Rajasthan Technical University, Kota

Department of Computer Science Engineering

Java Programming Lab

(2018-19)

Lab File

for

6 Semester

Enrollment No.: 16/492

Name: VIKRAM SINGH GURJAR

Course: COMPUTER SCIENCE

Section: CS-4

Assignment – 1

Date of Submission:15/04/2019

22/04/2019

Objectives:

1. Multithreading

Assignment - 2

Date of Submission: 29/04/2019

Objectives:

1. Exception Handling

Assignment - 3

Date of Submission: 05/04/2019

Objectives:

1. Graphical User Interface (GUI) in Java

Sr. No.	Program	Date	Remark	Page No.
10.	Write a Program to close the frame using closing button.	08/03/2019		28

Assignment - 4

Date of Submission: 12/04/2019

Objectives:

Sr. No.	Program	Date	Remark	Page No.
11.	Create a simple signup form that connects to a database using JDBC.	05/04/2019		30
12.	Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +, -,*, % operations. Add a text field to display the result. Handle any possible exceptions like divided by zero.	05/04/2019		35
13.	Develop an applet that receives an integer in one text field, and computes its factorial value and returns it in another text field, when the button named “Compute” is clicked.	05/04/2019		43
14.	Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the textfields,	05/04/2019		46

	Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an ArithmeticException. Display the exception in a message dialog box.			
15.	Write a Java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number	05/04/2019		49
16.	Write a Java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green with radio buttons. On selecting a button, an 63-69 6 appropriate message with “Stop” or “Ready” or “Go” should appear above the buttons in selected color. Initially there is no message shown	05/04/2019		

INTRODUCTION TO MULTITHREADING

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

Advantages of Java Multithreading:

1) It doesn't block the user because threads are independent and you can perform multiple operations at the same time.
2) You can perform many operations together, so it saves time.

3) Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.

Note: At least one process is required for each thread.

How to create thread:

There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

PROGRAM NO: – 1

PROBLEM DEFINITION:

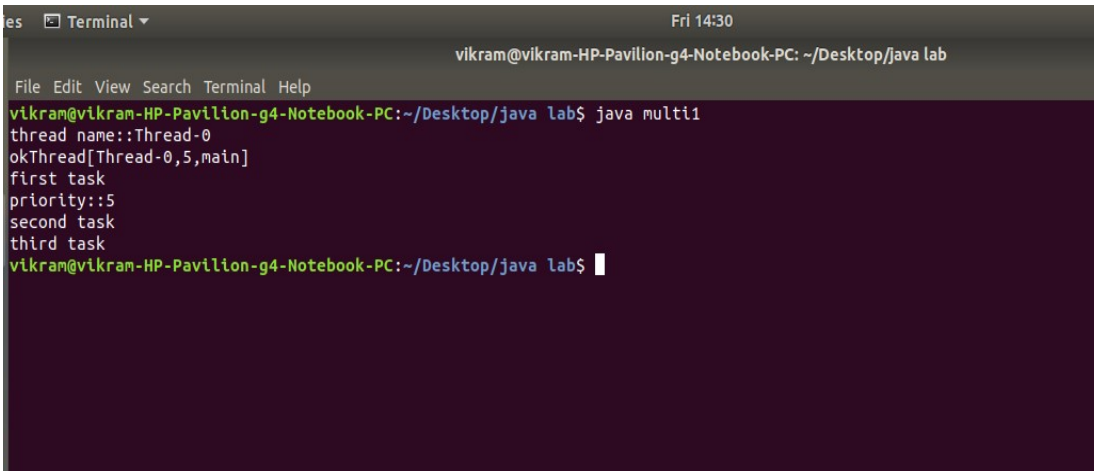
Write a Program to show multi-tasking using one thread.

IMPLEMENTATION: -

```
class multi1 extends Thread
{
public void run()
   {
System.out.println("ok"+Thread.currentThread());
fun();
try
   {
     Thread.sleep(1000);
   }
catch(Exception e)
    {
       System.out.println(e);
    }
fun2();
fun3();
}
public static void fun()
    {
         System.out.println("first task");
    }
public static void fun2()
  {
         System.out.println("second task");
  }
public static void fun3()
  {
         System.out.println("third task");
   }
public static void main(String ar[])
   {
multi1 ob=new multi1();
```

```
ob.start();
ob.getPriority();
System.out.println("thread name::"+ob.getName());
System.out.println("priority::"+ob.getPriority());
  }
}
```

OUTPUT SET: -



VIVA QUES. & ANS.:

Q.1 what is multithreading?
Ans: Multithreading in java is a process of executing multiple threads simultaneously.Java Multithreading is mostly used in games, animation, etc.

Q.2 advantage of multithreading?
Ans: i).You can perform many operations together, so it saves time.
ii) Threads are independent. At a time one thread is executed only.

Q.3 what is multitasking?
Ans:Multitasking is a process of executing multiple tasks simultaneously.

PROGRAM NO: – 2

PROBLEM DEFINITION:

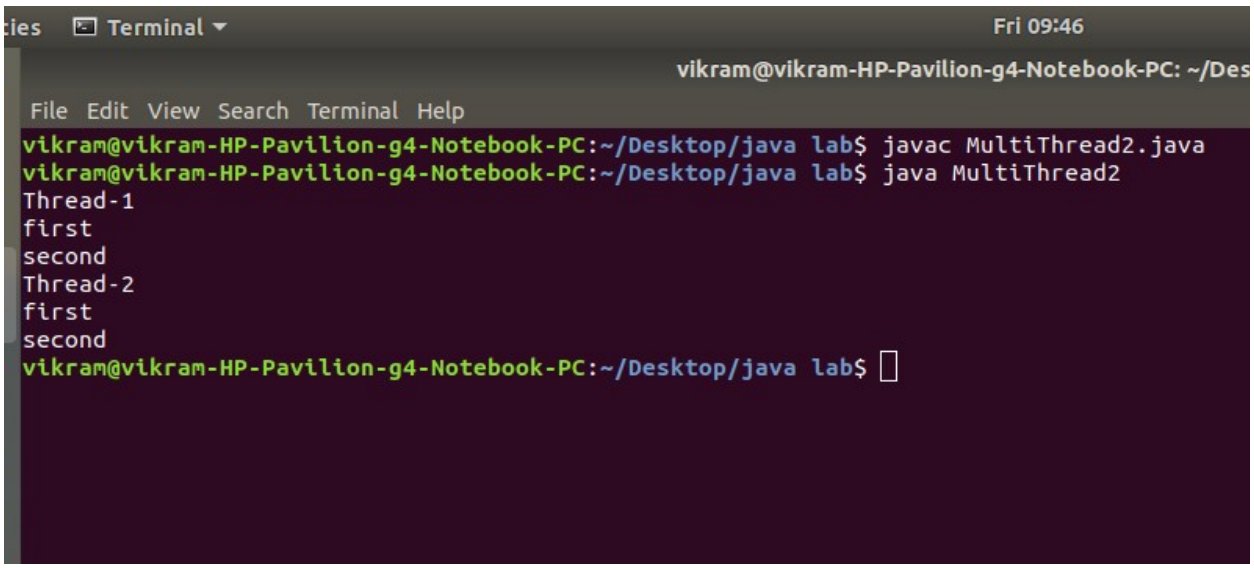
Write a Program to create two threads for one object.

IMPLEMENTATION: -

```
class MultiThread2 extends Thread
{
public void run()   {
        System.out.println(this.currentThread().getName());
     try {
             fun1();
             fun2();
         }
    catch(Exception e)      {
             System.out.println(e);
        }
    }
   public static void fun1()   {
             System.out.println("first");
    }
   public static void fun2()  {
             System.out.println("second");
    }
  public static void main(String arg[])
```

```
    {
      MultiThread2 ob=new MultiThread2();
            Thread t1 = new Thread(ob);
            Thread t2 = new Thread(ob);
            t1.start();
            t2.start();
    }
}
```

OUTPUT SET: -



PROGRAM NO: – 3

PROBLEM DEFINITION:

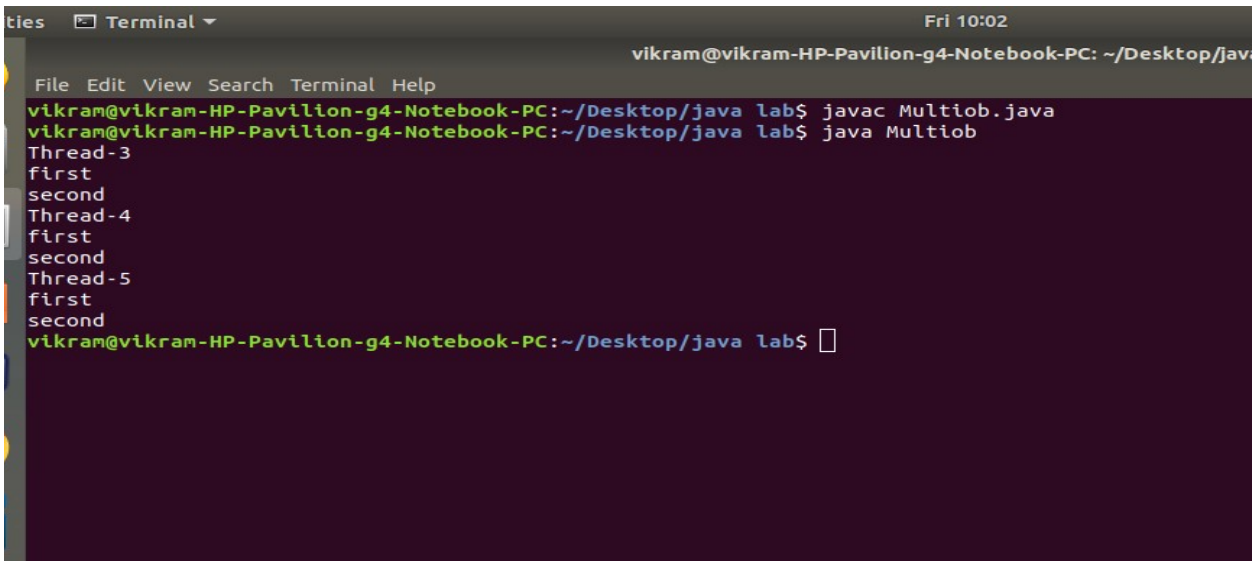
Write a Program to create a thread for each object of the class.

IMPLEMENTATION: -

```
class Multiob extends Thread
{
public void run()
   {
        System.out.println(this.currentThread().getName());
     try  {
          fun1();
          fun2();
         }
    catch(Exception e)    {
             System.out.println(e);
         }
    }
   public static void fun1()  {
             System.out.println("first");
    }
   public static void fun2()  {
             System.out.println("second");
    }
```

```
  public static void main(String arg[])
   {
     Multiob ob=new  Multiob();

     Multiob ob1=new  Multiob();
      Multiob ob2=new  Multiob();
          Thread t1 = new Thread(ob);
          t1.start();
          Thread t2 = new Thread(ob1);
            t2.start();


          Thread t3 = new Thread(ob2);
          t3.start();
   }
}
```

OUTPUT SET: -



PROGRAM NO: – 4

PROBLEM DEFINITION:

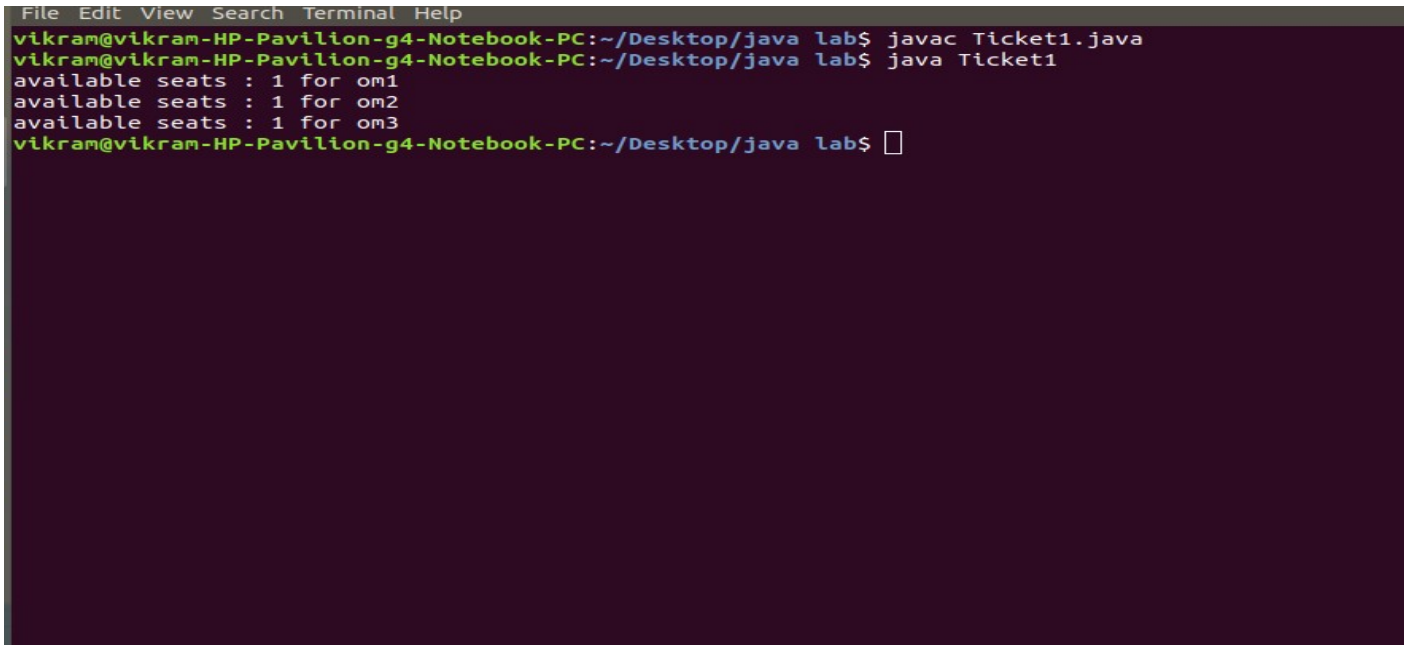
Write a Program to show that all threads are using the class procedure at the same time.

IMPLEMENTATION: -

```
class Ticket1 extends Thread
{
public void run()
   {
      int avail = 1;
      int req = 1;
      if(req <= avail)
      {
     try{
            System.out.println("available seats : "+avail+" for
"+this.currentThread().getName());
         }
    catch(Exception e)     {
             System.out.println(e);
         }
         avail = avail -1 ;
       }
    }


  public static void main(String arg[])
 {
```

```
    Ticket1 ob=new Ticket1();
          Thread t1 = new Thread(ob);
          Thread t2 = new Thread(ob);
          Thread t3 = new Thread(ob);
          t1.setName("ram1");
          t2.setName("ram2");
          t3.setName("ram3");
          t1.start();
          t2.start();
          t3.start();
    }
}
```

OUTPUT SET: -



PROGRAM NO: – 5

PROBLEM DEFINITION:

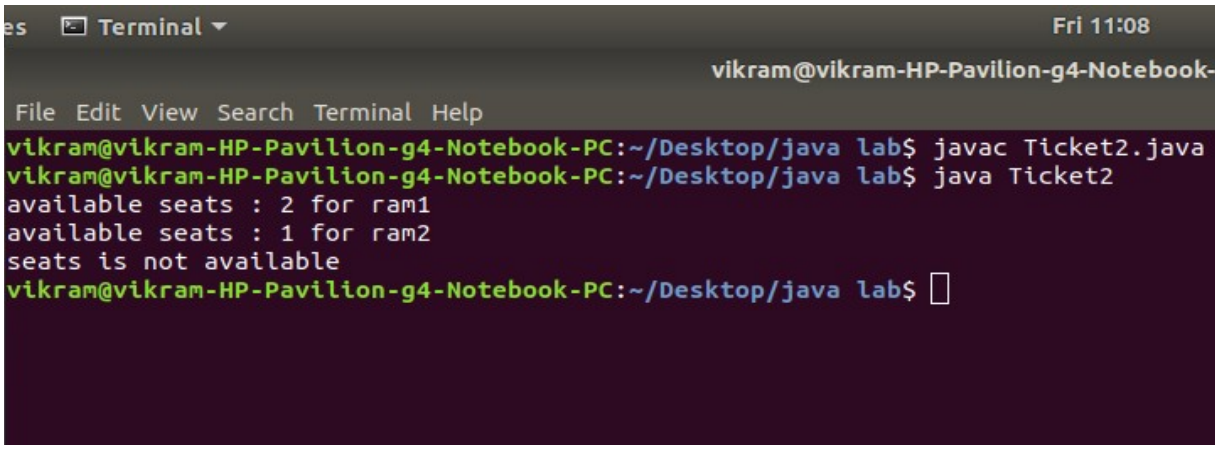
Write a Program to show that only one thread is using class at a time.

IMPLEMENTATION: -

```
class Ticket2 extends Thread
{
  int avail = 2;

      int req = 1;

public void run()
   {
    synchronized(this)
    {
        fun();
    }
  }
     void fun(){

      if(req <= avail)

      {
     try
         {
            System.out.println("available seats : "+avail+" for

 "+this.currentThread().getName());

Thread.sleep(1000);

         }

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

         avail = avail -1 ;
       }

else
       {
        System.out.println("seats is not available");
```

```
        }
    }

  public static void main(String arg[])

  {
     Ticket2 ob=new Ticket2();

          Thread t1 = new Thread(ob);

          Thread t2 = new Thread(ob);

          Thread t3 = new Thread(ob);

          t1.setName("om1");

          t2.setName("om2");

          t3.setName("hare krishna");

          t1.start();
          t2.start();

          t3.start();
    }
}
```

OUTPUT SET: -



INTRODUCTION TO EXCEPTION HANDLING

Error: An Error indicates serious problem that a reasonable application should not try to catch.

Exception: An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.

All exception and errors types are sub classes of class **Throwable**, which is base class of hierarchy. One branch is headed by **Exception**. This class is used for exceptional conditions that user programs should catch. NullPointerException is an example of such an exception. Another branch, **Error** are used by the Java run-time system(JVM) to indicate errors having to do with the run-time environment itself(JRE). StackOverflowError is an example of such an error.

Types of Exception: -

1. Built-in Exceptions (which are available in Java libraries)
2. User-defined Exceptions

Java exception handling is managed via five keywords: **try**, **catch, throw, throws**, and **finally**.

List of some Built-in Exceptions: -

1. **ArithmeticException -** It is thrown when an exceptional condition has occurred in an arithmetic operation.
2. **ArrayIndexOutOfBoundException -** It is thrown to indicate that an array has been accessed with an illegal index.
3. **ClassNotFoundException -** This Exception is raised when we try to access a class whose definition is not found
4. **FileNotFoundException -** This Exception is raised when a file is not accessible or does not open.
5. **IOException -** It is thrown when an input-output operation failed or interrupted
6. **InterruptedException -** It is thrown when a thread is waiting, sleeping, or doing some processing, and it is interrupted.
7. **NullPointerException -** This exception is raised when referring to the members of a null object. Null represents nothing
8. **NumberFormatException -** This exception is raised when a method could not convert a string into a numeric format.

9. **RuntimeException -** This represents any exception which occurs during runtime.
10. **NoSuchMethodException -** It is thrown when accessing a method which is not found.

Checked and Unchecked Exceptions: -

Checked - Are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using *throws* keyword.

Unchecked - Are the exceptions that are not checked at compiled time. In C++, all exceptions are unchecked, so it is not forced by the compiler to either handle or specify the exception. It is up to the programmers to be civilized, and specify or catch the exceptions.

In Java exceptions under *Error* and *RuntimeException* classes are unchecked exceptions, everything else under throwable is checked.

PROGRAM NO: – 6

PROBLEM DEFINITION:

Write a Program to show that exception is handled by exception class if both its exception class and parent class is given.

IMPLEMENTATION: -

```
class Assign1 {
public static void main(String[] args) {
int a, b, c;
          a=0;
          b=10;
          try{
             c=b/a;
        System.out.println(c);
          }
          catch(ArithmeticException e)  {
               System.out.println("exception is "+e);
          }
          catch(Exception e1) {
               System.out.println("exception "+e1);
          }    }
}
```

OUTPUT SET: -

```
vikram@vikram-HP-Pavilion-g4-Notebook-PC:~/Desktop/java lab$ javac Ex1.java
vikram@vikram-HP-Pavilion-g4-Notebook-PC:~/Desktop/java lab$ java Ex1
exception is java.lang.ArithmeticException: / by zero
```

PROGRAM NO: – 7

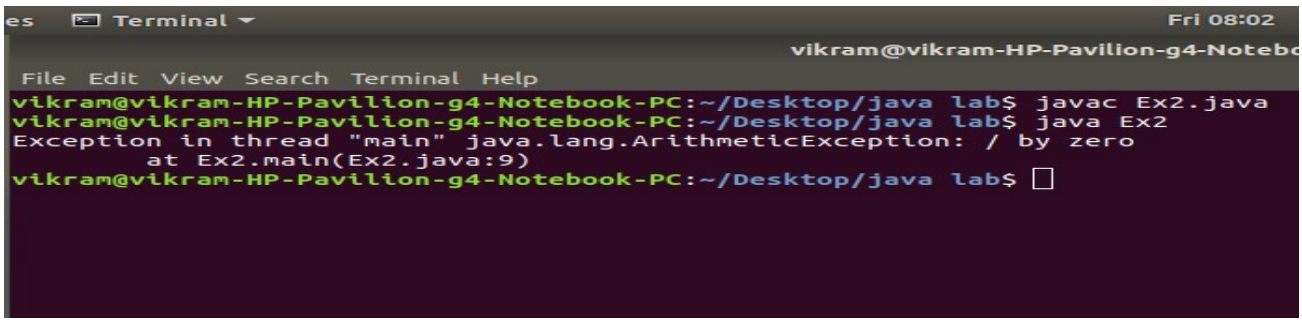
PROBLEM DEFINITION:

Write a program to show that exception stops the normal execution of the program.

IMPLEMENTATION: -

```
class Ex2
{
 public static void main(String[] args)
 {
          int a, b, c;
          a=0;
          b=10;
         c=b/a;

          System.out.println(c);
          System.out.println("hello");
          System.out.println("code after exception");
     }
}
```

OUTPUT SET: -



PROGRAM NO: – 8

PROBLEM DEFINITION:

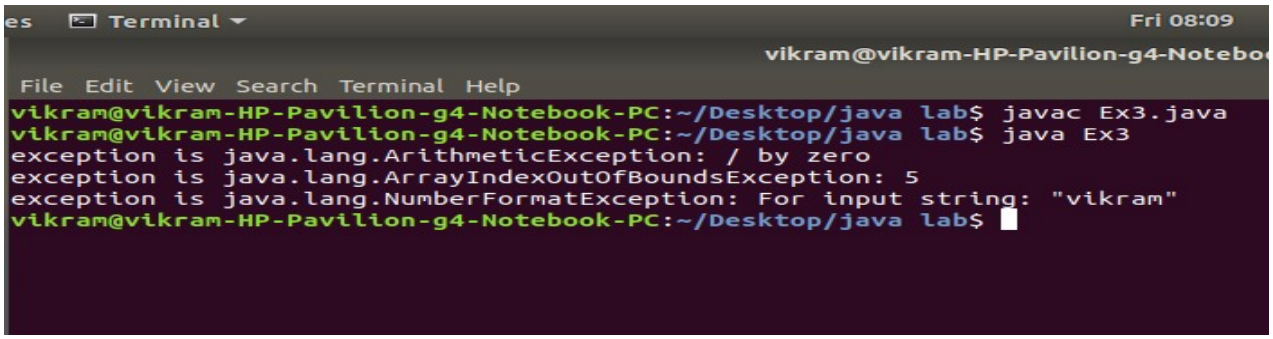
Write a program to generate ArithmeticException, ArrayIndexOutOfBoundsException, NumberFormatException.

IMPLEMENTATION

```
class Ex3
{
 public static void main(String[] args) {
          int a, b, c;
          a=0;
          b=10;
          int ab[] = new int[5];
          int k;
          try{
             c=b/a;
             System.out.println(c);
          }
          catch(ArithmeticException e   {
               System.out.println("exception is "+e);
          }
          try{
              System.out.println(ab[5]);
          }
          catch(ArrayIndexOutOfBoundsException e1)    {
```

```
                System.out.println("exception is "+e1);
            }
            try{
             k=Integer.parseInt("vikram");
            }
            catch(NumberFormatException e2){
                System.out.println("exception is "+e2);
            }
        }
}
```

OUTPUT SET: -



PROGRAM NO: – 9

PROBLEM DEFINITION:

Write a program to show that sleep() method of Thread class generate checked exception.

IMPLEMENTATION

```
public class Ex4 extends Thread
{
    public void run(){
             fun1();
             Thread.sleep(4000);
             fun2();
    }
   public static void fun1(  {
            System.out.println("first");
    }
   public static void fun2(){
            System.out.println("second");
    }
  public static void main(String arg[]){
          Ex4 ob=new Ex4();
          Thread t1 = new Thread(ob);
          t1.start();
   }
}
```

OUTPUT SET: -

```
vikram@vikram-HP-Pavilion-g4-Notebook-PC: ~/Desktop/java lab
File  Edit  View  Search  Terminal  Help
vikram@vikram-HP-Pavilion-g4-Notebook-PC:~/Desktop/java lab$ javac Ex4.java
Ex4.java:5: error: unreported exception InterruptedException; must be caught or declared to be thrown
            Thread.sleep(4000);
                        ^
1 error
vikram@vikram-HP-Pavilion-g4-Notebook-PC:~/Desktop/java lab$
```

GRAPHICAL USER INTERFACE (GUI)

Java GUI is divided into 2 parts: -

1. AWT package
2. SWING

AWT Package

Java AWT (Abstract Window Toolkit) is *an* API to develop GUI or window-based applications in java. The java.awt package provide classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckNox, Choice, List etc.
Component Class is the parent class of these sub-classes.

Useful Methods of Component Class: -

Method	Description
public void add(Component c)	inserts a component on this component.
public void setSize(int width,int height)	sets the size (width and height) of the component.
public void setLayout(LayoutManager m)	defines the layout manager for the component.
public void setVisible(boolean status)	changes the visibility of the component, by default false.

- To create simple awt example, you need a frame. There are two ways to create a frame in AWT.

  - By extending Frame class (inheritance)
  - By creating the object of Frame class (association)

Event and Listener (Java Event Handling)

Changing the state of an object is known as an event. The java.awt.event package provides many event classes and Listener interfaces for event handling.

Java Event Classes and Listener interfaces: -

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

PROGRAM NO: – 10

PROBLEM DEFINITION:

Write a program to close the frame using closing button.

IMPLEMENTATION

```
import java.awt.*;
import java.awt.event.*;
public class My2
{
     public static void main(String arg[])
     {
        Frame f = new Frame();
     f.setVisible(true);
     f.setSize(500,700);
     f.addWindowListener(new My3());

 }
}

 class My3 extends- WindowAdapter
{
public void windowClosing(WindowEvent arg) {
      System.exit(0);
}
}
```

OUTPUT SET: -

PROGRAM NO: – 11

PROBLEM DEFINITION:

Create a simple signup form that connects to a database using JDBC.

IMPLEMENTATION

```
import java.awt.*;

import java.awt.event.*;

import java.sql.*;

import java.util.*;


public class SampleForm implements ActionListener

{

        Frame f= new Frame("Sample

    Form"); Label

    user,pass,check,gender,year;

    TextField user1,pass1; Button submit;

    Checkbox check1,check2,check3,check4;

    String check11="",check21="",check31="",check41="";

    CheckboxGroup cbg;

    Choice c =       new Choice();


     SampleForm()

     {

    user=new Label("Username");       user.setBounds(50,70,
100,30);

    user1 = new TextField();    user1.setBounds(200,70,100,30);
```



```
 pass=new Label("Password");     pass.setBounds(50,110, 100,30);
```

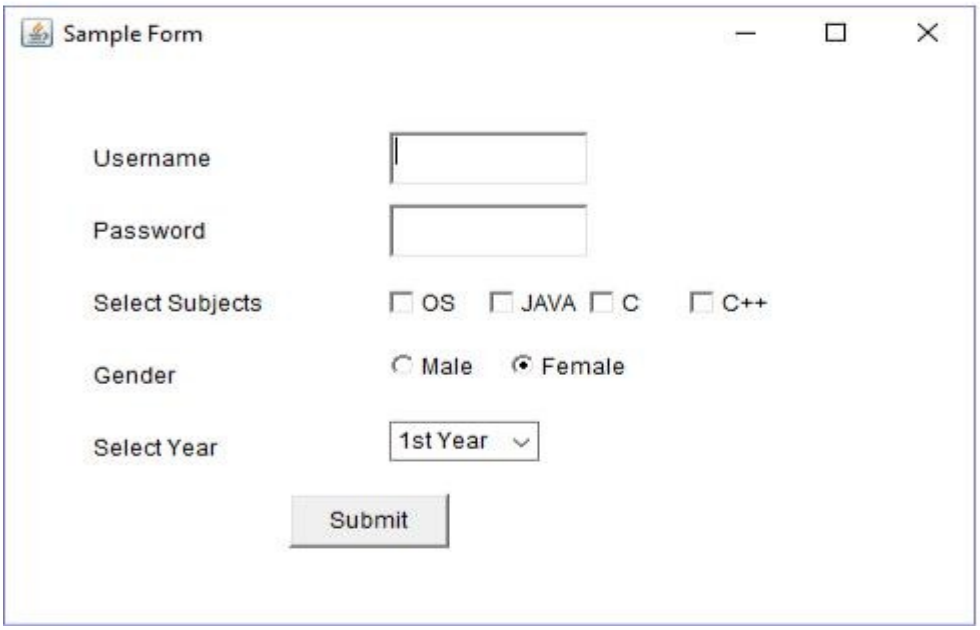
```
     pass1=new TextField();     pass1.setBounds(200,110,100,30);


    check=new Label("Select Subjects");
check.setBounds(50,150, 100,30);

    check1 = new Checkbox("OS"); check2 = new
Checkbox("JAVA"); check3 = new Checkbox("C");    check4 = new
Checkbox("C++");

    check1.setBounds(200,155,50,20);

    check2.setBounds(250,155,50,20);

    check3.setBounds(300,155,50,20);

    check4.setBounds(350,155,50,20);


    gender=new                      gender.setBounds(50,190,
Label("Gender"); 100,30);

    cbg = new CheckboxGroup();

     Checkbox checkBox1 = new Checkbox("Male", cbg, false);

        checkBox1.setBounds(200,190, 50,20);

        Checkbox checkBox2 = new Checkbox("Female", cbg, true);

        checkBox2.setBounds(260,190, 70,20);


  year=new Label("Select Year"); year.setBounds(50,230,100,30);

     c.setBounds(200,230, 75,75);

     c.add("1st Year"); c.add("2nd Year"); c.add("3rd Year");
c.add("4th Year");
```

```
    submit = new Button("Submit");

    submit.setBounds(150,270,80,30);

    submit.addActionListener(this);



    f.add(user);     f.add(pass)       f.add(user1);
f.add(pass1);     ;                    f.add(check);
                  f.add(submit);
    f.add(check1); f.add(check2); f.add(check3);
f.add(check4); f.add(gender); f.add(checkBox1);
f.add(checkBox2);

    f.add(year); f.add(c);



    f.setSize(500,350);

    f.setLayout(null);

    f.setVisible(true);

    f.addWindowListener(new ForClosing());



 }



 public void actionPerformed(ActionEvent e)

 {

     String name = user1.getText();

     String password = pass1.getText();

     String gender1 = null;

     String ab = null;

      gender1 =cbg.getSelectedCheckbox().getLabel();

      String year1= c.getItem(c.getSelectedIndex());
```

```
            Statement st = null;

            ResultSet rs = null;

            Connection con = null;

    try{

   Class.forName("com.mysql.jdbc.Driver");

   con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/
demo","root","");

      }

       Catch(Exception e2)

       {

           System.out.println(e2);

       }


       try{

              st = con.createStatement();

              st.executeUpdate("insert into demo(username,
password,gender,year)
values('"+name+"','"+password+"','"+gender1+"','"+year1+"')");

       }

        catch(Exception e4)

             {

                System.out.println(e4);

             }

 }
```

```
 public static void main(String[] args)
     { SampleForm SF = new SampleForm();
 }
}
 class ForClosing extends
WindowAdapter{ public void
windowClosing(WindowEvent arg) {
      System.exit(0); }
}
```

OUTPUT SET: -



PROGRAM NO: – 12

PROBLEM DEFINITION:

Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +, -,*, % operations. Add a text field to display the result. Handle any possible exceptions like divided by zero.

IMPLEMENTATION

```
import javax.swing.*;

import java.awt.event.*;

class Calculator implements ActionListener

{

     JFrame f;

     JTextField text;

     JButton
b1,b2,b3,b4,b5,b6,b7,b8,b9,b0,division,multi,subs,addition,decim
al,equal;


     static double a=0,b=0,result=0;

     static int operator=0;

     Calculator()

     {

          f=new JFrame("Calculator");

          text =new JTextField();

          b1=new JButton("1");           b2=new JButton("2");
     b3=new JButton("3");          b4=new JButton("4");
     b5=new JButton("5");


          b6=new JButton("6");           b0=new JButton("0");
     b8=new JButton("8");
```

```
     b7=new JButton("7");
b9=new JButton("9");


          division=new JButton("/");         multi=new
JButton("*"); subs=new JButton("-");              addition=new
JButton("+");

          decimal=new JButton(".");          equal=new
JButton("=");


          text.setBounds(30,40,280,30);

          b7.setBounds(40,100,50,40);

          b8.setBounds(110,100,50,40);

          b9.setBounds(180,100,50,40);

          division.setBounds(250,100,50,40);


          b4.setBounds(40,170,50,40);

          b5.setBounds(110,170,50,40);

          b6.setBounds(180,170,50,40);

          multi.setBounds(250,170,50,40);


          b1.setBounds(40,240,50,40);

          b2.setBounds(110,240,50,40);

          b3.setBounds(180,240,50,40);

          subs.setBounds(250,240,50,40);


          decimal.setBounds(40,310,50,40);
```

```
     b0.setBounds(110,310,50,40);

     equal.setBounds(180,310,50,40);

     addition.setBounds(250,310,50,40);



     f.add(text); f.add(b7);  f.add(b8);     f.add(b9);
f.add(division); f.add(b4);        f.add(b5);
f.add(b6);          f.add(multi);

     f.add(b1);               f.add(b2);      f.add(b3);
f.add(subs); f.add(decimal); f.add(b0);
f.add(equal); f.add(addition);



     f.setLayout(null);

     f.setVisible(true);

     f.setSize(350,450);

     f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

     f.setResizable(false);



     b1.addActionListener(this);
b2.addActionListener(this);

b3.addActionListener(this);
b4.addActionListener(this);

     b5.addActionListener(this);
b6.addActionListener(this);

b7.addActionListener(this);
b8.addActionListener(this);

     b9.addActionListener(this);
b0.addActionListener(this);
addition.addActionListener(this);
division.addActionListener(this);

     multi.addActionListener(this);
subs.addActionListener(this);
decimal.addActionListener(this);
equal.addActionListener(this);
```

```
                                                                    }
public void actionPerformed(ActionEvent e)
     {
          if(e.getSource()==b1)
               text.setText(text.getText().concat("1"));

          if(e.getSource()==b2)
               text.setText(text.getText().concat("2"));

          if(e.getSource()==b3)
               text.setText(text.getText().concat("3"));

          if(e.getSource()==b4)
               text.setText(text.getText().concat("4"));

          if(e.getSource()==b5)
               text.setText(text.getText().concat("5"));

          if(e.getSource()==b6)
               text.setText(text.getText().concat("6"));

          if(e.getSource()==b7)
               text.setText(text.getText().concat("7"));

          if(e.getSource()==b8)
```

```
        text.setText(text.getText().concat("8"));

    if(e.getSource()==b9)
        text.setText(text.getText().concat("9"));

    if(e.getSource()==b0)
        text.setText(text.getText().concat("0"));

    if(e.getSource()==decimal)
        text.setText(text.getText().concat("."));

    if(e.getSource()==addition)
    {
        a=Double.parseDouble(text.getText());
        operator=1;
        text.setText("");
    }

    if(e.getSource()==subs)
    {
        a=Double.parseDouble(text.getText());
        operator=2;
        text.setText("");
    }
```

```
if(e.getSource()==multi)
{
     a=Double.parseDouble(text.getText());
     operator=3;
     text.setText("");
}

if(e.getSource()==division)
{
     a=Double.parseDouble(text.getText());
     operator=4;
     text.setText("");
}

if(e.getSource()==equal)
{
     b=Double.parseDouble(text.getText());

     switch(operator)
     {
          case 1: result=a+b;
               break;
          case 2: result=a-b;
               break
          case 3: result=a*b;
```

```
                        break;
                    case 4:
                     if(b==0)
               {
                    JOptionPane.showMessageDialog(null,"Can not
divide by zero");
               }
               else{
                        result=a/b;
                     }
                        break
                    default: result=0;
               }
               text.setText(""+result);
          }



     }

     public static void main(String arg[])
     {
          new Calculator();
     }
}
```

OUTPUT SET: -

PROGRAM NO: – 13

PROBLEM DEFINITION:

Develop an applet that receives an integer in one text field, and computes its factorial value and returns it in another text field, when the button named “Compute” is clicked.

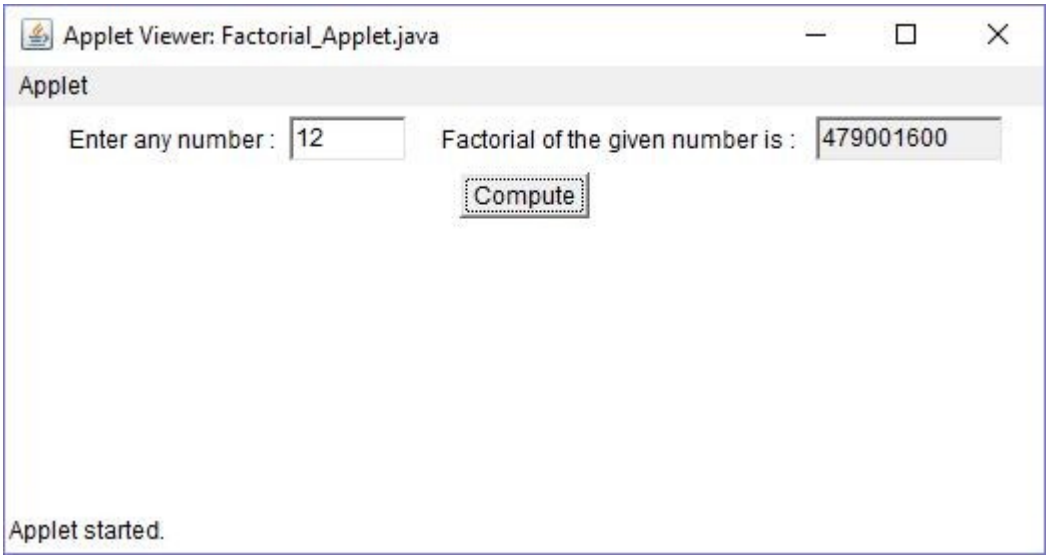
IMPLEMENTATION

```
import java.awt.*;

import java.awt.event.*;

import java.applet.*;




public class Factorial_Applet extends Applet implements
ActionListener

{

            TextField input,output;

            Button compute;

            int fact=0;

            public void init()

            {

                        compute=new Button("Compute");

Label inp=new Label("Enter any number:",Label.RIGHT);

Label opt=new Label("Factorial of the given number is :
",Label.RIGHT);

                        input=new TextField(5);

                        output=new TextField(10);

                        add(inp);
```

```
add(input);
                        add(opt);
                        add(output);
                        add(compute);
                        output.setText("0");
                        output.setEditable(false);
                        compute.addActionListener(this);
            }
            public void actionPerformed(ActionEvent ae)
            {
                        fact=1;
                        int n=Integer.parseInt(input.getText());
                        if(n<=16){
                              for(int i=n;i>=2;i--)
                               fact=fact*i;
                              output.setText(""+fact);
                        }
                        else
                        fact=-1;

                    repaint();
            }
            public void paint(Graphics g)
            {
                  if(fact==-1)
```

```
                    {
                          output.setText("0");
                          g.drawString("Sorry number exceeds
greater than 16",10,100); }
            }
}


/*
<html>
<applet code="Factorial_Applet.java"
width=500 height=200></applet>
</html>
*/
```

OUTPUT SET: -



PROGRAM NO: – 14

PROBLEM DEFINITION:

Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the textfields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an ArithmeticException. Display the exception in a message dialog box.

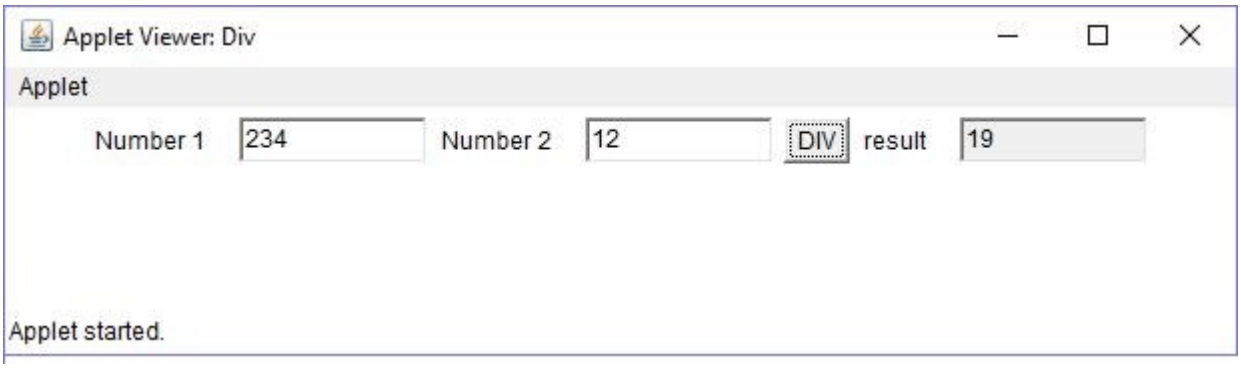
IMPLEMENTATION

```
import java.awt.*;

import java.awt.event.*;

import java.applet.*;

import javax.swing.*;


public class Div extends Applet implements ActionListener

{

    Frame f = new Frame("exception");

    TextField num1,num2,res;Label l1,l2,l3;

    Button div;

public void init()

{

    l1=new Label("Number 1"); num1=new TextField(10);

    l2=new Label("Number 2"); num2=new TextField(10);

      div=new Button("DIV");

    l3=new Label("result");res=new TextField(10);
     res.setText("0");
```

```
     res.setEditable(false);
    div.addActionListener(this);
    add(l1); add(num1); add(l2);
    add(num2); add(div);
     add(l3);       add(res);

}
public void actionPerformed(ActionEvent ae)
{
         String s1=num1.getText();
         String s2=num2.getText();
      try{
         int num1=Integer.parseInt(s1);
         int num2=Integer.parseInt(s2);

         if(num2==0)
           {
                 num1 = num1/num2;
           }
          else
            {
              int num3=num1/num2;
              res.setText(String.valueOf(num3));
             }
     }
```

```
     catch(Exception e){
                JOptionPane.showMessageDialog(f,e);
                res.setText("0");
                   }
}
}
/*
<html>
<applet code="Div"width=700 height=100>
</applet>
</html>
*/
```

OUTPUT SET: -



PROGRAM NO: – 15

PROBLEM DEFINITION:

Write a Java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

IMPLEMENTATION

```
import java.util.*;

public class MultiThreadApp

{
     public static void main(String args[])

     {
          A a = new A("one");
          a.start();
     }
}

class even implements Runnable
{
     public int x;
     public even(int x)

     {
         this.x=x;
     }

     public void run()

     {

System.out.println("Thread Name : Even Thread and "+x+"
 is even numbers &square of "+x+" is =" +(x*x));

     }
}

class odd implements Runnable

{
     public int x;
```

```
     public odd(int x)
     {
         this.x=x;
      }

      public void run()

      {

            System.out.println("Thread Name : odd Thread and
 "+x+" is odd numbers &cube of" +x+" is =" +(x*x*x));
      }
}

class A extends Thread
{
     public String tname;

     public Random r;

     public Thread t1,t2;

     A(String s)

     {
          tname=s;
      }

      public void run()
      {
         int num=0;

         r=new Random();
         try
         {
             for(int i=0;;i++)
             {
                  num=r.nextInt(1000);
                  System.out.println("Random number is "+num);
                  if(num%2==0)
                   {
                         t1=new Thread(new even(num));
                         t1.start();
                   }
                   else
                   {
                         t2=new Thread(new odd(num));
                         t2.start();
                   }
                   Thread.sleep(1000);
             }
```

```
        }
        catch(Exception e)
        {
                System.out.println(e);
        }
    }
}
```

OUTPUT SET: -

```
vikram@vikram-HP-Pavilion-g4-Notebook-PC:~$ cd Desktop
vikram@vikram-HP-Pavilion-g4-Notebook-PC:~/Desktop$ javac multi.java
vikram@vikram-HP-Pavilion-g4-Notebook-PC:~/Desktop$ java multi
Error: Could not find or load main class multi
vikram@vikram-HP-Pavilion-g4-Notebook-PC:~/Desktop$ java -cp .  multi
Random number is 572
Thread Name : Even Thread and 572 is even numbers &square of 572 is =327184
Random number is 307
Thread Name : odd Thread and 307 is odd numbers &cube of307 is =28934443
Random number is 505
Thread Name : odd Thread and 505 is odd numbers &cube of505 is =128787625
Random number is 894
Thread Name : Even Thread and 894 is even numbers &square of 894 is =799236
Random number is 887
Thread Name : odd Thread and 887 is odd numbers &cube of887 is =697864103
Random number is 816
Thread Name : Even Thread and 816 is even numbers &square of 816 is =665856
Random number is 237
Thread Name : odd Thread and 237 is odd numbers &cube of237 is =13312053
Random number is 829
Thread Name : odd Thread and 829 is odd numbers &cube of829 is =569722789
Random number is 900
Thread Name : Even Thread and 900 is even numbers &square of 900 is =810000
Random number is 401
Thread Name : odd Thread and 401 is odd numbers &cube of401 is =64481201
Random number is 456
Thread Name : Even Thread and 456 is even numbers &square of 456 is =207936
vikram@vikram-HP-Pavilion-g4-Notebook-PC:~/Desktop$
```

PROGRAM NO: – 16

PROBLEM DEFINITION:

Write a Java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green with radio buttons. On selecting a button, an appropriate message with “Stop” or “Ready” or “Go” should appear above the buttons in selected color. Initially there is no message shown.

IMPLEMENTATION

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

public class Light extends Applet implements ActionListener
{
Frame f = new Frame("exception");
TextField l1,l2,l3;
Button b1,b2,b3;
public void init()
{
b1=new Button("Red");
b2=new Button("Yellow");
b3=new Button("Green");
l1=new TextField(50);
l1.setBounds(180,60,150,20);

add(b1);   add(b2);
add(b3);    add(l1);
b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
}
public void actionPerformed(ActionEvent e)
 {
        if(e.getSource()==b1)
 l1.setText("STOP");
        if(e.getSource()==b2)
```

```
l1.setText("READY");
        if(e.getSource()==b3)
l1.setText("GO");
 }
}
/*
<html>
<applet code="Light" width=700 height=100>
</applet>
</html>
*/
```

OUTPUT SET: