()	Retrieves whether all the stored procedures are callable or not.
Boolean allTablesAreSelectable()	Retrieves whether the user can fire SELECT query on all connected tables.
int getDatabaseMinorVersion()	Retrives the minor version of connected database.
int getDatabaseMajorVersion**()**	Retrives the major version of connected database.
String getDatabaseProductName()	Retrieves the name of this database product.
String getDatabaseProductVersion()	Retrieves the product version of the connected database.

ResultSetInterface:-

Method	**Description**
boolean next()	is used to move the cursor to the one row next from the current position.
boolean previous()	is used to move the cursor to the one row previous from the current position.
boolean first()	is used to move the cursor to the first row in result set object.

boolean last()	is used to move the cursor to the last row in result set object.
boolean absolute(int row)	is used to move the cursor to the specified row number in the ResultSet object.
boolean relative(int row)	is used to move the cursor to the relative row number in the ResultSet object, it may be positive or negative.

Sample Programs:- 1.Program to insert data into employee table

```
import java.sql.*;
class Demo
{
public static void main(String args[])
{
try
{
Class.forName("org.postgresql.Driver");
Connection
con=DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/dbbca","postgres","123");
Statement stmt=con.createStatement();
stmt.executeUpdate("insert into tblemp(uname,upass)values('admin','admin')");
stmt.close();
con.close();
}
catch(Exception ex)
{
  System.out.println(ex);
}
}
}
```

2. Program to update data from employee table

```
import java.sql.*;
class Demoup
{
public static void main(String args[])
{
try
{
Class.forName("org.postgresql.Driver");
Connection
con=DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/dbbca","postgres","123");
Statement stmt=con.createStatement();
stmt.executeUpdate("update tblemp set uname='swapnil' where uname='admin'");
stmt.close();
con.close();
}
catch(Exception ex)
{
 System.out.println(ex);
```

```
}
}
}
```

3. Program to insert data into employee table

```
import java.sql.*;
    class Demodelete
    {
    public static void main(String args[])
    {
    try
    {
    Class.forName("org.postgresql.Driver");
    Connection
    con=DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/dbbca","postgres","12
    3");
Statement stmt=con.createStatement();
    stmt.executeUpdate("delete from tblemp where id=1");
stmt.close();
con.close();
}
catch(Exception ex)
{
System.out.println(ex);
}
}
}
```

4. Program to display details of employee.

```
import java.sql.*;
class Demodisp
{
public static void main(String args[])
{
try
{
Class.forName("org.postgresql.Driver");
Connection
con=DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/dbbca","postgres","123");
Statement stmt=con.createStatement();
ResultSet rs;
rs=stmt.executeQuery("select * from tblemp");
while(rs.next())
{
System.out.println("Name="+rs.getString("uname"));
}
stmt.close();
con.close();
}
catch(Exception ex)
{
  System.out.println(ex);
}
```

```
}
}
```

5.Program to display information of employee(id,name,salary).

```
import java.sql.*;
import java.io.*;
 class JDBCDemo
 {
  public static void main(String[] args) throws SQLException
 {
  Class.forName("org.postgresql.Driver");
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
    try
     {
          Connection
          con=DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/dbbca","postgres","12
          3");
      if(conn==null)
   System.out.println("Connection failed ");
   else
    {
      System.out.println(“Connection successful..”);
     stmt = conn.createStatement();
      rs = stmt.executeQuery("Select * from emp");
   while(rs.next())
   {
       System.out.print("ID = " + rs.getInt(1));
       System.out.println("Name = " + rs.getString(2));
       System.out.println("Salary = " + rs.getInt(3));
 }
 conn.close();
 }
}
  catch(Exception ex)
   {
    System.out.println(ex);
   }
  }
}// end of class
```

SET A

Lab Assignments

1. Create a student table with fields roll number, name, percentage. Insert values in the table. Display all the details of the student table in a tabular format on the screen (using swing).

2. Write a program to display information about the database and list all the tables in the database. (Use DatabaseMetaData).

SET B

1. Write a program to display information about all columns in the student table. (Use ResultSetMetaData).

2. Write a menu driven program (Command line interface) to perform the following operations on student table. 1. Insert 2. Modify 3. Delete 4. Search 5. View All 6. Exit

SET C

1. Create a table student with fields roll number, name, percentage using Postgresql. Insert values in the table. Display all the details of the student table in a tabular format on the screen. (Using Swing)

Signature of the instructor:_ _____ Date: _____

Assignment Evaluation

0:Not Done [ ] **2:Late Complete [ ]** **3.4:Complete [ ]**

1:Incomplete[ ]

Assignment 9: Servlets

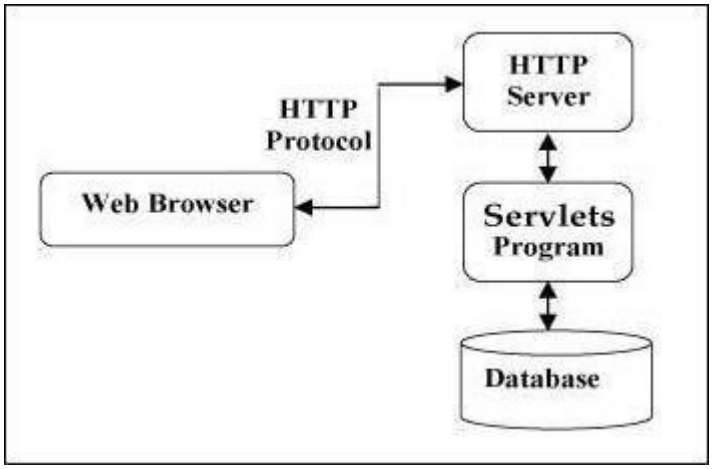
What are Servlets?

Java Servlets are programs that run on a Web or Application server and act as a middle layer between a requests coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

Servlets Architecture

The following diagram shows the position of Servlets in a Web Application.



Servlets Packages

Java Servlets are Java classes run by a web server that has an interpreter that supports the Java Servlet specification.

Servlets can be created using the **javax.servlet** and **javax.servlet.http**packages, which are a standard part of the Java's enterprise edition, an expanded version of the Java class library that supports large-scale development projects.

How to Create a Java Servlet Using Tomcat and Linux **Install and Configure Tomcat on Linux**

Visit http://tomcat.apache.org/ and download a binary core archive of the latest stable version. To install Tomcat extract the archive at appropriate location.

Example

[kamil@localhost Download]$ wget http://apache.skknet.net/tomcat/tomcat-6/v6.0.20/bin/apache-tomcat-6.0.20.tar.gz
[kamil@localhost Download]$ tar -xzf apache-tomcat-6.0.20.tar.gz
By default the Tomcat server is using port 8080. To change this or any other setting edit the XML configuration files at conf directory. To add a user with rights for admistration of the Tomcat server open file tomcat-users.xml and modify it as it is shown:

```
<tomcat-users>
        <user username="admin" password="secret"
roles="admin,manager"/>
</tomcat-users>
```

The example demonstrates how to add an user with name 'admin' and password 'secret'. The Tomcat server has to be restarted in order to recognize the new user.

Creating a Servlet

The servlet is a JAVA server application which dynamically process requests and construct responses. To create a HTTP servelet follow five simple steps:

1. Create Directory Structure

Enter Tomcat's main directory and navigate to webapps. Create a new directory for your servlet using the mkdir command. After this create directoris WEB-INF and classes (subdirectory of WEB-INF). The WEB-INF directory contains configuration information about the web application. The subdirectory class should include the source code of the servlet.

```
[kamil@localhost apache-tomcat-6.0.20]$ cd webapps/
[kamil@localhost webapps]$ mkdir hello [kamil@localhost
webapps]$ cd hello [kamil@localhost hello]$ mkdir WEB-INF
[kamil@localhost hello]$ cd WEB-INF [kamil@localhost
WEB-INF]$ mkdir classes
```

2. Configuration

The servlet must have a configuration XML file at directory WEB-INF. Create a text file with name web.xml. The file have to uses the XML schema. The name of the root tag have to be web-app and should contain subelements servlet (contains information about name and main class of the servlet) and servlet-mapping (URL pattern to execute the servlet). Example configration:

```
<web-app>
        <servlet>
                <servlet-name>HelloWorld</servlet-name>
                <servlet-class>HelloWorld</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>HelloWorld</servlet-name> <url-
                pattern>/HelloWorld</url-pattern>
        </servlet-mapping>
</web-app>
```

By default the Tomcat uses port 8080 and this servlet can be access with URL: http://localhost:8080/HelloWorld.

3. The Code

The name of the main class must be same as described at web.xml (configured at step2) and should be saved as a file at directory classes (created at step 1). The class should extend the abstact class HttpServlet and to override method doGet. The packages for input/output operations, servlet exception and HTTP request/responses have to be included. Example shows how to return HTML as a response:

```
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
        static final long serialVersionUID = 42L; public void
        doGet(HttpServletRequest request,
                        HttpServletResponse response)
                throws IOException, ServletException
        {
                response.setContentType("text/html");
                PrintWriter Printer = response.getWriter();
                Printer.println("<html>");
                Printer.println("<head>");
                Printer.println("\t<title>Kamil Khan Example</title>");
                Printer.println("</head>");
                Printer.println("<body>");
                Printer.println("<b>Hello World!</b>");
                Printer.println("</body>");
                Printer.println("</html>");
                return;
        }
}
```

Please note the static field serialVersionUID which should be declared because the main class is serializable and in order to avoid the warning:

```
    WARNING in HelloWorld.java (at line 5) public class
        HelloWorld extends HttpServlet
                    ^^^^^^^^^^
The serializable class HelloWorld does not declare a static final
serialVersionUID field of type long ----------
```

4. Build the Servlet

Before executing a servlet it has to be build using the source code, j2ee.jar and servlet-api.jar. The Servlet API is part of the of Tomcat and can be found at lib (subdirectory of the Tomcat main directory). To build the class HelloWorld, created at the previos steps under console execute:

[kamil@localhost classes]$ javac -classpath %J2EE_HOME%\lib\j2ee.jar - classpath ../../../../lib/servlet-api.jar HelloWorld.java [kamil@localhost classes]$
On success no warnings or errors will occur.

5. Run the Servlet

Start Tomcat server (if not started) and using a web browser open the URL configured at step 2 (example: http://localhost:8080/HelloWorld).

[kamil@localhost bin]$ ./startup.sh
Using CATALINA_BASE: /home/kamil/apache-tomcat-6.0.20
Using CATALINA_HOME: /home/kamil/apache-tomcat-6.0.20
Using CATALINA_TMPDIR: /home/kamil/apache-tomcat-6.0.20/temp
Using JRE_HOME: /usr

SERVLET LIFE CYCLE

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet

- The servlet is initialized by calling the init () method.
- The servlet calls service() method to process a client's request.
- The servlet is terminated by calling the destroy() method.
- Finally, servlet is garbage collected by the garbage collector of the JVM.

The Init() Method

When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.

The init method definition looks like this:

```
public void init() throws ServletException { //
   Initialization code...
}
```

The Service() Method

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

Here is the signature of this method:

```
public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException{


}
```

The doGet() Method

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

```
public void doGet(HttpServletRequest request,HttpServletResponse response)
      throws ServletException, IOException { //
      Servlet code }
```

The doPost() Method

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
      // Servlet code
}
```

After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this:

```
        public void destroy() {
          // Finalization code...
    }
```

Sample Code

- Import required java libraries

```
import java.io.*;
import javax.servlet.*; import
javax.servlet.http.*;
```

- Extend HttpServlet class

```
public class HelloWorld extends HttpServlet {

private String message;
public void init() throws ServletException
{
1)      Do required initialization message = "Arun Gangarde";
}

public void doGet(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException
{
//      Set response content type response.setContentType("text/html");
//      Actual logic goes here.
PrintWriter out = response.getWriter();
out.println("<h1>" + message + "</h1>"); }
public void destroy()
{
//  do nothing.
}
}
```

Reading Form Data using Servlet

Servlets handles form data parsing automatically using the following methods depending on the situation:

- **getParameter():** You call request.getParameter() method to get the value of a form parameter.
- **getParameterValues():** Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
- **getParameterNames():** Call this method if you want a complete list of all parameters in the current request.

GET Method Example using URL

Here is a simple URL which will pass two values to HelloForm program using GET method.

http://localhost:8080/HelloForm?first_name=KAMIL &last_name=KHAN

Given below is the **HelloForm.java** servlet program to handle input given by web browser. We are going to use **getParameter()** method which makes it very easy to access passed information:

```
//      Import required java libraries
import java.io.*;
import javax.servlet.*; import
javax.servlet.http.*;
//      Extend HttpServlet class
public class HelloForm extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
//      Set response content type response.setContentType("text/html");
PrintWriter out = response.getWriter();
String title = "Using GET Method to Read Form Data";

String docType ="<!doctype html public \"-//w3c//dtd html 4.0 " +
"transitional//en\">\n";

out.println(docType +"<html>\n" +"<head><title>" + title +
"</title></head>\n" + "<body bgcolor=\"#f0f0f0\">\n" +"<h1
align=\"center\">" + title + "</h1>\n" + "<ul>\n" +
"      <li><b>First Name</b>: "+ request.getParameter("first_name") + "\n" + "
<li><b>Last Name</b>: "+ request.getParameter("last_name") + "\n" +
"</ul>\n" +"</body></html>");
    }
  }
```

Assuming your environment is set up properly, compile HelloForm.java as follows:

```
$ javac HelloForm.java
```

If everything goes fine, above compilation would produce **HelloForm.class** file. Next you would have to copy this class file in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/classes and create following entries in **web.xml** file located in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/

```
<servlet>
    <servlet-name>HelloForm</servlet-name>
    <servlet-class>HelloForm</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>HelloForm</servlet-name> <url-
    pattern>/HelloForm</url-pattern>
</servlet-mapping>
```

Now type *http://localhost:8080/HelloForm?first_name=ZARA&last_name=ALI* in your browser's Location:box and make sure you already started tomcat server, before firing above command in the browser. This would generate following result:

Using GET Method to Read Form Data First

Name: KAMIL

Last Name: KHAN

HTTP Header Request Example

Following is the example which uses **getHeaderNames()** method of HttpServletRequest to read the HTTP header information. This method returns an Enumeration that contains the header information associated with the current HTTP request. Once we have an Enumeration, we can loop down the Enumeration in the standard manner, using *hasMoreElements()* method to determine when to stop and using *nextElement()* method to get each parameter name.

There are following methods which can be used to read HTTP header in your servlet program. These methods are available with *HttpServletRequest* object.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

// Extend HttpServlet class
public class DisplayHeader extends HttpServlet {

//         Method to handle GET method request. public void
doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
{ //   Set response content type response.setContentType("text/html");

PrintWriter out = response.getWriter();
String title = "HTTP Header Request Example"; String
docType =
"<!doctype html public \"-//w3c//dtd html 4.0 " +
"transitional//en\">\n";
out.println(docType +
"<html>\n" +
"<head><title>" + title + "</title></head>\n"+ "<body
bgcolor=\"#f0f0f0\">\n" +
"<h1 align=\"center\">" + title + "</h1>\n" +
"<table width=\"100%\" border=\"1\" align=\"center\">\n" + "<tr
bgcolor=\"#949494\">\n" +
"<th>Header Name</th><th>Header Value(s)</th>\n"+ "</tr>\n");

Enumeration headerNames = request.getHeaderNames();

while(headerNames.hasMoreElements()) {
String paramName = (String)headerNames.nextElement();
out.print("<tr><td>" + paramName + "</td>\n");
String paramValue = request.getHeader(paramName);
out.println("<td> " + paramValue + "</td></tr>\n"); }
out.println("</table>\n</body></html>");
}// Method to handle POST method request. public void
doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
doGet(request, response); }}
```

HTTP Header Response – Example

You already have seen setContentType() method working in previous examples and following example would also use same method, additionally we would use

setIntHeader() method to set Refresh header.

```
//  Import required java libraries
import java.io.*;
 import javax.servlet.*; import
javax.servlet.http.*; import
java.util.*; //Extend HttpServlet
class

public class Refresh extends HttpServlet {

//Method to handle GET method request.
public void doGet(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException
{
//Set refresh, autoload time as 5 seconds response.setIntHeader("Refresh", 5); Set
response content type response.setContentType("text/html"); // Get current time
Calendar calendar = new GregorianCalendar(); String
am_pm;

int hour = calendar.get(Calendar.HOUR);
int minute = calendar.get(Calendar.MINUTE);
 int second = calendar.get(Calendar.SECOND);
if(calendar.get(Calendar.AM_PM) == 0)
am_pm = "AM";
else
am_pm = "PM";

String CT = hour+":"+ minute +":"+ second +" "+ am_pm;
PrintWriter out = response.getWriter();

String title = "Auto Refresh Header Setting"; String
docType =
"<!doctype html public \"-//w3c//dtd html 4.0 " +"transitional//en\">\n";
out.println(docType +"<html>\n" +"<head><title>" + title + "</title></head>\n"+
"<body bgcolor=\"#f0f0f0\">\n" +"<h1 align=\"center\">" + title + "</h1>\n" +
"<p>Current Time is: " + CT + "</p>\n");

}
```

```
//Method to handle POST method request.
 public void doPost(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException {
doGet(request, response);

}

}
```

SET A

1) Write a SERVLET program in java to display current Date and Time on Browser.

2) Write a SERVLET program in java to accept details of student (SeatNo, Stud_Name, Class, Total_Marks). Calculate percentage and grade obtained and display details on page.

3) Write a SERVLET application to accept username and password, search them into database, if found then display appropriate message on the browser otherwise display error message.

4) Write SERVLET Program in java to accept number from user and display Fibonacci series.

SET B

1) Write a SERVLET program to Design an HTML page containing 4 option buttons (Painting, Drawing, Singing and swimming) and 2 buttons reset and submit. When the user clicks submit, the server responds by adding cookie containing the selected hobby and sends the HTML page to the client. Program should not allow duplicate cookies to be written.

2) Write a SERVLET program to change inactive time interval of session.

3) Write a SERVLET program which counts how many times a user has visited a web page. If user is visiting the page for the first time, display a welcome message. If the user is revisiting the page, display the number of times visited. (Use cookies)

4) Write a servlet program to display the details of Product (ProdCode, PName, Price) on the browser in tabular format. (Use database)

SET C

1) Write a SERVLET program that provides information about a HTTP request from a client, such as IP address and browser type. The servlet also provides information about the server on which the servlet is running, such as the operating system type, and the names of currently loaded servlets.

Signature of the instructor:_ _____ Date: _____

Assignment Evaluation

0: Not Done [ ] 1: Incomplete [ ] 2: Late Complete [ ]

3: Needs Improvement [ ]
4: complete[]
5:Well Done[]

Assignment 10: JSP

JavaServer Pages (JSPs) are similar to HTML files, but provide the ability to display dynamic content within Web pages. JSP technology was developed by Sun Microsystems to separate the development of dynamic Web page content from static HTML page design.

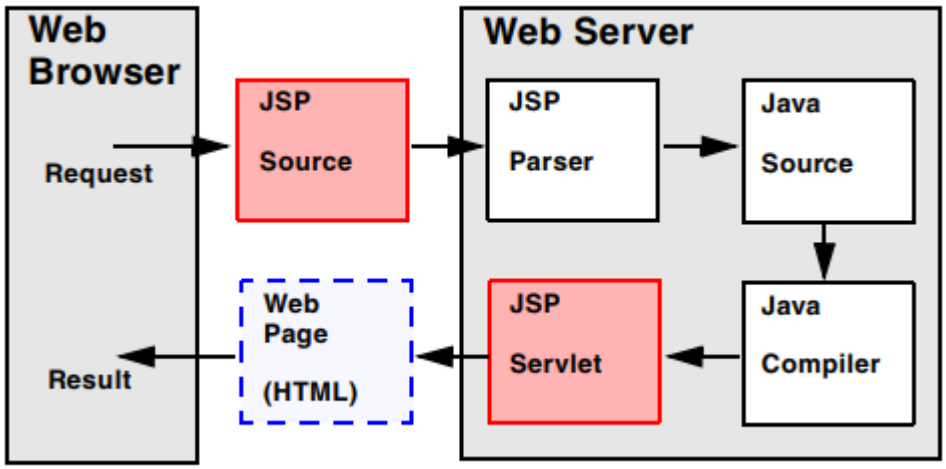
How JavaServer Pages work

JavaServer Pages are made operable by having their contents (HTML tags, JSP tags and scripts) translated into a servlet by the application server. This process is responsible for translating both the dynamic and static elements declared within the JSP file into Java servlet code that delivers the translated contents through the Web server output stream to the browser.

Because JSPs are server-side technology, the processing of both the static and dynamic elements of the page occurs in the server. The architecture of a JSP/servlet-enabled Web site is often referred to as thin-client because most of the business logic is executed on the server.

The following process outlines the tasks performed on a JSP file on the first invocation of the file or when the underlying JSP file is changed by the developer:

- The Web browser makes a request to the JSP page.
- The JSP engine parses the contents of the JSP file.
- The JSP engine creates temporary servlet source code based on the contents of the JSP. The generated servlet is responsible for rendering the static elements of the JSP specified at design time in addition to creating the dynamic elements of the page.
- The servlet source code is compiled by the Java compiler into a servlet class file.
- The servlet is instantiated. The init and service methods of the servlet are called, and the servlet logic is executed.
- The combination of static HTML and graphics combined with the dynamic elements specified in the original JSP page definition are sent to the Web browser through the output stream of the servlet's response object.



The JSP processing life-cycle on first-time invocation

Components of JavaServer Pages

JavaServer Pages are composed of standard HTML tags and JSP tags. The available JSP tags defined in the JSP 1.0 specification are categorized as follows:

- Directives
- Declarations
- Scriptlets
- Comments
- Expressions

This section describes each of these categories in more detail.

HTML tags

JavaServer Pages support all HTML tags. For a listing of HTML tags, refer to your HTML manual.

JSP directives

A JSP directive is a global definition sent to the JSP engine that remains valid regardless of any specific requests made to the JSP page. A directive always appears at the top of the JSP file, before any other JSP tags. This is due to the way the JSP parsing engine produces servlet code from the JSP file.

The syntax of a directive is:

<%@ directive directive_attr_name = value %>

Directives are grouped as follows:

page

The *page* directive defines page dependent attributes to the JSP engine.
Table 1. Attributes of the page directive

Attribute Name	Description
language	Identifies the scripting language used in scriptlets in the JSP file or any of its included files. JSP supports only the value of “java”. WebSphere extensions provide support for other scripting languages. <%@ page language = "java" %>
extends	The fully-qualified name of the superclass for which this JSP page will be derived. Using this attribute can effect the JSP engine’s ability to select specialized superclasses based on the JSP file content, and should be used with care.
import	When the language attribute of "java" is defined, the import attribute specifies the additional files containing the types used within the scripting environment. <%@ page import = "java.util.*" %>
session "true" | "false"	If *true*, specifies that the page will participate in an HTTP session and enables the JSP file access to the implicit session object. The default value is *true*.
buffer "none" | "sizekb"	Indicates the buffer size for the JspWriter. If *none*, the output from the JSP is written directly to the ServletResponse PrintWriter object. Any other value results in the JspWriter buffering the output up to the specified size. The buffer is flushed in accordance with the value of the autoFlush attribute. The default buffer size is no less than 8kb.
autoFlush "true" | "false"	If *true*, the buffer will be flushed automatically. If false, an exception is raised when the buffer becomes full. The default value is *true*.
isThreadSafe "true" | "false"	If *true*, the JSP processor may send multiple outstanding client requests to the page concurrently. If *false*, the JSP processor sends outstanding client requests to the page consecutively, in the same order in which they were received. The default is *true*.
info	Allows the definition of a string value that can be retrieved using Servlet.getServletInfo().
errorPage	Specifies the URL to be directed to for error handling if an exception is thrown and not caught within the page implementation. In the JSP 1.0 specification, this URL must point to a JSP page.
isErrorPage "true" | "false"	Identifies that the JSP page refers to a URL identified in another JSP’s errorPage attribute. When this value is *true*, the implicit variable *exception* is defined, and its value set to reference the Throwable object of the JSP source file which causes the error.

include

The *include* directive allows substitution of text or code to occur at translation time. You can use the include directive to provide a standard header on each JSP page, for example:

```
<%@ include file="copyright.html" %>
```

The include directive has the attributes shown in Table 2.

Table 2. Attributes for the include directive

Attribute Name	**Description**
file	Directs the JSP engine to substitute the text or code specified by file or URL reference. The URL reference can be another JSP file.

taglib

The *taglib* directive allows custom extensions to be made to the tags known to the JSP engine. This tag is an advanced feature. Refer to the Sun JavaServer Page 1.0 specification for more information about this tag.

Declarations

A declaration block contains Java variables and methods that are called from an *expression* block within the JSP file. Code within a declaration block is usually written in Java, however, the WebSphere application server supports declaration blocks containing other script syntax. Code within a declaration block is often used to perform additional processing on the dynamic data generated by a JavaBean property.
The syntax of a declaration is:

```
<%! declaration(s) %>
```

For example:

```
<%!
    private int getDateCount = 0;
    private String getDate(GregorianCalendar gc1)
        { ...method body here...}
%>
```

Scriptlets

JSP supports embedding of Java code fragments within a JSP by using a *scriptlet* block. Scriptlets are used to embed small code blocks within the JSP page, rather than to declare entire methods as performed in a declarations block. The syntax for a scriptlet is:

<% scriptlet %>

The following example uses a scriptlet to output an HTML message based on the time of day. You can see that the HTML elements appear *outside* the script declarations.

```
<% if (Calendar.getInstance().get(Calendar.AM_PM)
== Calendar.AM)
   {%>
      How are you this morning ?
<% } else
   { %>
      How are you this afternoon ?
<% } %>
```

Comments

You can use two types of comments within a JSP. The first comment style, known as an *output comment,* enables the comment to appear in the output stream on the browser. This comment is an HTML formatted comment whose syntax is:

<!-- comments ... -->

The second comment style is used to fully exclude the commented block from the output and is commonly used when uncommenting a block of code that so that the commented block is never delivered to the browser. The syntax is:

<%-- comment text --%>

You can also create comments containing dynamic content by embedding a scriptlet tag inside a comment tag. For example:

<!-- comment text <%= expression %> more comment text ->

Expressions

Expressions are scriptlet fragments whose results can be converted to String objects and subsequently fed to the output stream for display in a browser. The syntax for an expression is:

<%= expression %>

The following example calls the *incrementCounter* method declared in the declarations block and prints the result.

<%= incrementCounter() %>

All primitive types such as short, int, and long can be automatically converted to Strings. Your own classes must provide a *toString* method for String conversion.

```
<html>
<title>Date Display</title>
<body>

<!-- D I R E C T I V E S -->
<%@ page language = "java" %>
<%@ page import = "java.util.*" %>
<%@ page contentType = "TEXT/HTML" %>
<!-- S C R I P T L E T S-->
<H3>
<% if (Calendar.getInstance().get(Calendar.AM_PM) == Calendar.AM)
   {%>
   How are you this morning,
   <%} else {%>
```

```
    How are you this afternoon,
    <% }
%>
WebSphere 3 User ?
</H3>
<HR>
<!-- accessing  implicit objects -->
<% out.println("Here is the <b>Date DisplayJSP</b>");
%>
<!-- DECLARATIONS -->
<%!
private int calledCount = 0;
private String getDate(GregorianCalendar gcalendar) {
    StringBuffer dateStr = new StringBuffer();
    dateStr.append(gcalendar.get(Calendar.DATE));
    dateStr.append("/");
    dateStr.append(gcalendar.get(Calendar.MONTH) + 1);
    dateStr.append("/");
    dateStr.append(gcalendar.get(Calendar.YEAR));
    return (dateStr.toString());
}

private int incrementCounter() {
    return (++calledCount);
}

%>
<H1> Today's Date is: <%= getDate(new GregorianCalendar()) %> </H1>
<H1> This page has been called: <%= incrementCounter() %>
time(s)</H1> </body>
</html>
```

Calling a servlet from a JSP

You can invoke a servlet from a JSP either as an action on a form, or directly through the *jsp:include* or *jsp:forward* tags.

Form action

Typically, you want to call a servlet as a result of an action performed on a JavaServer Page. For example, you may want to process some data entered by the user in an HTML form when they click on the *Submit* button.

To invoke a servlet within the HTML <FORM> tag, the syntax is:

<FORM METHOD="POST|GET" ACTION="*application_URI*/*JSP_URL*">
<!-- Other tags such as text boxes and buttons go here -->
</FORM>

For example:

```
<HTML>
<HEAD> <TITLE> Call Servlet from JSP </TITLE> </HEAD>
<CENTER>
<H1> Call Servlet from JSP </H1>
<FORM                                                    method="POST"
action="/itsoservjsp/servlet/itso.servjsp.jspsamples.DateDisplayServlet">

<H2> DateDisplay Servlet Launcher </H2>Click the button below to display the current date
<P> <INPUT type="submit" name="CALL_SERVLET" value="Call the Servlet">
</FORM>
</CENTER>
</BODY></HTML>
```

JSP include tag

You can include the output of a servlet in a JSP using the jsp.include tag:
<jsp:include page="/servlet/itso.servjsp.servletapi.SHTMLServlet" />

```
<HTML><BODY>
<H2> JSP to Servlet </H2>
<HR>
<jsp:include
page="/servlet/itso.servjsp.servletapi.SHTMLServlet"  />
<HR>

<H2>End of servlet include</H2>
</HTML></BODY>
```

When you run this JSP the output of the servlet is imbedded in the JSP output.

JSP forward tag

You can forward processing from a JSP to a servlet using the jsp.forward tag:
<jsp:forward page="/servlet/itso.servjsp.servletapi.SHTMLServlet" />

```
<HTML><BODY>
<H2> JSP to Servlet </H2>
<HR>
<jsp:forward
page="/servlet/itso.servjsp.servletapi.SHTMLServlet"  />
<HR>

<H2>End of servlet include</H2>
</HTML></BODY>
```

Calling a JSP from a servlet

DateDisplayServlet's *doPost* method, which is called when the *Submit* button is clicked. The servlet simply calls the *sendRedirect* method of the HttpServletResponse object, directing the response to the *DateDisplay.jsp*. This example simply demonstrates the redirection capability of the response object. In reality, the doPost method could invoke other methods which process the form data, instantiate other beans that perform the business logic, and finally redirect the user to the JSP.

```
import javax.servlet.http.*;
```

```
public class DateDisplayServlet extends HttpServlet {

public void doPost(HttpServletRequest req, HttpServletResponse
resp)
                throws javax.servlet.ServletException,
                java.io.IOException {
      Redirect to the
   DateDisplay JSP page
   resp.sendRedirect("/Date
   Display.jsp");

      alternate call to JSP
      getServletContext().getRequestDispatcher("/DateDisplay.jsp
      ").forward(req,resp);
}
```

DateDisplayServlet demonstrating simple redirection

You can also use the RequestDispatcher object (see "Request dispatching" on page 81) to invoke a JSP:

```
getServletContext().getRequestDispatcher("/DateDisplay.jsp").forward(req,
resp);
```

SET A

1) Write a JSP script to accept username, store it into the session, compare it with password in another jsp file, if username matches with password then display appropriate message in html file.

2) Write a JSP program to create a shopping mall. User must be allowed to do purchase from two pages. Each page should have a page total. The third page should display a bill, which consists of a page total of whatever the purchase has been done and print the total

3) Create a JSP page to accept a number from an user and display it in words:
   Example: 123 – One Two Three.

3) Write a JSP program to check whether given number is Armstrong or not. (Use Include directive).

SET B

1) Write a JSP program to accept the details of Account (ANo, Type, Bal) and store it into database and display it in tabular form. (Use PreparedStatement interface)
2) Write a JSP page, which accepts user name in a text box and greets the user according to the time on server machine.
3) Write a JSP program to display the details of Hospital (HNo, HName, Address) in tabular form on browser
4) Write a JSP script to accept the details of Student (RNo, SName, Gender, Comp_Know , Class) and display it on the browser. Use appropriate controls for accepting data.

SET C

1) Write a JSP script to check whether given mail ID is valid or not. (Mail ID should contain one @ symbol)
2) Write a JSP script to accept UserName and his NickName through html page and then displays username when visit count to page is odd and displays nickname when the visit count to the page is even.

Signature of the instructor:_ _____ Date: _____

Assignment Evaluation

0:Not Done [ ]	**2:Late Complete [ ]**	**4:Complete [ ]**
1:Incomplete[ ]	**3:Needs Improvement [ ]**	**5:Well Done[ ]**

Savitribai Phule Pune University

Section-II

T. Y. B. C. A. (Science)

SEMESTER V

BCA - 506

Lab Course – II

Advanced Web Technology

Assignments

EDITORS:

Section-II: Advanced Web Technology

- Mrs. Veena Kiran Gandhi
- Mr. Salauddin Sajjan
- Mrs. Summaiya Tamboli
- Mr. Mohsin Tamboli
- Ms. Farzana Peer Mohammed

Table of Contents for Section-II

Assignment Completion Sheet

Lab Course II			
Advanced Web Technology Assignments			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teachers Sign**
1	Introduction to Classes & objects		
2	Inheritance ,Interface and Introspection		
3	PHP Global Variables & Form Methods		
4	Self Referencing Pages, Sticky forms with Multi-value Parameters		
5	File uploads and Form Validation		
6	Cookies &Sessions		
7	Database		
8	XML		
9	AJAX		
Total (Out of 45)			
Total (Out of 10)			

This is to certify that Mr./Ms._____

Has successfully completed the Advanced Web Technology course work for Lab Course II and has scored___ Marks out of 30.

Instructor **H.O.D / Coordinator**

Internal Examiner External Examiner

Section-II: Advanced Web Technology

Assignment 1: Introduction to Object Oriented Programming in PHP

By Ms. Farzana Peer Mohammed

Introduction:

Object-Oriented Programming (OOP) is a programming model that is based on the concept of classes and objects. As opposed to procedural programming where the focus is on writing procedures or functions that perform operations on the data, in object-oriented programming the focus is on the creations of objects which contain both data and functions together.

Object Oriented Concepts:

Before we go in detail, let's define important terms related to Object Oriented Programming.

- **Class:** Class is a programmer-defined data type, which includes local methods and local variables. A class may contain its own constants, variables (called "properties"), and functions (called "methods").
- **Object:** An individual instance of data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.
- **Member Variable**: These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attribute of the object once an object is created.
- **Member function**: These are the function defined inside a class and are used to access object data.
- **Constructor:** Constructor Functions are special type of functions which are called automatically whenever an object is created. It is a special function that initializes the properties of the class.
- **Destructor:** Like a constructor function you can define a destructor function using function __destruct (). You can release all the resources with-in a destructor.
- **Encapsulation:** Encapsulation means hiding or wrapping the code and data into a single unit to protect the data from outside world. It is used to protect class's internal data (properties and method) from code outside that class and hiding details of implementation. In PHP, encapsulation is provided by visibility specifiers.
- **Visibility Specifier in PHP:** There are 3 types of visibility available in php for controlling your property or method.

– **Public:** Public method or property can be accessible from anywhere. i.e. from inside the class, outside the class and in child class also.

– **Private:** Method or property with private visibility can only be accessible inside the class. You cannot access private method or property from outside of your class.

– **Protected:** Method or variable with protected visibility can only be access in the derived class. Or in other word in child class. Protected will be used in the process of inheritance.

Implementation of Object Oriented Concepts:

Function	**Description**	**Example**

class classname [extends baseclass]	Creates a class	Class student
		{
		[var $propery [= value];…] [function functionname (arguments)
		{
		//code
		}
		….]}

$instance = new classname();	Create an object	<?php $instance1 = new myclass (); //This can also be done with a variable: $newname= ‘hello’; $instance2 = new $newname(); ?>

class classname
{
function methodname()
{
Statements;
}
}
	Add a Method	<?php
		class myclass
		{ function mymethod()
		{ print “ hello, myclass}}
		?>
		To invoke the method on the object $instance1, we need to invoke the operator “->" to access the newly created function mymethod
		
		<?php
		$instance1=new myclass();
		$instance1->mymethod();
		?>

public $publicMemeber = “Public member”;	Adding Property Public	Public :
		<?php class maths
		{
		public $num;
		public function multi()
		{
		return $this->num*2;
		}
		}
		$math=new maths;
		$math->num=2; echo $math->multi();
		?>
		Output : 4

protected $protectedmember = "Protected Member"; Private $privatemember= "Private Member	Protected Private	Protected:
		<?php class maths
		{
		protected $num;
		public function setnum($num)
		{
		$this->num=$num;
		}
		
		public function multi()
		{
		return $this->num*2;}}
		
		class add extends maths
		{
		public function addtwo()
		{
		$new_num=$this->num + 2; return ($new_num);
		}
		}
		$math=new add;
		$math->setnum(14); echo $math->addtwo();
		?>
		Output : 16

Overriding	When we give a function in the child class the same name as a function in the parent class, this concept is called function overriding. Any method or class that is declared as final can not be overridden or inherited by another class.	<?php class Hello {function getMessage() { return 'Hello W orld!';} } class Goodbye extends Hello {function getMessage(){ return 'Goodbye W orld!';}} $hello=&new Hello(); Echo $hello->getMessage().' '; $goodbye = &new Goodbye(); Echo $goodbye->getMessage(). ' ';?> Output: Hello World! Goodbye W orld!
void _construct ([mixed $args [, $....]])	**Constructor** is a function which is called right after a new object is created.	Method 1 <?php class student { public $name; public $marks; function __construct($nm,$mk) {

		$this->name=$nm;
		$this->marks=$mk;
		}
		}
		$obj1=new student('abc',90);
		echo "Name = $obj1->name
		";
		echo "Marks = $obj1->marks
		";
		?>
		
		Method 2
		<?php
		class student
		{
		var $name;
		var $marks;
		function student($nm,$mk)
		{
		$this->name=$nm;
		$this->marks=$mk;
		}
		}
		$obj1=new student("abc",90);
		echo "Name = $obj1->name
		";
		echo "Marks = $obj1->marks
		";
		?>

void _destruct (void)	**Destructor** is a function which is called right after you release an object.	<?php
		class Student
		{
		var $name; var $address; var $phone;
		
		//This is constructor
		function _construct()
		{
		this->name="abc"; this->address="pqr"; this->phone=1111;
		}
		
		function __destruct()
		{
		echo "Student Object Released";}
		
		function printstudentinfo()
		{
		Echo this->name . "\n"; echo this->address . "\n"; echo this->phone . "\n";
		}

		} $stud =new student(); $stud->printstudentinfo(); $stud=NULL; ?>

SET A

1. Write a Calculator class that can accept two values, then add them, subtract them, multiply them together, or divide them on request.

   For example:

   $calc = new Calculator(3, 4);

   echo $calc- >add(); // Displays "7"

   echo $calc- >multiply(); // Displays "12"

2. Create a class named DISTANCE with feet and inches as data members. The class has the following member functions: convert_feet_to_inch() , convert_inch_to_feet() . Display options using radio button and display conversion on next page.
3. Write a PHP program to create class circle having radius data member and two member functions find_circumfernce () and find_area() . Display area and Circumference depending on user's preference.

SET B:

1. Write a PHP program to create a class Employee that contains data members as Emp_Name, Dept_name , Basic_sal,DA, HRA,TA , IT,PF,PT , GROSS, DEDUCTION ,NET . It has member functions calculate_gross , calculate_deductions , Calculate_net_salary . Display pay slip of employee. Create and Initialize members Emp_Name, Dept_name , Basic_sal of Employee object by using parameterized constructor.

   2. Write a PHP program to create a class temperature which contains data members as Celsius and Fahrenheit . Create and Initialize all values of temperature object by using parameterized constructor . Convert Celsius to Fahrenheit and Convert Fahrenheit to Celsius using member functions. Display conversion on next page.

3. Write a PHP program to create a class Worker that has data members as Worker_Name, No_of_Days_worked, Pay_Rate. Create and initialize the object using default constructor, Parameterized constructor. Also write necessary member function to calculate and display the salary of worker.

SET C:

1. Write a PHP program to create a class article having articleid, name, articleqty, price. Write menu driven program to perform following functions :(Use array of objects)
   i. Display details of all articles purchased.
   ii. Display details of articles whose price exceeds 500
   iii. Display details of articles whose quantity exceeds 50

Signature of the instructor: ---------- Date:----------

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: well done [ ]

Assignment 2: Inheritance, Interface & Introspection

By Ms. Farzana Peer Mohammed

- **Inheritance:** When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
- **Polymorphism:** This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it makes take different number of arguments and can do different task.
- **Overloading:** a type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments. Similarly functions can also be overloaded with different implementation.
- **Data Abstraction:** Any representation of data in which the implementation details are hidden (abstracted).
- **Interface:** Interfaces allow you to create code which specifies which methods a class must implement, without having to define how these methods are handled. Interfaces are defined in the same way as a class, but with the *interface* keyword replacing the *class* keyword and without any of the methods having their contents defined. All methods declared in an interface must be public; this is the nature of an interface. Note that it is possible to declare a constructor in an interface.
- **Introspection:** Introspection is the ability of a program to examine an object's characteristics, such as its name, parent class (if any), properties, and methods. With introspection, you can write code that operates on any class or object. You don't need to know which methods or properties are defined when you write your code; instead, you can discover that information at runtime, which makes it possible for you to write generic debuggers, serializes, profilers, etc.

Function	**Description**	**Example**

class extendedClass
extends classname
	Inheritance
	It is the ability of PHP to extend classes that inherit the characteristics of the parent class.
	
	It is not possible to extend multiple classes ; a class can only inherit from one base class.
		<?php
		class myclass
		{
		//property declaration
		public $var='a default value';
		
		//method declaration
		public function desplayVar()
		{
		echo $this->var;
		}
		}
		class extendedClass extends myclass

		{
		//redefine the parent method
		function displayVar()
		{
		echo “Extending Class”;
		parent::displayVar();
		}
		}
		$extend =new extendedClass();
		$extend->displayVar();
		?>
		Output :
		Extending class
		a default value

Overriding	When we give a function in the child class the same name as a function in the parent class, this concept is called function overriding. Any method or class that is declared as final can not be overridden or inherited by another class.	<?php class Hello {function getMessage() { return ‘Hello W orld!’;} } class Goodbye extends Hello {function getMessage(){ return ‘Goodbye W orld!’;}} $hello=&new Hello(); Echo $hello->getMessage().’ ’; $goodbye = &new Goodbye(); Echo $goodbye->getMessage(). ’ ’;?> Output: Hello World! Goodbye W orld!
class_exist()	**Introspection** We can use this function to determine whether a class exists.	$class = class_exists(classname);
get_declared_classes()	This function returns array of defined classes and checks if the class name is in returned array.	$classes = get_declared_classes();
get_class_methods()	We can use this function to get the methods and properties of class	$methods=get_class_methods(classna me);
get_class_vars()	This function returns only properties that have default values.	$properties=get_class_vars(classnam e);

get_parent_class()	This function is used to find the class's parent class.	$superclass=get_parent_class(classname);
is_object()	Is_object function is used to make sure that it is object.	$obj= is_obj(var);
get_class()	get_class() function is used to get the class to which an object belongs and to get class name	$classname= get_class(object);
method_exists()	This function is used to check if method on an object exists .	$method_exists=method_exists(object ,method);
get_object_vars()	This function returns an array of properties set in an object	$array=get_object_vars(object);

Interfaces	An **interface** is declared similar to a class but only include function prototypes (without implementation) and constants. When a class uses an interface the class must define all the methods / function of the interface otherwise the PHP engine will give you an error.
	
	The interface's function /methods cannot have the details filled in. that is left to the class that uses the interface.
		Example of an interface class duck
		{ function quack()
		{
		echo "quack,quack,qk, qk…";
		}
		}
		Interface birds
		{
		function breath(); function eat();
		}
		Class duck implements birds
		{ function quack()
		{
		echo "quack,quack,qk, qk…";
		}
		}
		function breath()
		{
		echo "duck is breathing";
		}
		function eat()
		{

		echo " duck is eating"; }

Encapsulation	**Encapsulation** is an ability to hide details of implementation.	<?php class A
		{
		function check()
		{
		if(isset ($this))
		{
		echo "$this is defined ("; echo get_class($this); echo ")\n";
		}
		else
		{
		echo "this is not defined";
		}
		}
		}
		class B
		{
		function bcheck()
		{
		A::check();
		}
		}
		
		$a=new A();
		$a->check();
		A::check();
		$b=new B();
		$b->bcheck();
		B::bcheck();
		?>
		
		Output:
		
		$this is defined(a)
		$this is not defined
		$this is defined(b)
		$this is not defined

LAB Assignments :

SET A

1) Write class declarations and member function definitions for following :
   employee(code, name, designation). Design derived classes as emp_account(account_no, joining_date) from employee and emp_sal(basic_pay, earnings, deduction) from emp_account. Write a PHP Script to create 5 objects (pass details using parameterized constructor) and Display details of Employees who having Maximum and Minimum Salary.
2) Define an interface which has methods area( ), volume( ). Define constant PI. Create a class

cylinder which implements this interface and calculate area and volume.

3) Write a PHP script to demonstrate the introspection for examining class .(use function et_declared_classes() ,get_class_methods() and get_class_vars()).
4) Write PHP script to demonstrate the concept of introspection for examining object.

SET B

1) Create class rectangle and derive a class square from class Rectangle. Create another class circle. Create an interface with only one method called area(). Implement this interface in all the classes. Include appropriate data members and constructors in all classes. W rite a program to accept details of a square, circle and rectangle and display the area.
2) Define a class Employee having private members – id, name, department, salary. Define parameterized constructors. Create a subclass called "Manager" with private member bonus. Create 6 objects of the Manager class and display the details of the manager having the maximum total salary (salary + bonus).
3) Derive a class square from class Rectangle. Create one more class circle. Create an interface with only one method called area (). Implement this interface in all the classes. Include appropriate data members and constructors in all classes. Write a program to accept details of a square, circle and rectangle and display the area.
4) Write a PHP Script to create a super class Vehicle having members Company and price. Derive 2 different classes LightMotorVehicle (members – mileage) and HeavyMotorVehicle (members – capacity-in-tons). Define 5 Object of each subclass and display details in table format .

SET C:

1) Create an interface for the classes that handle cars, having methods to setModel() and getModel() methods. Create another interface Vehicle having setHasWheels($bool) and getHasWheels() methods which is boolean. Create class miniCar with two properties model and hasWheels property and implement the two interfaces.

Signature of the instructor:---------- Date:----------

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: well done [ ]

Assignment 3 – PHP Global Variables and Form Methods

By- Mrs. Veena K. Gandhi

Server configuration and request information—including form parameters and cookies—are accessible in three different ways from your PHP scripts. Collectively, this information is referred to as EGPCS (environment, GET, POST,cookies, and server).

If the register_globals option in php.ini is enabled, PHP creates a separate global variable for every form parameter, every piece of request information, and every server configuration value

Regardless of the setting of register_globals, PHP creates six global arrays that contain the EGPCS information. The global arrays are:

- **$HTTP_COOKIE_VARS or $_COOKIE :**Contains any cookie values passed as part of the request, where the keys of the array are the names of the cookies
- **$HTTP_GET_VARS or $_GET** : Contains any parameters that are part of a GET request, where the keys of the array are the names of the form parameters
- **$HTTP_POST_VARS or $_POST :** Contains any parameters that are part of a POST request, where the keys of the array are the names of the form parameters
- **$HTTP_POST_FILES or $_FILES :** Contains information about any uploaded files
- **$HTTP_SERVER_VARS** or **$_SERVER :** Contains useful information about the web server, as described in the next section
- **$HTTP_ENV_VARS or $_ENV :**Contains the values of any environment variables, where the keys of the array are the names of the environment variables

The $_REQUEST array is also created by PHP if the register_globals option is on. PHP also creates a variable called $PHP_SELF, which holds the name of the current script, relative to the document root).

Server information :

$_SERVER is a PHP super global array which holds information about the items like Server information, Header information, Details on PHP page request, File name or path information, Remote user information, HTTP Authentication Details.

Element/Code	Description
$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string

$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_HOST']	Returns the Host header from the current request
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web serve
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80
$_SERVER['SCRIPT_NAME']	Returns the path of the current script
$_SERVER['SCRIPT_URI']	Returns the URI of the current page

Example: to display server information like name , script name , user agent etc.

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

Output :

```
/php/demo_global_server.php
www.w3schools.com
www.w3schools.com
https://www.w3schools.com/php/showphp.asp?filename=demo_global_server
Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.112
Safari/537.36
/php/demo_global_server.php
```

PHP Form Methods:

Scripts will interact with their clients using one of the two HTTP methods. The methods are GET and POST

When a form is submitted using the GET method; its values are encoded directly in the query string portion of the URL. This method should not be used when sending passwords or other sensitive information. However, because the variables are displayed in the URL, it is possible to bookmark the page

The get method is not suitable for large variable values; the value cannot exceed 100 characters
The $_GET array is used to collect values from a form sent with method="GET"

When a form is submitted using the POST method, its values will not be displayed the query string portion of the URL and has no limits on the amount of information to send . However, there is

an 8 Mb max size for the POST method, by default (can be changed by setting the post_max_size in the php.ini file). The $_POST array is used to collect values from a form sent with method="POST"

Example using GET method

Form1.html

```
<html>
<body>
/* form submitted using 'get' method, action specifies  next page which is to be loaded when button is clicked*/
<form action="welcome.php" method="GET">
// textbox is to take user input
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
// Submit button is to submit the value
<input type="submit" />
</form>
</body>
</html>
```

welcome.php

```
<html>
<body>
// $_GET to receive the data sent from Form1.html
Welcome <?php echo $_GET["fname"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
</body>
</html>
```

Example using POST method

form1.html

```
<html>
<body>
/* form submitted using 'post' method, action specifies next page which is to be loaded when button is clicked */
<form action="welcome1.php" method="POST">
// textbox is to take user input
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
// Submit button is to submit the value to next page
<input type="submit" />
</form>
</body>
</html>
```

welcome1.php

```
<html>
<body>
// $_GET to receive the data sent from form1.html
Welcome <?php echo $_POST["fname"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old!
</body>
</html>
```

Lab Assignments:

SET A

1) Write PHP script to display GATEWAY INTERFACE, Server Address, Server Name, server software, server protocol.
2) Write PHP program to accept user details such as user-id, name, Address, email, and mobile no. Display same information on next page.
3) Write a PHP program that takes name and age from an HTML page. If the age is less than 18, it should send a page with "hello <name>,and u are not authorized to visit this site" otherwise it should send "welcome <name> to this site" message where name should be replaced with the entered name otherwise it should send welcome<name> to the site message.

SET B

1) Write a PHP Program to display all Server information in table format.
2) Write PHP program to accept client no, client name, driving license number, age etc. If age of client is less than 18 display message "You are not authorized to ride" , otherwise display client information on next page.
3) Write a PHP script to accept user preference like background color, text font, login message. On the next page display login message using the preferences.

SET C

1) A college has given roll number to each student, The roll number is six digit number where first two digits are faculty(B.Sc., BCA, BA) third digit is year (Ist(1), IInd(2) and IIIrd(3)) and last three digit are actual number. Write PHP script to accept a number and print faculty, year and roll number of student.

2) Write a PHP script that enables the user to construct his own web form with the capabilities to choose control & then allow the user to submit the form & display the selected controls on next page**.**

Signature of the instructor:---------- Date:----------

Assignment Evaluation

0: Not Done [ ] 1: Incomplete [ ] 2: Late Complete [ ]

3: Needs Improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Assignment 4 – Self Processing Pages, Sticky forms with Multi-value Parameters

By- Mrs. Veena K. Gandhi

Self Processing Page:

- Self processing page means one PHP page can be used to both generate a form and process it.You can use PHP_SELF variable for generating self processing page. PHP_SELF is a variable that returns the current script being executed. This variable returns the name and path of the current file (from the root folder). You can use this variable in the action field of the FORM.
- <form name="form1" method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>" >

Example A self-processing page

```
<html>
<head><title>Temperature Conversion</title></head>
<body>

<?php
 if ($_SERVER['REQUEST_METHOD'] == 'GET') {
?>

<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="POST">
Fahrenheit temperature:
<input type="text" name="fahrenheit" /> <br />
<input type="submit" name="Convert to Celsius!" />
</form>

<?php
 }
 elseif ($_SERVER['REQUEST_METHOD'] == 'POST')
 {
   $fahr = $_POST['fahrenheit'];
   $celsius = ($fahr - 32) * 5/9;
   printf("%.2fF is %.2fC", $fahr, $celsius);
 } else
{
   die("This script only works with GET and POST requests.");
 }
?>

</body>
</html>
```

Sticky Forms:

Sticky form remembers the values you entered in the input fields. Good example of sticky form is Google search box. Sticky form helps user to type the same form again supplying the values in inputs. Sticky form is form in which the results of a query are accompanied by a search form whose default values are those of the previous query. To make sticky form , You just include the attribute value for text fields, and selected/checked for other elements:

Example :

```
<html>
<body>
<form action="<?php $_SERVER['PHP_SELF']; ?>" method="POST">
<b>Your Name : </b><input type="text" name="name" value="<?php if(isset($_POST['name'])) echo
$_POST['name'];?>">

<p><input type="submit" name="submit" value="Submit" /></p>

</form>
<?php
   echo "Your Name is =". $_POST['name']."<br>";
?>
</body>
</html>
```

Multi – Valued Parameters:

HTML selection lists, created with the select tag, can allow multiple selections. To ensure that PHP recognizes the multiple values that the browser passes to a form-processing script, you need to make the name of the field in the HTML form end with [].When PHP engine sees a submitted form field name with square brackets at the end, it creates a nested array of values within the $_GET or $_POST and $_REQUEST superglobal array, rather than a single value.

For example:

```
<select name="languages[]">
  <input name="c">C</input>
  <input name="c++">C++</input>
  <input name="php">PHP</input>
  <input name="perl">Perl</input>
</select>
```

Now, when the user submits the form , $_GET['languages'] contains an array instead of a simple string. This array contains the values that were selected by the user.

Example

```
<html>
<head><title>LANGAUGES</title></head>
<body>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
Select your Language :<br>
<select name="languages[]"  multiple>
<option value="c">  C </option>
<option value ="c++"> C++ </option>
<option value ="php"> PHP </option>
<option value ="perl"> Perl </option>
</select>
<br>
<input type="submit" name="s" value="My Languages!" />
</form>
<?php
```

```
if (array_key_exists('s', $_GET))
 {
$lang = join ("  ,", $_GET['languages']);
echo "You know $lang languages.";
}
?>
</body>
</html>
```

Sticky Multi – Valued Parameters:

You can make multiple selection form elements sticky. You'll need to check to see whether each possible value in the form was one of the submitted value.

For example :

RED: <input type="checkbox" name="attributes[]" value="red" <?= if (is_array($_GET['attributes']) and in_array('red', $_GET['attributes'])) { "checked"; } ?> >

Consider following example to implement sticky multi-value parameters

```
<html>
<head><title>LANGAUGES</title></head>
<body>

<?php
$c1 = $_GET['c1'];
?>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
Qualification : <br>
<input type="checkbox" name="c1[]" value="ssc" <?php if(in_array('ssc', $_GET['c1'])) {echo
"checked"; }?>> SSC <br>

<input type="checkbox" name="c1[]" value="hsc"
<?php if(is_array($_GET['c1']) and in_array('hsc', $_GET['c1'])) { echo "checked"; }?>>  HSC <br>

<input type="checkbox" name="c1[]" value="bca"
<?php if(is_array($_GET['c1']) and in_array('bca', $_GET['c1'])) { echo "checked";}?>>  BCA <br>

<input type="checkbox" name="c1[]" value="mca"
<?php if(is_array($_GET['c1']) and in_array('mca', $_GET['c1'])) {echo "checked";}?>> MCA <br>

<input type="submit" name="s" value="My Qualification" />
</form>
<?php
if (array_key_exists('s', $_GET))
 {
$a = join ("  ,", $_GET['c1']);
echo "You Qualification : $a";
}
?>
</body>
</html>
```

Lab Assignments:

SET A:

1) Write a PHP program to create a simple distance calculator that can accept distance in meters from user. Convert it into centimeter or kilometer according to user preference(use radio buttons and Self Processing form)
2) Write a PHP script for the following: Design a form to accept a number from the user. Perform the operations and show the results.
   - Check whether number is palindrome or not.
   - Reverse the number using recursions.

   (Use the concept of self processing page.)
3) Write a PHP script to accept an Indian currency and then convert it in dollar or pounds (radio buttons) according to user's preference. (use concept of self processing form)
4) Write a PHP script to accept a weight in kg and Height in meter of person and then calculate BMI. BMI= weight in Kg/ (Height in meter)2

If BMI is in following categories , then display message accordingly(use concept of self processing form)
Underweight = <18.5 , Normal weight = 18.5–24.9 ,Overweight = 25–29.9 ,Obesity = BMI of 30 or greater.

5) Write a PHP program for the following create a calculator that can store two values and perform operations like add, subtract, multiplication and divide (using Self Processing form)

SET B:

1) Write a PHP program to accept two strings from user and check whether entered strings are matching or not. (Use sticky form concept)
2) Write PHP program to create student registration form and display student information. (Use sticky form concept).
3) Write PHP program to calculate simple interest. (Use sticky form concept)
4) Write PHP program accept name, select your cities you would like to visit (Use multi-valued parameter), and display selected information on page.
5) Write PHP program to select list of subjects from list box and display selected subject on information. (Use sticky multi-valued parameter

SET C:

1) Write PHP program to accept client name, marital status, languages known , property details (House, Mobile , TV , Car). Display selected information same page. (Use multi-value parameter)
2) Write PHP program to accept name of student , Gender(male ,female) using radio buttons ,Qualification(SSC,HSC,BCA,MCA) using check boxes . Display information of student. (Use sticky multi-valued parameter)
3) Write a PHP script to accept a string and then display each word of string in reverse order. (use concept of self processing form)

Signature of the instructor:---------- Date:----------

Assignment Evaluation

0: Not Done [ ] 1: Incomplete [ ] 2: Late Complete [ ]

3: Needs Improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Assignment 5 – File uploads and Form Validation

By- Mrs. Veena K. Gandhi

A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script.

In your "php.ini" file, search for the file_uploads directive, and set it to On:

file_uploads = On

There is one global PHP variable called **$_FILES**. This variable is an associate double dimension array and keeps all the information related to uploaded file.

The keys are:

name

The name of the file, as supplied by the browser. It's difficult to make meaningful use of this, as the client machine may have different filename conventions than the web server (e.g., if the client is a Windows machine that tells you the file is*D:\PHOTOS\ME.JPG*, while the web server runs Unix, to which that path is meaningless).

type

The MIME type of the uploaded file, as guessed at by the client.

size

The size of the uploaded file (in bytes). If the user attempted to upload a file that was too large, the size is reported as 0.

tmp_name

The name of the temporary file on the server that holds the uploaded file. If the user attempted to

upload a file that was too large, the name is reported as "none".

For example :

- **$_FILES['filename']['tmp_name']** − the uploaded file in the temporary directory on the web server.
- **$_FILES['filename']['name']** − the actual name of the uploaded file.
- **$_FILES['filename']['size']** − the size in bytes of the uploaded file.
- **$_FILES['filename']['type']** − the MIME type of the uploaded file.
- **$_FILES['filename']['error']** − the error code associated with this file upload.
-

The correct way to test whether a file was successfully uploaded is to use the function is_uploaded_file(), as follows:

```
if (is_uploaded_file($_FILES['toProcess']['tmp_name'])
{
        // successfully uploaded
}
```

To move a file, use the move_uploaded_file() function:

move_uploaded_file($_FILES['toProcess']['tmp_name'], "path/to/put/file/$file);

The call to move_uploaded_file() automatically checks whether it was an uploaded file. When a script finishes, any files uploaded to that script are deleted from the temporary directory.

Consider sample program to upload file

```
<<html>
<body>
<form action="upload.php" method="post" enctype="multipart/form-data">
Select File to upload:
<input type="file" name="uploadedfile" id="fileToUpload">
<input type="submit" value="Upload File/Image" name="submit">
</form>
</body>
</html>

<?php
$target_path = "D:/uploads/";

$target_path = $target_path . basename( $_FILES['uploadedfile']['name']);

if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path))
 {
    echo "The file ".  basename( $_FILES['uploadedfile']['name']). " has been uploaded";
   }
else{
    echo "There was an error uploading the file, please try again!";
}
?>
```

Form Validation:

When you allow users to input data, you typically need to validate that data before using it or storing it for later use. There are several strategies available for validating data like Form Fields should not be empty. Check type of data entered by user or length of data entered by user. (e.g check age of person is integer or cannot be negative etc.) Check specific conditions for form fields(e.g. email validation).

PHP provides empty() function to check a variable is empty. We can use this function to check if all the text fields are empty or not. isset() function can be used to check the gender radio button is checked or not. We can validate email format by using PHP filter_var() function.

- **empty(var_name) :** The empty() function is used to check whether a variable is empty or not.
- **isset(var_name) :** this function determines if a variable is set and is not NULL
- **filter_var(*var, filtername, options*)** : The filter_var() function filters a variable with the specified filter.

Consider the following script to validate email.

```
<?php
$email = "john.doe@example.com";
```

```
if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
  echo("$email is a valid email address");
} else {
  echo("$email is not a valid email address");
}
?>
```

Example : Dsign form which accepts name and gender . If fields are blank display error message “All fields are required” otherwise display information

```
<html>
<head>
<style>
span.error {color: red;}
</style>
</head>
<body>

<?php

$nameErr = $genderErr = "";
$name = $gender = "";

if ($_SERVER["REQUEST_METHOD"] == "POST")
 {
  if (empty($_POST["name"]))
    $nameErr = "Name is required";
   else
    $name = $_POST["name"];

  if (!isset($_POST["gender"]))
    $genderErr = "Gender is required";
   else
    $gender =$_POST["gender"];
  }
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* Required field.</span></p>
<form method="POST" action="<?php echo $_SERVER['PHP_SELF']?>">
  Name: <input type="text" name="name"  value="<?php  if ($name)  echo "$name";?>" >
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>

  <input type="radio" name="gender" value="female" <?php  if($gender=="female") { echo "checked";
}?>>Female
  <input type="radio" name="gender" value="male"  <?php  if($gender=="male") { echo "checked"; }?>>Male
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
if($name && $gender)
{
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $gender;
```

```
}
?>
</body>
</html>
```

Lab Assignments:

SET A:

1. Write a PHP Program to Upload the file and display its information like name, size, type , etc.

2. Write a PHP program to accept student information like name, address, class and Upload student photo and display same on form.

SET B:

1. **Write** a PHP program to accept Name , address , Pincode ,Gender information. If any field is blank , it display error message "all fields are required" . If pincode is more than 6 digits , it should give error.

2. Write a PHP program to accept empno, name, pan card information, email . If any field is blank , form should display error message "all fields are required". Pan card number should be 10 digits and First 5 characters should be letter , next 4 characters should be digit and last character should be letter.

3. Write a PHP script to create a form that accept theusers full name and their email addresses. Use case conversion function to capitalize the first letter of each name, user submits and print result back to browser. Check that the user's email address contains the @ symbol.

SET C:

1) Write a PHP script for creating a self-processing page for a form. The form should allow the user to enter the following attributes: Username, user city preference(pune/Mumbai/Chennai/kolkata),user birth date, occupation, sex. If any of the values is not entered by the user, the page is presented again with a message specifying the attributes that are empty. Any form attributes that the user already entered, are set to the values the user entered. The text of submit button changes from "create" to "continue", when the user is correcting the form. Display the details entered by the user on the next form

Signature of the instructor:---------- Date:----------

Assignment Evaluation

0: Not Done [ ] 1: Incomplete [ ] 2: Late Complete [ ]

3: Needs Improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Assignment 6 - Cookies and Sessions

By- Mrs. Veena K. Gandhi.

Cookies :

- A cookie is basically a string that contains several fields. A server can send one or more cookies to a browser in the headers of a response. Some of the cookie's fields indicate the pages for which the browser should send the cookie as part of the request.
- The setcookie() function is used to send a cookie to the browser:
- setcookie(name[,value[,expire[,path[,domain[,secure]]]]]); where parameters are as
- **name :** A unique name for a particular cookie. You can have multiple cookies with different names and attributes. The name must not contain whitespace or semicolons.
- **value :** The arbitrary string value attached to this cookie. The original Netscape specification limited the total size of a cookie (including name, expiration date, and other information) to 4 KB, so while there's no specific limit on the size of a cookie value, it probably can't be much larger than 3.5 KB.
- **expire** : The expiration date for this cookie. If no expiration date is specified, the browser saves the cookie in memory and not on disk. When the browser exits, the cookie disappears. The expiration date is specified as the number of seconds.
- **path :** The browser will return the cookie only for URLs below this path. The default is the directory in which the current page resides.
- **domain** : The browser will return the cookie only for URLs within this domain. The default is the server hostname.
- **secure** :The browser will transmit the cookie only over https connections. The default is false, meaning that it's okay to send the cookie over insecure connections

  This function creates the cookie string from the given arguments and creates a Cookie header with that string as its value. Because cookies are sent as headers in the response, setcookie() must be called before any of the body of the document is sent.
  When a browser sends a cookie back to the server, you can access that cookie through the $_COOKIE array. The key is the cookie name, and the value is the cookie's value field.

Consider a sample program to keep track of number of times the web page has been accessed

```
<?php
            if(isset($_COOKIE['accesses']))
        $cnt = $_COOKIE['accesses'];
      else
          $cnt = 0 ;

    setcookie('accesses', ++$cnt);

      echo "You have visited this page $cnt times ";
 ?>
```

Sessions :

Session allow us to easily create multi page forms, save user authentication information from page to page, and store persistent user preferences on a site. A session can be defined as a series pf related interactions between a single client and the Web server. The session may consist of multiple requests to the same script or a variety of different resources on the same web site.

To enable sessions for a page, call session_start()before any of the document has been generated:

```
<?php  session_start() ?>
<html>
...
</html>
```

This assigns a new session ID if it has to, possibly creating a cookie to be sent to the browser, and loads any persistent variables from the store. into the associative array $HTTP_SESSION_VARS. The keys are the variables' names (e.g., $HTTP_SESSION_VARS['hits'])

Functions :

- bool **session_start** ([array $options = []]) :to enable session for a page. This function assigns a new session ID to the new session.
- bool **session_register** (mixed $name [, mixed $...]) : to register a variable with the session by passing the name of the variable. When a session is started, you can store any number of variables in the $_SESSION superglobal array and then access them on any session enabled page.
- string **session_id** ([string $id]) : **session_id()** is used to get or set the session id for the current session.
- bool **session_destroy** (void) : **session_destroy()** destroys all of the data associated with the current session. It does not unset any of the global variables associated with the session, or unset the session cookie. To use the session variables again, session_start() has to be called.
- **bool session_unregister(string $name)** : session_unregister() unregisters the global variable named name from the current session.

Consider example which prints visitor count of web page

```
<?php
session_start(); //start the PHP_session function
if(isset($_SESSION['page_count']))
   $_SESSION['page_count'] += 1;
else
    $_SESSION['page_count'] = 1;
 echo 'You are visitor number ' . $_SESSION['page_count'];
?>
```

Lab Assignments:

SET A:

1. A web application that lists all cookies stored in the browser on clicking "list cookies" button, add cookies if necessary.
2. Write a PHP program to store current date-time in a COOKIE and display the 'Last visited on' date-time on the web page upon reopening of the same page.
3. Write a script to keep track of number of times the web page has been accessed using session.
4. Create a login form with a username and password. Once the user logs in, the second form should be displayed to accept user details (name, city, phoneno). If the user doesn't enter information within a specified time limit, expire his session and give a warning otherwise Display Details using sessions.
5. Write PHP program to store Customer information like customer no, name, address, mobile no. On second page, accept product code, product name, Qty, Rate. Display Bill on third page including customer and product details.

SET B:

1. Write a PHP script to accept username and password. If in the first three chances, username and password entered is correct, then display second form, otherwise display error message.
2. Create a form to accept student information (name, class, address). Once the student information is accepted, accept marks in next form (Java, PHP, SE, OS, Pract1, and Pract2). Display the mark sheet for the student in the next form containing name, class, marks of the subject, total and percentage using cookies.
3. Change the preferences of your web page like font style, font size, font color, background color using cookie. Display selected settings on next web page and actual implementation (with new settings) on third web page.
4. Create a form to accept employee details like name, address and mobile number. Once the employee information is accepted, then accept LIC information like policy_no, name, premium. Display employee details and LIC details on next form.
5. Write a PHP script to accept Employee details (Eno, Ename, Add.) on first page. On second page accept earning (Basic, DA, HRA). On third page Print Employee Information (Eno, Ename, Add, Basic, DA, HRA, Gross)

SET C :

1. Write a program to create a shopping mall. User must be allowed to do purchase from three pages. Each page should have a page total. The fourth page should display a bill, which consists of a page total of whatever the purchase has been done and print the total. (Use $_SESSION).

Signature of the instructor:---------- Date:----------

Assignment Evaluation

0: Not Done [ ] 1: Incomplete [ ] 2: Late Complete [ ]

3: Needs Improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Assignment No. 7: Database(POSTGRESQL)

-by Mrs.Summaiya Tamboli

PostgreSQL supports a wide variety of built-in data types and it also provides an option to the users to add new data types to PostgreSQL, using the CREATE TYPE command. Table lists the data types officially supported by PostgreSQL. Most data types supported by PostgreSQL are directly derived from SQL standards. The following table contains PostgreSQL supported data types for your ready reference

Category	Data type	Description
Boolean	boolean, bool	A single true or false value.
Binary types	bit(n)	An n-length bit string (exactly n) binary bits)
	bit varying(n), varbit(n)	A variable n-length bit string (upto n) binary nbits)
Character Types	character(n)	A fixed n-length character string
	char(n)	A fixed n-length character string
	character varying(n)	
	varchar (n)	
	text	A variable length character string of unlimited length
Numeric types	smallint, int2	A signed 2-byte integer
	integer, int, int4	A signed, fixed precision 4-byte number
	bigint, int8	A signed 8-byte integer, up to 18 digits in length
	real, float4	A 4-byte floating point number
	float8, float	An 8-byte floating point number
	numeric(p,s)	An exact numeric type with arbituary precision p, and scale s.
Currency	money	A fixed precision, U.S style currency
	serial	An auto-incrementing 4-byte integer
Date and time types	date	The calendar date(day, month and year)
	time	The time of day
	time with time zone	the time of day, including time zone information
	timestamp(includes time	
	Interval)	An arbitrarily specified length

Functions used for postgreSQL database manipulation

Function name	Purpose	Example
resource **pg_connect** (string $connection_string [, int $connect_type]);	Open a PostgreSQL connection	$conn = pg_connect("host", "port", "options", "tty", "dbname");
resource **pg_pconnect** (string $connection_string [, int $connect_type]);	Open a persistent PostgreSQL connection	$conn_string = "host=sheep por t=5432 dbname=test user=lamb password=bar";
resource **pg_prepare** ([resource $connection],string $stmtname , string $query)	Submits a request to create a prepared statement with the given parameters, and waits for completion.	$result = pg_prepare($dbconn, "my_query", 'SELECT * FROM shops W HERE name = $1');
resource **pg_execute** ([resource $connection], string $stmtname , array $params)	Sends a request to execute a prepared statement with given parameters, and waits for the result.	$result = pg_execute($dbconn, "my_query", array("Joe's W idge ts"));

resource **pg_query** ([resource $connection], string $query)	Execute a query	$result = pg_query($conn, "SEL ECT author, email FROM autho rs");
		if (!$result) {
		echo "An error occurred.\n";
		exit;
		}

array **pg_fetch_assoc**(resourc e $result [, int$row])	Fetch a row as an associative array	while ($row = pg_fetch_assoc($ result)) echo $row['id'];
bool **pg_close** ([resource $connection])	Closes a PostgreSQL connection	pg_close($dbconn);

Example to create php Postgrsql Connectivity and display records

```
<?php
$conn=
pg_connect("dbname=publisher");
if (!$conn) {
  echo "An error
  occurred.\n"; exit;
}

$result = pg_query($conn, "SELECT id, author, email FROM
authors"); if (!$result) {
  echo "An error
  occurred.\n"; exit;
}
```

```
while ($row =
  pg_fetch_assoc($result)) { echo
  $row['id'];
  echo
  $row['author'
  ]; echo
  $row['email']
  ;
}
?>
```

Lab Assignments :

SET A:

Q 1) Consider the following entities and their relationships
Emp (emp_no,emp_name,address,phone,salary)
Dept (dept_no,dept_name,location)
Emp-Dept are related with one-many relationship Create a RDB in 3NF for the above and solve following
Using above database write a PHP script which will print a salary statement in the format given below, for a given department. (Accept department name from the user).

Deparment Name :

Maximum Salary	Minimum Salary	Sum Salary

Q2) Consider the following entities and their relationships
Doctor (doc_no, doc_name, address, city, area)
Hospital (hosp_no, hosp_name, hosp_city)

Doctor and Hospital are related with many-many relationship. Create a RDB in 3 NF for the above and Using above database, write a PHP script which accepts hospital name and print information about doctors visiting / working in that hospital in tabular format.

Q3)Consider the following entities and their relationships
Item (item_no, Item_name, qty)
Supplier(supp_no, supp_name, address, city, phoneno)

Item and Supplier are related with many–many relationship with rate and discount as the attribute of relationships.
Create a RDB in 3 NF for the above and using above database, write a PHP script, Create a login form, after validating user name and password give option to 1) Add item 2) Search for an item by name

Q 4) Consider the following entities and their relationships
Movie (movie_no, movie_name, release_year)
Actor (actor_no, name)
Relationship between movie and actor is many – many with attribute rate in Rs. Create a RDB in 3 NF for the above and using above database, write PHP scripts for the following:(Hint: Create HTML form having three radio buttons)
Accept actor name and display the names of the movies in which he has acted.
Insert new movie information.
Update the release year of a movie. (Accept the movie name from user)

Q5) Write a PHP program to accept username and password from the user. Validate it against the login table in the database. If there is a mismatch between username and password, then, display the error message as ―invalid user name and password; else display the message as ―Login successful on the browser.

SET B:

Q1) Considerer the following entities and their relationships
project(pno integer, p_name char(30), ptype char(20),duration integer)
employee (eno integer, e_name char (20), qualification char (15), joindate date)

The relationship between project - employee: M-M, with descriptive attributes as start_date (date), no_of_hours_worked (integer).
Using above database write a script in PHP to accept a project name from user and display information of employees working on the project.

Q 2) Consider the following entities and their relationships
student (sno integer, s_name char(30), s_class char(10), s_addr char(50))
teacher (tno integer, t_name char (20), qualification char (15),experience integer)
The relationship between student-teacher: m-m with descriptive attribute subject Using above database write a script in PHP to accept a teacher name from user and display the names of students along with subjects to whom teacher is teaching

Q3)Consider the following entities and their relationships.
Employee (Emp_no, Emp_name, Basic_sal)
Dept (Dept_no, Dept_name)
Employee and Dept are related with many–one relationship.
Create a RDB in 3 NF for the above .
Using above database write a PHP script for the following to display the employees earning salary greater than user define salary. (accept salary from user).

Q 4) Considerer the following entities and their relationships

Student (Stud_id,name,class)
Competition (c_no,c_name,type)
Relationship between student and competition is many-many with attribute rank and year. Create a RDB in 3NF for the above and solve the following.
Using above database write a script in PHP to accept a competition name from user and display information of student who has secured 1st rank in that competition.
.
Q:5) Consider the following relational database:
Project (P_Group_No, Project_Title)
Student (Seat no, Name, Class, P_Group_No)
Write a PHP script to accept project title and display list of students those who are working in a particular project.

Set C

Q:1) Make Simple application using PHP program to implement Insert, Update,delete, and Search operations on Employee table with attributes(empno, empname, date_of_join, address, salary). (Use Radio Buttons)

.Q2) Consider the following entities and their relationships
BillMaster(billno, custname, billdate)
BillDetails(itemname, qty, rate, discount)

BillMaster and BillDetails are related with one-to-many relationship.
Create a RDB in 3 NF for the above and solve following
Write PHP script to print the bill in following format Accept the Bill number from user.

BillNo : **BillDate :**
Customer Name :

Sr. No.	Particular	Quantity	Rate	Discount	Total

Signature of the instructor:_____ Date:_____

Assignment Evaluation

0:Not Done [ ] **2:Late Complete [ ]** **4:Complete [ ]**

1:Incomplete[ ] **3:Needs Improvement [ ]** **5:Well Done[ ]**

Assignment 8: XML

By Mr. Salauddin Sajjan

Introduction to XML :

XML stands for EXtensible Markup Language. It is a text-based markup language derived from Standard Generalized Markup Language (SGML). XML was designed to store and transport data. XML was designed to be both human- and machine-readable. XML is a markup language much like HTML. XML was designed to describe data. XML tags are not predefined . You must define your own tags.XML is self describing.

XML document are well – formed and valid. A well - formed XML document follows the basic XML syntax rules.A valid document also follows the rules imposed by a DTD or an XSD.

A simple document is shown in the following example −

```
<?xml version = "1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```

The following image depicts the parts of XML document.

Document Prolog Section :

Document Prolog comes at the top of the document, before the root element. This section contains −

- XML declaration
- Document type declaration

Document Elements Section:

Document Elements are the building blocks of XML. These divide the document into a hierarchy of sections, each serving a specific purpose.

XML declaration :

It contains details that prepare an XML processor to parse the XML document. It is optional, but when used, it must appear in the first line of the XML document.

```
<?xml version="version_number" encoding="encoding_declaration"
standalone="standalone_status" ?>
```

An XML declaration should abide with the following rules:

- The XML declaration is case sensitive and must begin with "<?xml>" where "xml" is written in lower-case. If the XML declaration is included, it must contain version number attribute.
- The Parameter names and values are case-sensitive.The names are always in lower case.
- The order of placing the parameters is important. The correct order is:*version, encoding and standalone.* Either single or double quotes may be used.
- The XML declaration has no closing tag i.e. </?xml>

Example of XML declaration :

- <?xml >
- <?xml version="1.0">
- <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <?xml version='1.0' encoding='iso-8859-1' standalone='no' ?>

DTD :Document Type Declaration :

- The XML Document Type Declaration, commonly known as DTD, is a way to describe XML language precisely.
- DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language.
- An XML DTD can be either specified inside the document, or it can be kept in a separate document and then liked separately.
- Basic syntax of a DTD is as follows:

```
<!DOCTYPE element DTD identifier
 [
   declaration1
   declaration2
  ........
]>
```

XML Tags :

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>. Opening and closing tags must be written with the same case.

For example,

<Message>This is incorrect</message>

<message>This is correct</message>

XML Elements :

- An XML file is structured by several XML-elements, also called XML-nodes or XML-tags. XML-elements' names are enclosed by triangular brackets < > .
- Each XML-element needs to be closed either with start or with end elements as shown below: <element>....</element>.
- An XML document can have only one root element
- An XML-element can contain multiple XML-elements as its children, but the children elements must not overlap.
- In XML, all elements must be properly nested within each other.

XML attributes:

- An XML-element can have one or more attributes.
- Attribute names in XML (unlike HTML) are case sensitive. That is, *HREF* and*href* are considered two different XML attributes.
- Same attribute **cannot have two values in a syntax**

So XML follows tree structure

```
<root>
  <child>
```

```
            <subchild>.....</subchild>
          </child>
        </root>
<?xml version = "1.0" ?>
<BookStore>
    <Books>
        <PHP>
            <title>Programming PHP</title>
            <publication>O'RELLY</publication>
        </PHP>
        <PHP>
            <title>Beginners PHP</title>
            <publication>WROX</publication>
        </PHP>
    </Books>
</BookStore>
```

SimpleXML :

- SimpleXML is an extension that allows us to easily manipulate and get XML data.
- The SimpleXML extension is the tool of choice for parsing an XML document.
- SimpleXML turns an XML document into a data structure you can iterate through like a collection of arrays and objects.
- The SimpleXML extension includes interoperability with the DOM for writing XML files and built-in XPath support.
- SimpleXML is easier to code than the DOM, as its name implies.

SimpleXMLElement class represents an element in an XML document.

- To create root element of xml document, first create object of SimpleXMLElement class and initialize with root element.
- For example :
- $bk=new SimpleXMLElement("<bookstore/>");

Methods or functions of simpleXMLElement class

Function name	description	syntax	example
addChild()	The addChild() function adds a child element to the SimpleXML elemen	addChild(name,value);	$book = $bk->addchild("book");
addAttribute()	adds an attribute to the SimpleXML element.	addAttribute(name, value);	$book->addAttribute("Category" , "Technical");
getName()	Returns the name of the XML tag referenced by the SimpleXML element	getName();	$bk->getName();
asXML()	Returns a well-formed XML string (XML version 1.0) from a SimpleXML object	asXML([filename]);	echo $bk->asXML();

children()	Returns the children of a specified node as an array	children()	foreach ($book->children() as $child)
			{
			echo "Child node: " . $child . "
			";
			}

attributes();	Returns the attributes/values of an element	attributes();	foreach ($book->attributes () as $k=>$v)
			{
			echo $k : $v . "
			";
			}

count();	The count() function counts the children of a specified node.	count();	$cnt=$book->count();
simplexml_load_file()	Converts an XML file into a SimpleXMLElement object	simplexml_load_file(file)	$xml=simplexml_load_file("note.xml");
simplexml_load_string()	The simplexml_load_string() function converts a well-formed XML string into a SimpleXMLElement object.		<?php $note=<<<XML <note> <to>Tove</to> </note> XML; $xml=simplexml_load_string($note);

Reading XML document

```
<?php
$bk = simplexml_load_file("book.xml");
echo htmlspecialchars($bk->asXML());
?>
```

- With SimpleXML, all the elements in XML document are represented as tree of SimpleXMLElement objects. Any given element's children are available as properties of elements SimpleXMLElement object.

- For example ,We can access element name as properties $book->title , $book->publisher etc.

Consider an application that reads "Book.xml" file into simple XML object. Display attributes and elements.

```
        //book .xml
<?xml version='1.0' encoding='UTF-8'?>
<bookstore>
<book category="Technical">
<title> LET US C </title>
<author> YASHWANT KANETKAR </author>
<year> 1980 </year>
</book>
<book category="Cooking">
<title> COOKING EVERYDAY </title>
<author> TARALA DALAL  </author>
<year> 2000 </year>
</book>
```

```
<book category="YOGA">
<title> LIGHT ON YOGA </title>
<author> B.K.IYENGAR </author>
<year> 1990 </year>
</book>
</bookstore>

// book.php
<?php
$xml = simplexml_load_file("book.xml");

echo $xml->getName() . "<br />";

foreach($xml->children() as $child)
  {
    echo $child->getName() . "<br>";
    foreach($child->attributes() as $k=>$v)
    {
      echo $k . "=". $v . "<br>";
      foreach($child->children() as $i=>$j)
      {
              echo $i .":". $j."<br>";

     }
  }
}
?>
```

LAB Assignments:

SET A :

1) Create a XML file which gives details of movies available in "Mayanagari CD Store" from following categories a) Classical b) Action c) Horror

Elements in each category are in the following format

```
<Category>
 <Movie Name>----</Movie Name>
 <Release Year>----</Release Year>
</Category>
```

Save the file with name "movies**.xml".**

2) Create a XML file which gives details of books available in "ABC Bookstore" from following categories.

1) Technical
2) Cooking
3) Yoga

and elements in each category are in the following format

```
<Book>
    <Book_PubYear>          ----------</Book_PubYear>
    <Book_Title>    ----------</Book_Title>
    <Book_Author>----------</Book_Author>
</Book>
```

Save the file as "Book.xml"

3) Write a PHP script to create XML file named “Course.xml”

```
<Course>
 <Computer Science>
  <Student name>.....</Student name>
  <Class name>.....</Class name>
  <percentage>.....</percentage>
 </Computer Science>
</Course>
```

Store the details of 5 students who are in TYBSc.

SET B:

1) Write PHP script to generate an XML code in the following format

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<CD Store>
  <Movie>
   <Title>Mr. India</Title>
   <Release Year>1987</ Release Year >
  </Movie>
  <Movie>
   <Title>Holiday</Title>
   <Release Year>2014</ Release Year >
  </Movie>
  <Movie>
   <Title>LOC</Title>
   <Release Year>2003</ Release Year >
  </Movie>
 </CD Store>
```

2) Write a script to create “cricket.xml” file with multiple elements as shown below:

```
            <CricketTeam>
        <Team country=”India”>
      <player>____</player>
      <runs>_____</runs>
      <wicket>____</wicket>
            </Team>
          </CricketTeam>
```

Write a script to add multiple elements in “cricket.xml” file of category, country=”Australia”.

3) Write a PHP script to accept an XML file which should comprise the following:

```
<cricket>
        <player>abc</player>
        <runs>1000</runs>
        <wickets>50</wickets>
        <noofnotout>10</noofnotout>
</cricket>
```

For at least 5 players. Display the details of players who have scored more than 1000 runs and at least 50 wickets.

SET C:

1) Write a PHP script to accept an XML file which should comprise the following :

```
<languages>
 <lang name="C">
  <appeared>1972</appeared>
  <creator>Dennis Ritchie</creator>
 </lang>
 </languages>
```

For at least 5 records. Display the details of C language.

Signature of the instructor:_____________________________ Date:_______________

Assignment Evaluation

0:Not Done [ ] **2:Late Complete [ ]** **4:Complete [ ]**

1:Incomplete[ ] **3:Needs Improvement [ ]** **5:Well Done[ ]**

Assignment 9: AJAX

By Mr.Mohsin Tamboli

AJAX stands for **Asynchronous JavaScript and XML**. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script. When a user wants more content, they click a link. With AJAX, a user can click something and content can be loaded into the page, using JavaScript, **without reloading the entire page.**

Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server. With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and Update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.

AJAX cannot work independently. It is used in combination with other technologies to create interactive WebPages.

1) **JavaScript**
   - ✓ Loosely typed scripting language.
   - ✓ JavaScript function is called when an event occurs in a page.
   - ✓ Glue for the whole AJAX operation.
2) **DOM**
   - ✓ API for accessing and manipulating structured documents.
   - ✓ Represents the structure of XML and HTML documents.
3) **CSS**
   - ✓ Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript.
4) **XMLHttpRequest**
   - ✓ JavaScript object that performs asynchronous interaction with the server.

XMLHttpRequest is a JavaScript object capable of calling the server and capturing its response. It is used to send HTTP or HTTPS requests to a web server and load the server response data back into the script.

Creating an XMLHttpRequest Object :
All modern browsers (IE7+, Firefox, Chrome, Safari, and Opera) have a builtin XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

```
xmlhttp=newXMLHttpRequest();
```

When a request to a server is sent, we want to perform some actions based on the response.
The onreadystatechange event is triggered every time the readyState changes. The readyState property holds the status of the XMLHttpRequest.

Three important properties of the XMLHttpRequest object:

- **readyState** :The readyState property defines the current state of the XMLHttpRequest object.

The following table provides a list of the possible values for the readyState property:

State	Description
0	The request is not initialized.
1	The request has been set up.
2	The request has been sent.
3	The request is in process
4	The request is completed.

- **OnReadyStateChange :** Determine the function called when the objects readyState changes.

```
xmlobj.onreadystatechange=function()
{
}
```

- **responseText:**Returns the response as a string.
- **responseXML:**Returns the response as XML. This property returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.
- **Status:**Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
- **statusText:**Returns the status as a string (e.g., "Not Found" or "OK").
- **Methods of XMLHttpRequest object :**

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest objec

- **open(method, URL, async)**

  Specifies the method, URL, and other optional attributes of a request. The method parameter can have a value of "GET", "POST", or "HEAD". The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.

- **send(content):** Sends the request.
- **abort()**Cancels the current request.

LAB Assignments:

SET A:

1) Write Ajax program to read a textfile and print the contents of the file when the user clicks on the Print button.

2)Write a simple PHP program which implements Ajax for addition of two numbers

3) Write an Ajax program to display list of games stored in an array on clicking OK button.

4) Write Ajax program to carry out validation for a username entered in textbox. If the textbox is blank, print 'Enter username'. If the number of characters is less than three,print' Username is too short'. If value entered is appropriate the print 'Valid username'.

5)Write a PHP script using AJAX concept, to check user name and password are valid or Invalid (use database to store user name and password).

SET B:

1) Create employee table as follows
EMP(eno, ename, designation, salary).
Write Ajax program to select the employees name and print the selected employee's details.

2)Write Ajax program to get book details from XML file when user select a book name. Create XML file for storing details of book(title, author, year, price).

3) Write Ajax program to print Movie details by selecting an Actor's name.
Create table MOVIE and ACTOR as follows with 1 : M cardinality
MOVIE (mno, mname, release_yr) and ACTOR(ano, aname)

4) Create student table as follows
Student(sno, sname, per).
Write Ajax program to select the student name and print the selected student's details.

5) Write Ajax program to get player details from XML file when user select a player name. Create XML file for storing details of player(Country ,player name, wickets, runs).

SET C:

1) Write Ajax program to fetch suggestions when is user is typing in a textbox. (eg like google suggestions. Hint create array of suggestions and matching string will be displayed)

Signature of the instructor:_____ Date:_____

Assignment Evaluation

0:Not Done [ ] **2:Late Complete [ ]** **4:Complete [ ]**

1:Incomplete[ ] **3:Needs Improvement [ ]** **5:Well Done[ ]**