Practical 1

1A. Write a program to create a class and implement a default, overloaded and copy Constructor.

I. Default Constructor.

Program:

```
class Student3
 {
int id;
String name;
 void display()
{
System.out.println(id+" "+name);
 }
public static void main(String args[])
{
Student3 s1=new Student3();
Student3 s2=new Student3();
s1.display();
s2.display();
}
 }
```

Output:

```
0 null
0 null
```

II. Parameterized Constructor

Program:

```
class Student 4
 {
 int id;
 String name;
 Student4 (int i,String n)
 {
 id = i;
 name = n;
 }
void display()
{
System.out.println(id+" "+name);
 }
 public static void main(String args[])
 {
Student4 s1 = new Student4(111,"Karan");
Student4 s2 = new Student4(222,"Aryan");
s1.display(); s2.display();
 }
}
```

Output:

```
111 Karan
222 Aryan
```

III. Copy Constructor.

Program:

```
public class Fruit {
    private double fprice;
    private String fname;
  Fruit(double fPrice, String fName) {
        fprice = fPrice;
        fname = fName;
    }
 Fruit(Fruit fruit) {
        System.out.println("\nAfter invoking the Copy Constructor:\n");
        fprice = fruit.fprice;
        fname = fruit.fname;
    }
double showPrice() {
        return fprice;
    }
 String showName() {
        return fname;
}
  public static void main(String args[]) {
        Fruit f1 = new Fruit(399, "Ruby Roman Grapes");
        System.out.println("Name of the first fruit: " + f1.showName());
        System.out.println("Price of the first fruit: " + f1.showPrice());
        Fruit f2 = new Fruit(f1);
        System.out.println("Name of the second fruit: " + f2.showName());
        System.out.println("Price of the second fruit: " + f2.showPrice());
    }
}
```

Output:

```
Name of the first fruit: Ruby Roman Grapes
Price of the first fruit: 399.0

After invoking the Copy Constructor:
Name of the second fruit: Ruby Roman Grapes
Price of the second fruit:399.0
```

1B. Write a program to create a class and implement the concepts of Method Overloading

Program:

```
class Addition {
    public void sum()
    {
     System.out.println("No addition is performed");
    }
    public void sum (int a, int b)
   {
    System.out.println("The addition is: " + (a+b));
   }
   public void sum (int a, int b, int c)
  {
   System.out.println("The addition is:" + (a+b+c));
  }
  public void sum (double a, double b)
 {
   System.out.println("The addition is: "+ (a+b));
  }
}
 public class Dome {
  public static void main(String args[])
   {
        Addition obj = new Addition ();
         obj.sum ();
         obj.sum (2,6);
         obj.sum (4,5,6);
         obj.sum (2.3, 5.6);
   }
}
```

Output:

```
No addition is performed
```

```
The addition is: 8

The addition is:15

The addition is: 7.8999999999999995
```

1C. Write a program to create a class and implement the concepts of Static methods

Program:

```
class Cllg{
   int rollno;
   String name;
   static String college="IT";
   static void change(){
      college="LORDS";
   }
   Cllg(int r,String n){
      rollno=r;
      name=n;
   }
   void display(){
      System.out.println(rollno+ " " +name+ " "+college);
   }
}
 public class Test{
   public static void main(String args[]){
      Cllg.change();
      Cllg s1=new Cllg(111,"karan");
      Cllg s2=new Cllg(222,"Neha");
      Cllg s3=new Cllg(333,"Rohit");
      s1.display();
      s2.display();
      s3.display();
```

```
    }
}
```

Output:

```
111 karan LORDS
222 Neha LORDS
333 Rohit LORDS
```

Practical 2

2A. Write a program to implement the concepts of Inheritance and Method overriding.

I.Multilevel Inheritance

Program:

```
import java.util.*;
class Student {
    Scanner sc = new Scanner(System.in);
    String name;
    int rollNo;

    void getDetails() {
        System.out.println("Enter your name:");
        name = sc.next();
        System.out.println("Enter your Roll no:");
        rollNo = sc.nextInt();
    }
}
class Marks extends Student {
    int marks;
    void getMarks() {
        System.out.println("Enter the marks:");
        marks = sc.nextInt();
    }
}
class Result extends Marks {
    void display() {
        System.out.println("Roll No: " + rollNo);
        System.out.println("Name: " + name);
        System.out.println("Marks: " + marks);
    }
}
 public class Multilevel {
    public static void main(String[] args) {
        Result obj = new Result();
        obj.getDetails();
        obj.getMarks();
        obj.display();
    }
}
```

Output:

```
Enter your name:
neha
Enter your Roll no:
49
Enter the marks:
60
```

Roll No: 49

Name: neha

Marks: 60

II. Hierarchical Inhertiance

Program:

```
import java.util.*;
 class Account {
      Scanner sc = new Scanner(System.in);
       int user_id;
       int password;
    void getDetails() {
        System.out.println("Enter your user id:");
        user_id=sc.nextInt();
        System.out.println("Enter your password:");
        password = sc.nextInt();
    }
    void putDetails() {
        System.out.println("User ID: " + user_id);
        System.out.println("Password: " + password);
    }
 }
 class saving extends Account{
      Scanner sc = new Scanner(System.in);
       int s_account;
       int s_balance;
    void getSaving() {
        System.out.println("Enter your s_account:");
        s_account=sc.nextInt();
        System.out.println("Enter your s_balance:");
        s_balance = sc.nextInt();
    }
     void putSaving() {
        System.out.println("Saving account:" +  s_account);
        System.out.println("Saving balance: " + s_balance);
    }
}
class current extends Account{
    Scanner sc = new Scanner(System.in);
      int c_account;
      int c_balance;
    void getCurrent() {
        System.out.println("Enter your c_account:");
        c_account=sc.nextInt();
        System.out.println("Enter your c_balance:");
        c_balance = sc.nextInt();
    }
    void putCurrent() {
        System.out.println("current account:" +  c_account);
        System.out.println("current balance: " + c_balance);
    }
}
   public class Hierarchial
```

```
{
   public static void main(String[] args) {
     Account obj1=new Account();
     saving obj2=new saving();
     current obj3 =new current();
       obj1.getDetails();
       obj1.putDetails();
       obj2.getSaving();
       obj2.putSaving();
       obj3.getCurrent();
       obj3.putCurrent();
  }
}
```

Output:

```
Enter your user id:
12
Enter your password:
1234
User ID: 12
Password: 1234
Enter your s_account:
3
Enter your s_balance:
34000
Saving account:3
Saving balance: 34000
```

```
Enter your c_account:
3
Enter your c_balance:
56789
current account:3
current balance: 56789
```

III. Method overriding.

Program:

```
class Vehical {
        void run(){
        System.out.println("vehical run fast");
    }
}
class bike extends Vehical{
    @Override
    void run(){
        System.out.println("bike run fast");
    }
}
 public class road {
    public static void main(String args[]){
        bike b=new bike();
        b.run();
    }
}
```

Output:

```
bike run fast
```

2B. Write a program to implement the concepts of Abstract classes and methods

Program:

```
abstract class Person
{
 public abstract void display();
}
class EmployeeA extends Person
{
 int empno ;
 String empname, addr;
 public EmployeeA(int empno , String empname, String addr)
{
 this.empno = empno;
 this.empname= empname;
 this.addr= addr;
 }
 public void display()
{
 System.out.println("Employee Details ");
 System.out.println("Employee Id is: "+empno);
 System.out.println("Employee Name is :"+empname);
 System.out.println("Employee Address is: "+addr);
 }
}
class Worker extends Person
```

```
{
 int hours ;
 String name;
 public Worker (int hours, String name)
{
 this.hours= hours;
 this.name= name;
 }
 public void display()
{
 System.out.println("Worker Details ");
 System.out.println("Worker working hours is "+hours);
 System.out.println("Worker name is "+name);
 }
}
public class AbstractDemo2
{
 public static void main(String args[])
{
 EmployeeA e = new EmployeeA(1,"Ajay","Mumbai");
 e.display();
 Worker w = new Worker (10,"Abhi");
 w.display();
 }
 }
```
Output:

```
Employee Details
Employee Id is: 1
Employee Name is: Ajay
Employee Address is: Mumbai
Worker Details:
Worker working hours is: 10
Worker name is: Abhi
```

2C. Write a program to implement the concept of interfaces.

Program:

```java
import java.util.*;
interface Student {
   void getDetails();
}

interface Sports {
   void getSportsMarks();
}

class Result implements Student, Sports {
   Scanner sc = new Scanner(System.in);
   int id;
   String name;
   int marks;
   int sMarks;

   @Override
```

```
    public void getDetails() {
        System.out.println("Enter your id:");
        id = sc.nextInt();
        System.out.println("Enter your name:");
        name = sc.next();
        System.out.println("Enter your marks:");
        marks = sc.nextInt();
    }

    @Override
    public void getSportsMarks() {
        System.out.println("Enter your sports marks:");
        sMarks = sc.nextInt();
    }

    void display() {
        System.out.println("Id: " + id);
        System.out.println("Name: " + name);
        System.out.println("Marks: " + marks);
        System.out.println("Sports marks: " + sMarks);
    }
}

public class MultipleInheritance {
    public static void main(String args[]) {
        Result obj = new Result();
        obj.getDetails();

        obj.getSportsMarks();

        obj.display();

    }

}
```

Output:

```
Enter your id:

30

Enter your name:

neha

Enter your marks:

80

Enter your sports marks:

90

Id: 30

Name: neha

Marks: 80

Sports marks: 90
```

Practical 3 (A)

Aim: Write a program to raise built-in exceptions and raise them as per the requirements.

Code:

```
import java.util.Scanner;
class except
{
public static void main(String args[])
{
try
{
int a,b;
float c;
int x[]={10,20,30};
Scanner sc=new Scanner(System.in);
System.out.println("Enter two numbers:");
a=sc.nextInt();
b=sc.nextInt();
c=(int)a/b;
System.out.println("Division of a and b = " +c );
System.out.println("Element of array = " + x[6]);
}
catch(ArrayIndexOutOfBoundsException e1)
{
System.out.println("Array Index Out of range");
}
catch(ArithmeticException e2)
{
   System.out.println("You can't divide a number by zero");
}
catch(Exception e)
{
System.out.println("Error");
}
}
}
```

Practical No: 3 (B)

Aim: Write a program to define user defined exceptions and raise them as per the requirements

Code:

```
import java.util.*;

class InvalidAgeException extends Exception {
    public InvalidAgeException(String msg) {
        super(msg);
    }
}

public class UExcept {
    public static void main(String args[]) {
        try {
            int age;
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter your age:");
            age = sc.nextInt();
            if (age >= 18)
                System.out.println("You are eligible for voting");
            else
                throw new InvalidAgeException("You are not eligible for voting");
        } catch (InvalidAgeException e) {
            System.out.println("My own error class: " + e.getMessage());
        }
    }
}
```

Practical No: 7 (A)

Aim: Write a program for the following layout: FLOW LAYOUT

```
import java.awt.*;

import javax.swing.*;


public class FlowLayoutExample {

    JFrame frameObj;


    FlowLayoutExample() {

        frameObj = new JFrame();


        // creating the buttons

        JButton b1 = new JButton("1");

        JButton b2 = new JButton("2");

        JButton b3 = new JButton("3");

        JButton b4 = new JButton("4");

        JButton b5 = new JButton("5");

        JButton b6 = new JButton("6");

        JButton b7 = new JButton("7");

        JButton b8 = new JButton("8");

        JButton b9 = new JButton("9");

        JButton b10 = new JButton("10");


        // adding the buttons to frame

        frameObj.add(b1);

        frameObj.add(b2);

        frameObj.add(b3);

        frameObj.add(b4);

        frameObj.add(b5);

        frameObj.add(b6);

        frameObj.add(b7);

        frameObj.add(b8);
        frameObj.add(b9);

        frameObj.add(b10);


        // parameterless constructor is used, therefore, alignment is center and

        // horizontal as well as the vertical gap is 5 units.

        frameObj.setLayout(new FlowLayout());

        frameObj.setSize(300, 300);

        frameObj.setVisible(true);

    }


    public static void main(String argvs[]) {

        new FlowLayoutExample();

    }

}
```

Practical No: 7 (B)

Aim: Write program for the following layout: GRID LAYOUT

```
import java.awt.*;

import javax.swing.*;


public class MyGridLayout {

    MyGridLayout() {

        JFrame f = new JFrame();


        JButton b1 = new JButton("1");

        JButton b2 = new JButton("2");

        JButton b3 = new JButton("3");

        JButton b4 = new JButton("4");

        JButton b5 = new JButton("5");

        JButton b6 = new JButton("6");

        JButton b7 = new JButton("7");

        JButton b8 = new JButton("8");

        JButton b9 = new JButton("9");


        f.add(b1);

        f.add(b2);

        f.add(b3);
        f.add(b4);
        f.add(b5);
        f.add(b6);
        f.add(b7);
        f.add(b8);
        f.add(b9);

        f.setLayout(new GridLayout(3, 3)); // Set the layout to a 3x3 grid

        f.setSize(300, 300);
        f.setVisible(true);
    }

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

Practical No: 7

(C) Aim: Write program for the following layout: Border LAYOUT Code:

```
import java.awt.*;
import javax.swing.*;

public class Border {
    Border() {
        JFrame f = new JFrame();

        JButton b1 = new JButton("NORTH");
        JButton b2 = new JButton("SOUTH");
        JButton b3 = new JButton("EAST");
        JButton b4 = new JButton("WEST");
        JButton b5 = new JButton("CENTER");

        f.add(b1, BorderLayout.NORTH);
        f.add(b2, BorderLayout.SOUTH);
        f.add(b3, BorderLayout.EAST);
        f.add(b4, BorderLayout.WEST);
        f.add(b5, BorderLayout.CENTER);
        f.setSize(300, 300);
        f.setVisible(true);
    }

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

Practical No: 8

Aim: Write a program to demonstrate the following events. A. ActionEvent

```
import java.awt.*;
import java.awt.event.*;

public class ActionListenerExample implements ActionListener {
    TextField tf; // Declaring TextField as an instance variable to make it accessible within actionPerformed method

    public static void main(String[] args) {
        ActionListenerExample example = new ActionListenerExample(); // Creating an instance of ActionListenerExample
        example.createUI(); // Calling createUI method on the instance
    }

    public void createUI() {
        Frame f = new Frame("ActionListener Example"); // Creating Frame object
        tf = new TextField(); // Initializing TextField
        tf.setBounds(50, 50, 150, 20); // Setting bounds for TextField
        Button b = new Button("Click Here"); // Creating Button object
        b.setBounds(50, 100, 60, 30); // Setting bounds for Button
        b.addActionListener(this); // Adding ActionListener to Button
        f.add(b); // Adding Button to Frame
        f.add(tf); // Adding TextField to Frame
        f.setSize(400, 400); // Setting size for Frame
        f.setLayout(null); // Setting layout to null for absolute positioning
        f.setVisible(true); // Setting Frame visibility
    }

    public void actionPerformed(ActionEvent e) {
        tf.setText("Welcome to Javatpoint."); // Setting text in TextField
when Button is clicked
    }
}
```

B. MouseEvent

```
import java.awt.*;
import java.awt.event.*;

public class MouseListenerExample extends Frame implements MouseListener {
    Label l;

    MouseListenerExample() {
```

```
    addMouseListener(this);
    l = new Label();
    l.setBounds(20, 50, 100, 20);
    add(l);
    setSize(300, 300);
    setLayout(null);
    setVisible(true);
}

public void mouseClicked(MouseEvent e) {
    l.setText("Mouse Clicked");
}

public void mouseEntered(MouseEvent e) {
    l.setText("Mouse Entered");
}

public void mouseExited(MouseEvent e) {
    l.setText("Mouse Exited");
}

public void mousePressed(MouseEvent e) {
    l.setText("Mouse Pressed");
}

public void mouseReleased(MouseEvent e) {
    l.setText("Mouse Released");
}
```

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

C. KeyEvent

```
import java.awt.*;
import java.awt.event.*;

public class KeyListenerExample extends Frame implements KeyListener {
    Label l;
    TextArea area;

    KeyListenerExample() {
        l = new Label();
        l.setBounds(20, 50, 100, 20);
        area = new TextArea();
        area.setBounds(20, 80, 300, 300);
        area.addKeyListener(this);
        add(l);
        add(area);
        setSize(400, 400);
        setLayout(null);
        setVisible(true);
    }

    public void keyPressed(KeyEvent e) {
        l.setText("Key Pressed");
    }
    public void keyReleased(KeyEvent e) {
        l.setText("Key Released");
    }

    public void keyTyped(KeyEvent e) {
        l.setText("Key Typed");
    }

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