EXPERIMENT - 5(a)

Aim: **Write a program that implements method overriding.**

Code:

```
package javaapplication31;
class Parent {
   void show() {
      System.out.println("Parent's show()");
   }
}
class Child extends Parent
   { @Override
   void show() {
      System.out.println("Child's
      show()");
   }
}
public class JavaApplication31 {
   public static void main(String[] args)
      { Parent obj1 = new Parent();
      obj1.show();
      Parent obj2 = new Child();
      Obj2.show();
}
}
```

```
run:
Parent's show()
Child's show()
BUILD SUCCESSFUL (total time: 0 seconds)
```

EXPERIMENT - 5(b)

Aim: **Write a program to illustrate simple inheritance.**

Code:

```
package javaapplication32;
class one {
    public void print_one() {
        System.out.println("Class
        One");
    }
}
class two extends one
{
    public void print_two() {
        System.out.println("Class
        Two");
    }
}
public class JavaApplication32 {
    public static void main(String[] args)
        { two obj = new obj();
        obj.print_one();
        obj.print_two();
}
}
```

```
run:
Class One
Class Two
BUILD SUCCESSFUL (total time: 0 seconds)
```

EXPERIMENT - 5(c)

Aim: Write a program to illustrate multilevel inheritance.

Code:

```
package javaapplication33;
class one {
   public void print_one() {
      System.out.println("Class
      One");
   }
}
class two extends one {
   public void print_two()
   {
      System.out.println("Class Two");
   }
}
class three extends two {
   public void print_three()
   {
      System.out.println("Class Three");
   }
}
public class JavaApplication33 {
```

```
    public static void main(String[] args)
        { three obj = new three();
        obj.print_one();
        obj.print_two();
        obj.print_three();
    }
}
```

Output:

```
run:
Class One
Class Two
Class Three
BUILD SUCCESSFUL (total time: 0 seconds)
```

EXPERIMENT - 5(d)

Aim: **Write a program illustrating all uses of super keyword.**

Code:

```
package javaapplication34;
class Vehicle {
    int maxSpeed =
    120; void message()
    {
      System.out.println("This is Vehicle class");
    }
    Vehicle() {
      System.out.println("Constructor of Vehicle class");
    }
  }
  class Car extends Vehicle
    { int maxSpeed = 180;
    void display()    {
      System.out.println("Maximum Speed: " + super.maxSpeed);
    }
    void message() {
      System.out.println("This is Car
      class");
    }
```

```
    void display2() {
        message();
        super.message();
    }
    Car() {
        super();
        System.out.println("Constructor of Car class");
    }
}
public class JavaApplication34 {
    public static void main(String[] args)
        { Car obj = new Car();
        obj.display();
        obj.display2()
        ;
    }
}
```

OUTPUT

```
run:
Constructor of Vehicle class
Constructor of Car class
Maximum Speed: 120
This is Car class
This is Vehicle class
BUILD SUCCESSFUL (total time: 0 seconds)
```

EXPERIMENT - 6(a)

Aim: Write a program that creates an abstract class shape. Let triangle and rectangle inherit shape. Add necessary functions.

Code:

```
package javaapplication36;

import java.util.Scanner;

import java.lang.Math;

abstract class shape {

   int color;

   abstract void getSides();

   abstract void perimeter();

}

class rectangle extends shape {

   Scanner sc = new

   Scanner(System.in); int length,

   breadth;

   void getSides() {

      System.out.println("Enter the length and breadth of rectangle.");

      length = sc.nextInt();

      breadth = sc.nextInt();

   }

   void perimeter() {

      int ar = 2*(length+breadth);
```

```
        System.out.println("Perimeter is " + ar);
    }
}
class triangle extends shape {
    Scanner sc = new
    Scanner(System.in); int a,b,c;
    void getSides() {
        System.out.println("Enter the sides of triangle.");
        a = sc.nextInt();
        b =
        sc.nextInt(); c =
        sc.nextInt();
    }
    void perimeter() {
        int ar = a+b+c;
        System.out.println("Perimeter is "+ar);
    }
}
public class JavaApplication36 {
    public static void main(String[] args)
        { rectangle r = new rectangle();
        r.getSides();;
        r.perimeter();


        triangle t = new triangle();
        t.getSides();
```

```
        t.perimeter();
    }
}
```

Output:

```
run:
Enter the length and breadth of rectangle.
10 20
Perimeter is 60
Enter the sides of triangle.
10 20 30
Perimeter is 60
BUILD SUCCESSFUL (total time: 5 seconds)
```

EXPERIMENT - 6(b)

Aim: Write a program that creates an abstract class shape. Let triangle and rectangle inherit shape. Add necessary functions.

Code:

```
package javaapplication36;

import java.util.Scanner;

import java.lang.Math;

abstract class shape {

   int color;

   abstract void getSides();

   abstract void perimeter();

}

class rectangle extends shape {

   Scanner sc = new

   Scanner(System.in); int length,

   breadth;

   void getSides() {

      System.out.println("Enter the length and breadth of rectangle.");

      length = sc.nextInt();

      breadth = sc.nextInt();

   }

   void perimeter() {

      int ar = 2*(length+breadth);
```

```
        System.out.println("Perimeter is " + ar);
    }
}
class triangle extends shape {
    Scanner sc = new
    Scanner(System.in); int a,b,c;
    void getSides() {
        System.out.println("Enter the sides of triangle.");
        a = sc.nextInt();
        b =
        sc.nextInt(); c =
        sc.nextInt();
    }
    void perimeter() {
        int ar = a+b+c;
        System.out.println("Perimeter is "+ar);
    }
}
public class JavaApplication36 {
    public static void main(String[] args)
        { rectangle r = new rectangle();
        r.getSides();;
        r.perimeter();


        triangle t = new triangle();
        t.getSides();
```

```
        t.perimeter();
    }
}
```

OUTPUT

```
run:
Enter the length and breadth of rectangle.
10 20
Perimeter is 60
Enter the sides of triangle.
10 20 30
Perimeter is 60
BUILD SUCCESSFUL (total time: 5 seconds)
```

EXPERIMENT - 6(b)

Aim: **Write a java package to show dynamic polymorphism and interfaces.**

Code:

```
package javaapplication37;
import java.util.*; interface
shape {
   public void
   getSides(); public
   void area();
}
class rectangle implements shape {
   int l , b;
   Scanner sc = new
   Scanner(System.in); public void
   getSides() {
      System.out.println("Enter the sides of rectangle");
      l = sc.nextInt();
      b = sc.nextInt();
   }
   public void area()
      { int ar = l*b;
      System.out.println("The area is "+ar);
   }
}
```

```
class triangle implements shape {
   int b,h;
    Scanner sc = new
   Scanner(System.in); public void
   getSides() {
      System.out.println("Enter the base and height");
      b = sc.nextInt();
      h = sc.nextInt();
   }
   public void area()
      { int ar = b*h/2;
      System.out.println("The area is "+ar);
   }
}
class derived2 extends rectangle{
   public void getSides() {
      System.out.println("Inside derived class");
   }
}
public class JavaApplication37 {
   public static void main(String[] args)
      { rectangle r = new rectangle();
      triangle t = new triangle();
      r.getSides();
      r.area();
      t.getSides();
```

```
        t.area();
        rectangle d = new derived2();
        d.getSides();
    }
}
```

OUTPUT

```
run:
Enter the sides of rectangle
10 20
The area is 200
Enter the base and height
10 20
The area is 100
Inside derived class
BUILD SUCCESSFUL (total time: 4 seconds)
```

EXPERIMENT - 6(c)

Aim: **Write a program that creates interface and implements it.**

Code:

```
package javaapplication41;
import java.util.*; interface
shape {
   public void
   getSides(); public
   void area();
}
class rectangle implements shape {
   int l , b;
   Scanner sc = new
   Scanner(System.in); public void
   getSides() {
      System.out.println("Enter the sides of rectangle");
      l = sc.nextInt();
      b = sc.nextInt();
   }
   public void area()
      { int ar = l*b;
      System.out.println("The area is "+ar);
   }
}
```

```
class triangle implements shape {
   int b,h;
    Scanner sc = new
   Scanner(System.in); public void
   getSides() {
      System.out.println("Enter the base and height");
      b = sc.nextInt();
      h = sc.nextInt();
   }
   public void area()
      { int ar = b*h/2;
      System.out.println("The area is "+ar);
   }
}
public class JavaApplication41 {
   public static void main(String[] args)
      { rectangle r = new rectangle();
      triangle t = new triangle();
      r.getSides();
      r.area();
      t.getSides();
      t.area();
   }
}
```

```
run:
Enter the sides of rectangle
10 20
The area is 200
Enter the base and height
10 20
The area is 100
BUILD SUCCESSFUL (total time: 3 seconds)
```

EXPERIMENT - 6(d)

Aim: **Write a program to illustrate interface inheritance.**

Code:

```
package javaapplication42;
import java.util.*; interface
interfaceA {
   void Name();
}
interface interfaceB extends interfaceA {
   void Institute();
}
class class1 implements interfaceB
   { public void Name() {
      System.out.println("Inside NAme()");
   }
   public void Institute() {
      System.out.println("Inside Institute()");
   }
}
public class JavaApplication42 {
   public static void main(String[] args)
      { class1 c = new class1();
```

```
        c.Name();
        c.Institute();
    }
}
```

OUTPUT

```
run:
Inside NAme()
Inside Institute()
BUILD SUCCESSFUL (total time: 0 seconds)
```

EXPERIMENT - 7(a)

Aim: Write an application that shows how to create a user-defined exception.

Code:

```
package javaapplication2;
class myException extends Exception
   { private int detail;
   myException(int a)
      { detail = a;
   }
   public String toString() {
      return "myException[" + detail + "]";
   }
}
public class JavaApplication2 {
   static void commute(int a) throws myException {
      System.out.println("called Compute["+a+"]");
      if(a>10) {
         throw new myException(a);
      }
      System.out.println("Normal Exit");
   }
   public static void main(String[] args)
      { try {
```

```
            commute(1);
            commute(20);
        }
        catch(myException e) {
            System.out.println("Caught "+ e);
        }
    }
}
```

OUTPUT

```
run:
called Compute[1]
Normal Exit
called Compute[20]
Caught myException[20]
BUILD SUCCESSFUL (total time: 0 seconds)
```

EXPERIMENT - 7(b)

Aim: Create a customized exception and also make use of all the 5 exception keywords.

Code:

```
package javaapplication2;

class myException extends Exception

   { private int detail;

   myException(int a)

      { detail = a;

   }

   public String toString() {

      return "myException[" + detail + "]";

   }

}

public class JavaApplication2 {

   static void commute(int a) throws myException {

      System.out.println("called Compute["+a+"]");

      if(a>10) {

         throw new myException(a);

      }

      System.out.println("Normal Exit");

   }
```

```
    public static void main(String[] args)
      { try {
        commute(9);
        commute(25);
      }
      catch(myException e) {
        System.out.println("Caught "+ e);
      }
      finally {
        System.out.println("Inside Finally");
      }
    }
  }
```

OUTPUT

```
run:
called Compute[9]
Normal Exit
called Compute[25]
Caught myException[25]
Inside Finally
BUILD SUCCESSFUL (total time: 0 seconds)
```

EXPERIMENT - 7(c)

Aim: Write an Applet that displays "Hello World" (Background color-black, text color-blue and your name in the status window).

Code:

```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Color;
/*
<applet code="HelloWorld" width=200 height=60>
</applet>
*/
public class HelloWorld extends Applet {
public void init() {
      setBackground(Color.BLACK);
   }
   @Override
   public void paint(Graphics g) {
            g.setColor(Color.BLUE);
      g.drawString("Hello World", 20,
            20); showStatus("Aakash
            Garg");
   }
}
```

OUTPUT


Applet Viewer: HelloWorld
Applet
Hello World
Aakash Garg

EXPERIMENT - 7(d)

Aim: Develop an analog clock using applet.

Code:

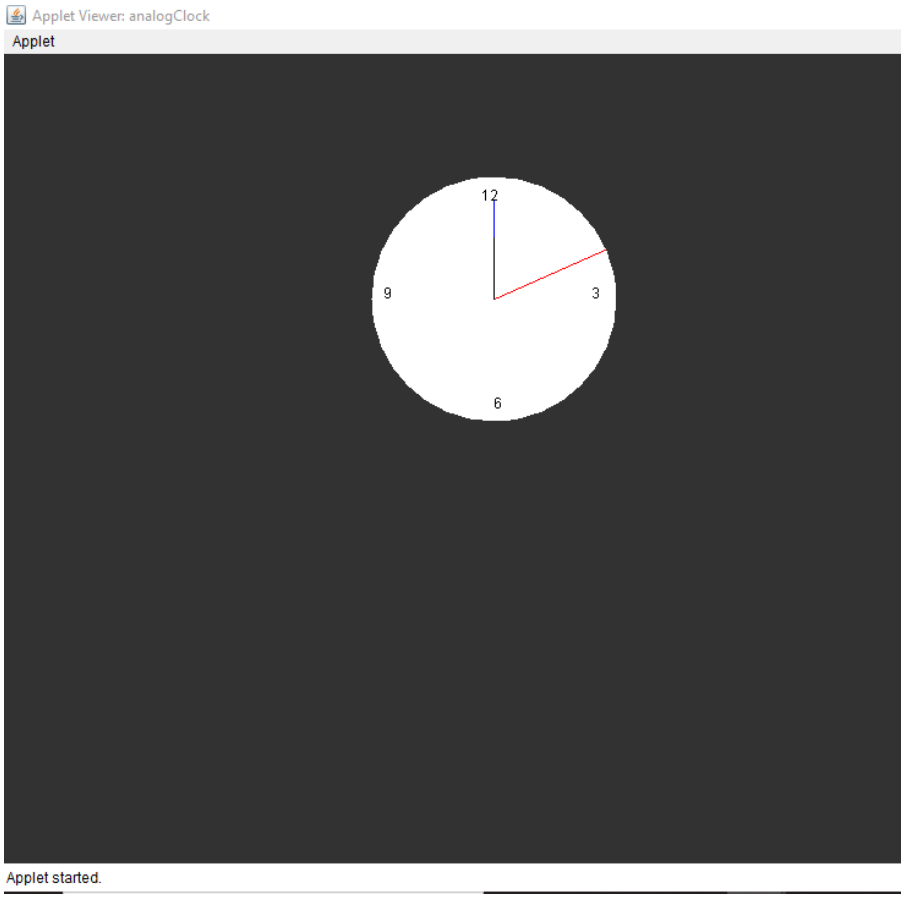
```
import java.applet.Applet;
import java.awt.*;
import java.util.*;
/*
<applet code="analogClock" width="200" height="60">
</applet>
*/
public class analogClock extends Applet
   { @Override
   public void init()
   {
      this.setSize(new Dimension(800, 400));
      setBackground(new Color(50, 50,
      50)); new Thread() {
         @Override
         public void
         run()
         {
            while (true)
               { repaint();
```

```
            }
        }
    }.start();
}
@Override
public void paint(Graphics g)
{
    Calendar time = Calendar.getInstance();
    int hour =
    time.get(Calendar.HOUR_OF_DAY); int
    minute = time.get(Calendar.MINUTE);
    int second =
    time.get(Calendar.SECOND); if (hour >
    12) {
        hour -= 12;
    }
    g.setColor(Color.white);
    g.fillOval(300, 100, 200,
    200);
    g.setColor(Color.black);
    g.drawString("12", 390,
    120);
    g.drawString("9", 310, 200);
    g.drawString("6", 400, 290);
    g.drawString("3", 480,
    200); double angle;
    int x, y;
    angle = Math.toRadians((15 - second) *
```

```
6); x = (int)(Math.cos(angle) * 100);
```

```
        y = (int)(Math.sin(angle) * 100);
        g.setColor(Color.red);
        g.drawLine(400, 200, 400 + x, 200 -
        y);
        angle = Math.toRadians((15 - minute) *
        6); x = (int)(Math.cos(angle) * 80);
        y = (int)(Math.sin(angle) * 80);
        g.setColor(Color.blue);
        g.drawLine(400, 200, 400 + x, 200 -
        y);
        angle = Math.toRadians((15 - (hour * 5)) *
        6); x = (int)(Math.cos(angle) * 50);
        y = (int)(Math.sin(angle) * 50);
        g.setColor(Color.black);
        g.drawLine(400, 200, 400 + x, 200 -
        y);
    }
}
```

OUTPUT



EXPERIMENT - 8(a)

Aim: Write a java program to show multithreaded producer and consumer application.

Code:

```
package javaapplication2;

class CubbyHole {

  private int contents;

  private boolean available = false;

  public synchronized int get() {

    while (available == false)

      { try {

        wait();

      } catch (InterruptedException e) {}

    }

    available =

    false;

    notifyAll();

    return contents;

  }

  public synchronized void put(int value)

    { while (available == true) {

      try {

        wait();
```

```
        } catch (InterruptedException e) { }
      }
      contents = value;
      available = true;
      notifyAll();
    }
}
class Consumer extends Thread
  { private CubbyHole
  cubbyhole; private int number;
  public Consumer(CubbyHole c, int number)
      { cubbyhole = c;
      this.number = number;
  }
  public void run()
      { int value = 0;
      for (int i = 0; i < 10; i++)
        { value =
        cubbyhole.get();
        System.out.println("Consumer #" + this.number + " got: " + value);
      }
  }
}
class Producer extends Thread {
  private CubbyHole
  cubbyhole; private int
  number;
```

```
  public Producer(CubbyHole c, int number)
    { cubbyhole = c;
    this.number = number;
  }
  public void run() {
    for (int i = 0; i < 10;
      i++) {
      cubbyhole.put(i);
      System.out.println("Producer #" + this.number + " put: " + i);
      try {
        sleep((int)(Math.random() * 100));
      } catch (InterruptedException e) { }
    }
  }
}
public class JavaApplication2 {
   public static void main(String[] args)
    { CubbyHole c = new CubbyHole();
    Producer p1 = new Producer(c, 1);
    Consumer c1 = new Consumer(c,
    1); p1.start();
    c1.start();
   }
}
```

OUTPUT

```
run:
Producer #1 put: 0
Consumer #1 got: 0
Producer #1 put: 1
Consumer #1 got: 1
Producer #1 put: 2
Consumer #1 got: 2
Producer #1 put: 3
Consumer #1 got: 3
Producer #1 put: 4
Consumer #1 got: 4
Producer #1 put: 5
Consumer #1 got: 5
Producer #1 put: 6
Consumer #1 got: 6
Producer #1 put: 7
Consumer #1 got: 7
Producer #1 put: 8
Consumer #1 got: 8
Producer #1 put: 9
Consumer #1 got: 9
BUILD SUCCESSFUL (total time: 0 seconds)
```

EXPERIMENT - 8(b)

Aim: Write an application that executes two threads. One thread displays "An" every 1000 milliseconds and other displays "B" every 3000 milliseconds. Create the threads by extending the Thread class.

Code:

```
package javaapplication2;

class newThread extends Thread {

   int time;

   String name;

   newThread(int myTime, String myName)

      { time = myTime;

      name = myName;

      start();

   }

   public void run()

      { try {

         for(int i=0;i<5;i++) {

            System.out.println(name);

            Thread.sleep(time);

         }

      }

      catch(InterruptedException e) {

         System.out.println("Execution Interrupted");
```

```
        }
    }
}
public class JavaApplication2 {
    public static void main(String[] args) {
      newThread ob1 = new newThread(1000,"An");
      newThread ob2 = new newThread(3000,"B");
      try{
         Thread.sleep(15000);
      }
      catch(InterruptedException e) {
         System.out.println("Execution Interrupted");
      }
    }
}
```

OUTPUT

```
run:
An
B
An
An
B
An
An
B
B
B
BUILD SUCCESSFUL (total time: 15 seconds)
```

EXPERIMENT - 9(a)

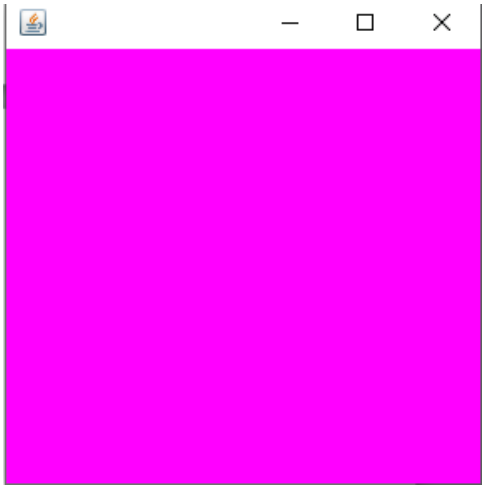
Aim: Write a program that illustrates how to process mouse click, enter, exit, press and release events. The background color changes when the mouse is entered, clicked, pressed, released or exited.

Code:

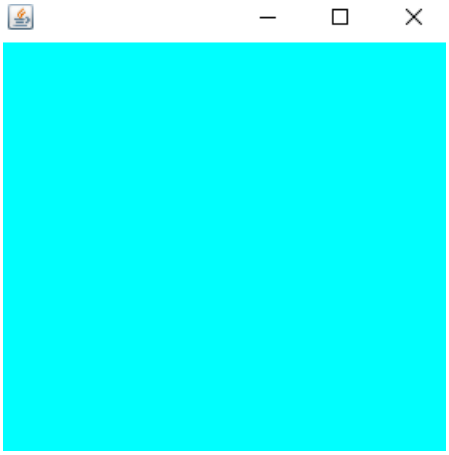
```
package javaapplication3;

import java.awt.*;

import java.awt.event.*;

public class JavaApplication3 extends Frame implements MouseListener

{

 public JavaApplication3( )

 {

  addMouseListener(this);

  setSize(300,300);

  setVisible(true);

 }

 public void mousePressed(MouseEvent e)

 {

  setBackground(Color.red);

  System.out.println("Mouse is Pressed");

 }

 public void mouseReleased(MouseEvent e)

 {
```

```
  setBackground(Color.blue);
  System.out.println("Mouse is Released");
 }
 public void mouseClicked(MouseEvent e)
 {
  setBackground(Color.green);
  System.out.println("Mouse is
  Clicked");
 }
 public void mouseEntered(MouseEvent e)
 {
  setBackground(Color.cyan);
  System.out.println("Mouse is Entered");
 }
 public void mouseExited(MouseEvent e)
 {
  setBackground(Color.magenta);
  System.out.println("Mouse is
  Exited");
 }
 public static void main(String args[])
 {
  new JavaApplication3();
 }
}
```

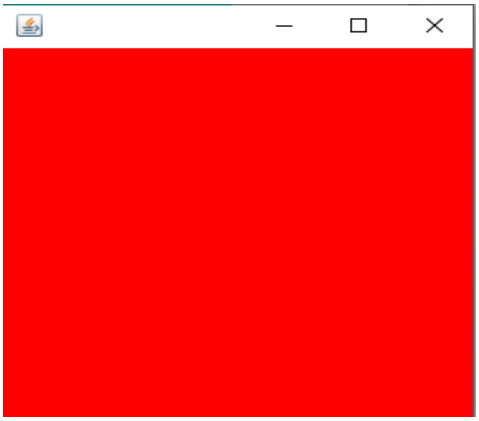
OUTPUT



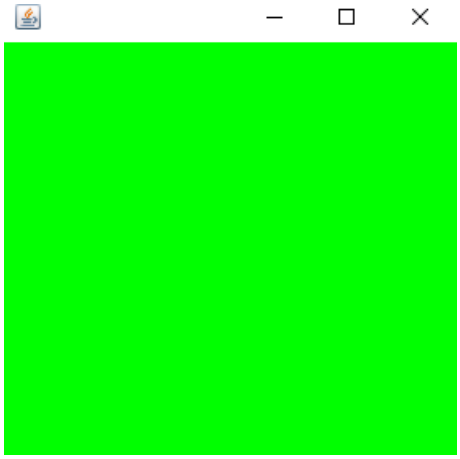
Exit



Enter



Pressed



Released

EXPERIMENT - 9(b)

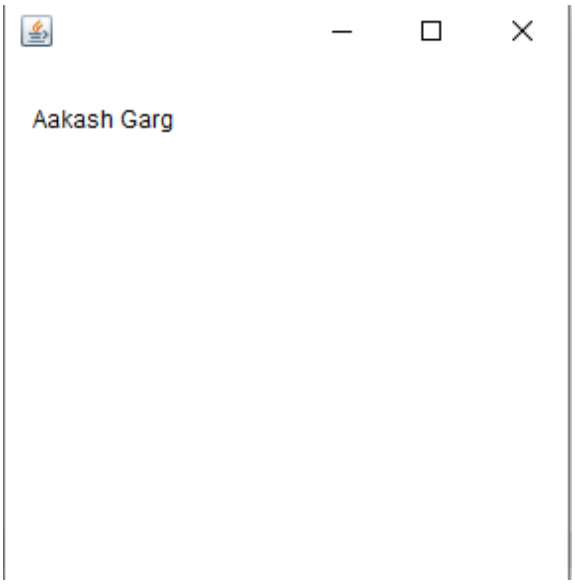
Aim: Write a program that displays your name whenever the mouse is clicked.

Code:

```
package javaapplication5;

import java.awt.*;

import java.awt.event.*;

public class JavaApplication5 extends Frame implements

   MouseListener{ Label l;

   JavaApplication5(){

      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("Aakash Garg");

   }

   public void mouseEntered(MouseEvent e) {

   }
```

```
    public void mouseExited(MouseEvent e) {
    }
    public void mousePressed(MouseEvent e) {
    }
    public void mouseReleased(MouseEvent e) {
    }
public static void main(String[] args)
    { new JavaApplication5();
}
}
```

OUTPUT



EXPERIMENT - 10(a)

Aim: Write a program that read from a file and write to file.

Code:

```
package javaapplication6;
import
java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.FileReader;
import java.io.IOException;
public class JavaApplication6
{
   public static void main(String[] args) throws IOException {
      String str = "File handling in Java. Writing a character stream using fileWriter.";
      FileWriter fw=new FileWriter("output.txt");
      for (int i = 0; i < str.length();
         i++) fw.write(str.charAt(i));
      System.out.println("Writing successful");
      fw.close();
      int ch;
      FileReader fr = new FileReader("output.txt");
      while ((ch=fr.read())!=-1)
         System.out.print((char)ch);
      fr.close();
   } }
```

OUTPUT

output - Notepad

File Edit Format View Help

```
File handling in Java. Writing a character stream using fileWriter.
```

```
run:
Writing successful
File handling in Java. Writing a character stream using fileWriter.BUILD SUCCESSFUL (total time: 0 seconds)
```

EXPERIMENT - 10(b)

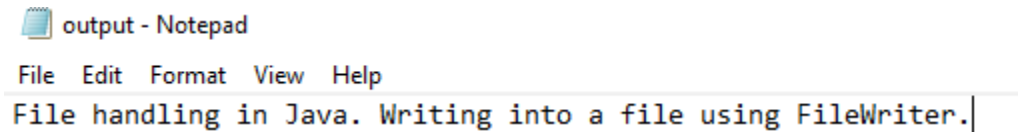
Aim: Write a program to convert the content of a given file into the uppercase content of the same file.

Code:

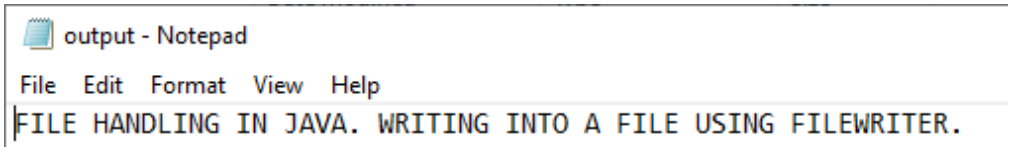
```
package javaapplication6;
import
java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.FileReader;
import java.io.IOException;
public class JavaApplication6
{
   public static void main(String[] args) throws IOException
      { int ch;
      FileWriter fw=new FileWriter("output2.txt");
      FileReader fr = new
      FileReader("output.txt");
      while((ch=fr.read())!=-1) {
         if(Character.isLowerCase(ch)) {
            fw.write(Character.toUpperCase(ch));
         }
         else {
            fw.write(ch);
         }
```

```
        fw.close();
      fr.close();
      FileReader r = new
      FileReader("output2.txt"); FileWriter w =
      new FileWriter("output.txt"); while
      ((ch=r.read())!=-1) {
         w.write(ch);
      }
      w.close();
      r.close();
      FileReader f = new
       FileReader("output.txt"); while
       ((ch=f.read())!=-1)
         System.out.print((char)ch);
      f.close();
    }
  }
```

OUTPUT




```
run:
FILE HANDLING IN JAVA. WRITING INTO A FILE USING FILEWRITER.BUILD SUCCESSFUL (total time: 0 seconds)
```



EXPERIMENT - 10(c)

Aim: JDBC (Database connectivity with MS-Access)

Code:

```
package

javaapplication8;

import java.sql.*;

public class JavaApplication8 {

   public static void main(String[]

      args) { try{

  String database="student.mdb";

  String url="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)}; DBQ=" + database +
";DriverID=22;READONLY=true";

  Class.forName("sun.jdbc.odbc.JdbcOdbcDrive

  r"); Connection

  c=DriverManager.getConnection(url);

  Statement st=c.createStatement();

  System.out.println("Database Connected

  Successfully"); ResultSet rs=st.executeQuery("select

  * from login"); while(rs.next()){

   System.out.println(rs.getString(1));

  }

}catch(Exception ee){System.out.println(ee);}

   }

}
```

OUTPUT

```
run:
Database Connected Successfully
BUILD SUCCESSFUL (total time: 0 seconds)
```