JAVA
LAB RECORD

Name: Rinoy Kuriyakose

Roll no: 56 – R3

INDEX

26/08/2020

Experiment:1

Aim:

Program to find the smallest element in a Array

Concept Used:

Arrays

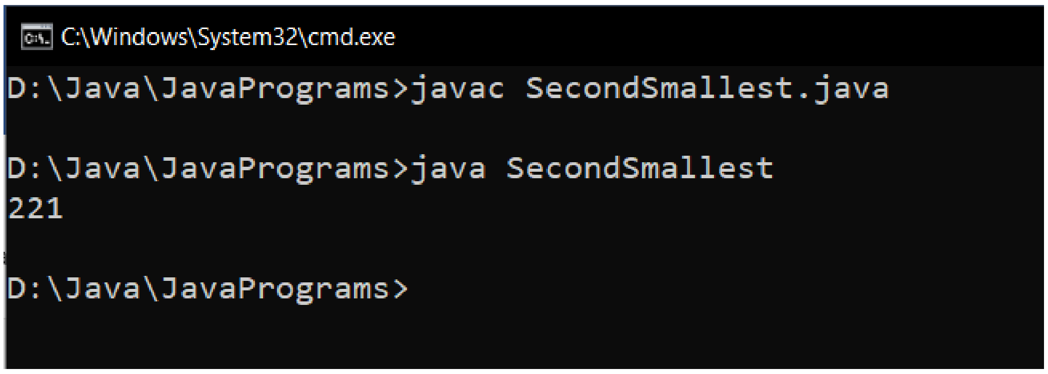
Algorithm:

```
1. Initialize the array
2. min = arr[0]
3. for i=1 to size-1
4.     if min >arr[i]
5.         temp =min
6.         min =arr[i]
7.     endif
8. endfor
9. print temp
```

Program:

```java
public class SecondSmallest {
 public static void main(String[] args) {
    int i, min,temp=0;
    int[] arr = new int[]{998, 221, 233, 412, 425, 696, 567, 548, 99, 1000};
    min = arr[0];
    for (i = 1; i < 10; i++) {
       if (min>arr[i]) {
          temp = min;
          min = arr[i];
       }
    }
    System.out.println(temp);
 }
}
```

Output:



C:\Windows\System32\cmd.exe

```
D:\Java\JavaPrograms>javac SecondSmallest.java

D:\Java\JavaPrograms>java SecondSmallest
221

D:\Java\JavaPrograms>
```

Result:

The second smallest element in an array was found successfully.

26/08/2020

Experiment:2

Aim:

Program to check whether a given number is prime or not

Concept Used:

Logic for prime number

Algorithm:

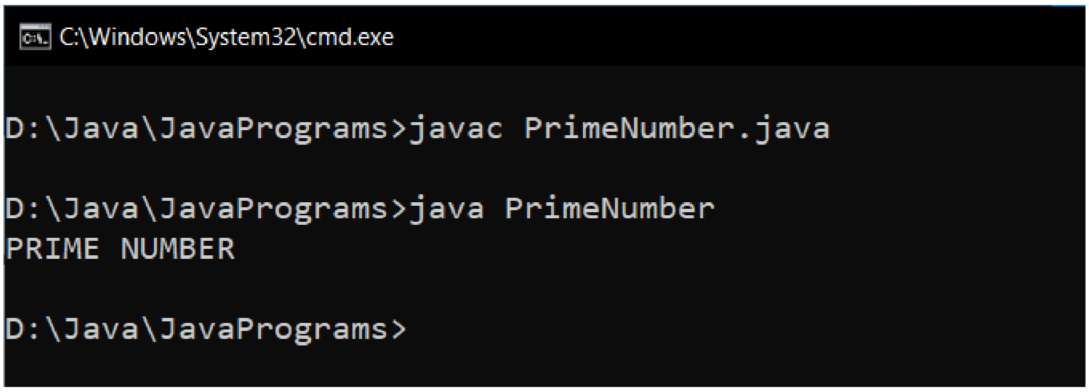
```
// num =0

1.  for i=2 to num/2
2.      if num=0 or num =1
3.          n = 1
4.      elseif num%i=0
5.          n=1
6.          break
7.      endif
8.  endfor
9.  If num =0
10.   print "PRIME NUMBER"
11. else
12.   print "NOT PRIME NUMBER"
```

Program:

```java
public class PrimeNumber {
  public static void main(String[] args) {
    int x=23,i,n=0;
    for (i=2;i<=x/2;i++)
    {
      if(x==0||x==1){
        n=1;
      }
      else if (x%i==0){
        n=1;
        break;
      }
    }
```

```
            if(n==0){
               System.out.println("PRIME NUMBER");
            }else{
               System.out.println("NOT PRIME NUMBER");
            }
      }
}
```

Output:



```
C:\Windows\System32\cmd.exe

D:\Java\JavaPrograms>javac PrimeNumber.java

D:\Java\JavaPrograms>java PrimeNumber
PRIME NUMBER

D:\Java\JavaPrograms>
```

Result:

Prime number program was implemented successfully.

26/08/2020

Experiment:3

Aim:

Program to Multiply two given matrices

Concept Used:

2-D Arrays, Matrix Multiplication

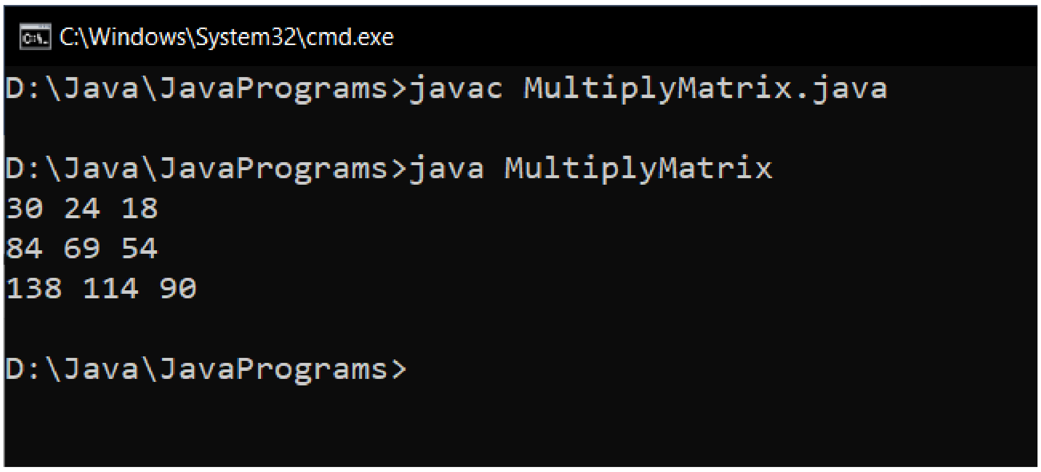
Algorithm:

```
1. for i=0 to n
2.     for j=0 to n
3.         multiply [i][j] =0
4.         for k=0 to n
5.             multiply[i][j]+=arr1[i][k]*arr2[k][j]
6.         endfor
7.         print multiply[i][j]
8.     endfor
9. endfor
```

Program:

```
public class MultiplyMatrix
{
  public static void main(String args[])
  {
    int arr1[][]={{1,2,3},{4,5,6},{7,8,9}};
    int arr2[][]={{9,8,7},{6,5,4},{3,2,1}};
    int multiply[][]=new int[3][3];

    for(int i=0;i<3;i++){
       for(int j=0;j<3;j++){
          multiply[i][j]=0;
          for(int k=0;k<3;k++){
             multiply[i][j]+=arr1[i][k]*arr2[k][j];
            }
          System.out.print(multiply[i][j]+" ");
        }
        System.out.println();
     }
  }
}
```

Output:



```
C:\Windows\System32\cmd.exe
D:\Java\JavaPrograms>javac MultiplyMatrix.java

D:\Java\JavaPrograms>java MultiplyMatrix
30 24 18
84 69 54
138 114 90

D:\Java\JavaPrograms>
```

Result:

Given two matrices were multiplied and result was displayed.

17/09/2020

Experiment:4

Aim:

Design a class to represent a bank account. Include the following members.
Data Members:

Name of the depositor

Account Number

Type of Account

Balance amount in the account

Methods

To deposit an amount

To withdraw an amount after checking balance

To display the name and balance

Incorporate default and parameterized constructor to provide initial values

Concept Used:

Constructor Chaining.

Algorithm:

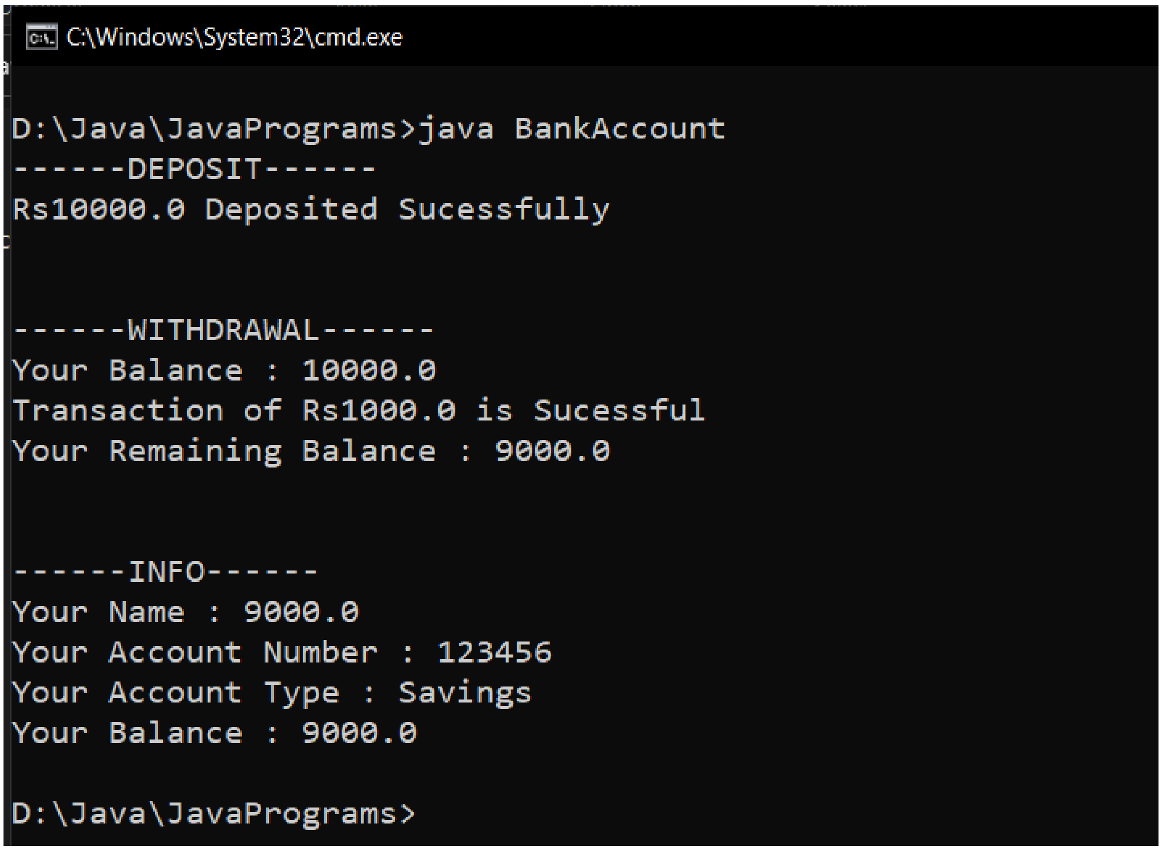
```
Algorithm deposit(x)
   1. Amt = Amt + x

Algorithm withdraw(x)
   1. Amt = Amt –x

Algorithm display()
   1. Print Name
   2. Print Amt
   3. Print AccType
   4. Print AccNumber
```

Program:

```java
public class BankAccount {
   String Name;
   int AccNumber;
   String AccType;
   float Amt;

   BankAccount() {
      Amt = 0;
   }

   BankAccount(String n,int ac,String t){
      Name=n;
      AccNumber=ac;
      AccType=t;
   }
   void deposit (float a){
      Amt=a;
      System.out.println("------DEPOSIT------");
      System.out.println("Rs"+Amt+" Deposited Sucessfully");
   }
   void withdraw(float w){
      System.out.println("\n\n------WITHDRAWAL------");
      System.out.println("Your Balance : "+ Amt);
      Amt=Amt-w;
      System.out.println("Transaction of Rs"+w+" is Sucessful");
      System.out.println("Your Remaining Balance : "+ Amt);
   }
   void info(){
      System.out.println("\n\n------INFO------");
      System.out.println("Your Name : "+ Amt);
      System.out.println("Your Account Number : "+AccNumber);
      System.out.println("Your Account Type : "+ AccType);
      System.out.println("Your Balance : "+ Amt);
   }
   public static void main (String args[]){
      BankAccount Acc1=new BankAccount("Rinoy",123456,"Savings");
      Acc1.deposit(10000);
      Acc1.withdraw(1000);
      Acc1.info();
```

```
    }
}
```

Output:


C:\Windows\System32\cmd.exe

```
D:\Java\JavaPrograms>java BankAccount
------DEPOSIT------
Rs10000.0 Deposited Sucessfully


------WITHDRAWAL------
Your Balance : 10000.0
Transaction of Rs1000.0 is Sucessful
Your Remaining Balance : 9000.0


------INFO------
Your Name : 9000.0
Your Account Number : 123456
Your Account Type : Savings
Your Balance : 9000.0

D:\Java\JavaPrograms>
```

Result :

Two bank account objects with the given fields and methods are created using default and parameterized constructors.

17/09/2020

Experiment:5

Aim:

Write a Java program which creates a class named 'Employee' having the following members: Name, Age, Phone number, Address, Salary. It also has a method named 'printSalary( )' which prints the salary of the Employee. Two classes 'Officer' and 'Manager' inherits the 'Employee' class. The 'Officer' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an officer and a manager by making an object of both of these classes and print the same.

Algorithm:

```
Class Employee
    1. Declare fields name, age, phone, address, salary

    2. Define printSalary()  method

Class Officer inherits Class Employee
    1. Declare additional field specialization

    2. Define constructors for with and without specialization

    3. Define printInfo() method

Class Manager inherits Class Employee

    1. Declare additional field department

    2. Define constructors for with and without department

    3. Define printInfo() method
```

Concepts Used :

Inhertitance, Method overloading, super keyword.

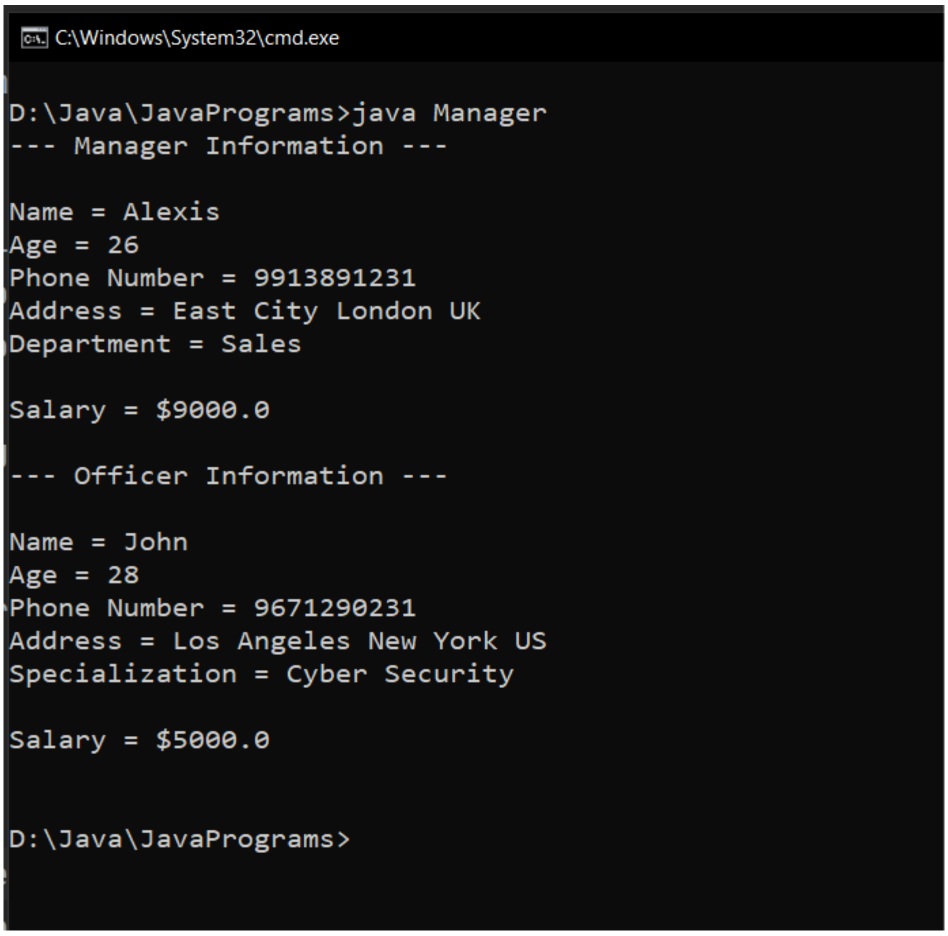
Program:

```
class Employee{
   String name;
   int age;
   String phoneNumber;
   String address;
```

```
    double salary;
    public void printSalary(){
      System.out.println("Salary = $"+salary);
      System.out.println();
    }
    Employee(String a, int b, String c, String d, double e){
        name=a;
        age=b;
        phoneNumber=c;
        address=d;
        salary=e;
    }
}
public class Manager extends Employee{
    String department;
    Manager(String a, int b, String c, String d, double e,String f){
        super(a,b,c,d,e);
        department=f;
    }
    public void printInfo(){
        System.out.println("--- Manager Information ---");
        System.out.println();
        System.out.println("Name = "+name);
        System.out.println("Age = "+age);
        System.out.println("Phone Number = "+phoneNumber);
        System.out.println("Address = "+address);
        System.out.println("Department = "+department);
        System.out.println();
    }
    public static void main(String args[]){
        Manager m1 =new Manager("Alexis",26,"9913891231","East City London
UK",9000,"Sales");
        Officer o1 =new Officer("John"  ,28,"9671290231","Los Angeles New York
US",5000,"Cyber Security");
        m1.printInfo();
        m1.printSalary();
        o1.printInfo();
        o1.printSalary();
    }

}
```

```
class Officer extends Employee{
    String specialization;
    Officer(String a, int b, String c, String d, double e,String f ){
        super(a,b,c,d,e);
        specialization=f;
    }
    public void printInfo(){
        System.out.println("--- Officer Information ---");
        System.out.println();
        System.out.println("Name = "+name);
        System.out.println("Age = "+age);
        System.out.println("Phone Number = "+phoneNumber);
        System.out.println("Address = "+address);
        System.out.println("Specialization = "+specialization);
        System.out.println();
    }
}
```

Output:



```
C:\Windows\System32\cmd.exe

D:\Java\JavaPrograms>java Manager
--- Manager Information ---

Name = Alexis
Age = 26
Phone Number = 9913891231
Address = East City London UK
Department = Sales

Salary = $9000.0

--- Officer Information ---

Name = John
Age = 28
Phone Number = 9671290231
Address = Los Angeles New York US
Specialization = Cyber Security

Salary = $5000.0


D:\Java\JavaPrograms>
```

Result:

Objects of both Officer and Manager classes are created and their details are printed.

23/09/2020

Experiment:6

Aim:

Write two Java classes Employee and Engineer. Engineer should inherit from Employee class. Employee class to have two methods display() and calcSalary(). Write a program to display the engineer salary and to display from Employee class using a single object instantiation (i.e., only one object creation is allowed). ●display() only prints the name of the class and does not return any value. Ex. “ Name of class is Employee.” ● calcSalary() in Employee displays “Salary of employee is 10000” and calcSalary() in Engineer displays “Salary of employee is 20000.”

Concepts Used:

Inhertitance, static keyword.

Algorithms:

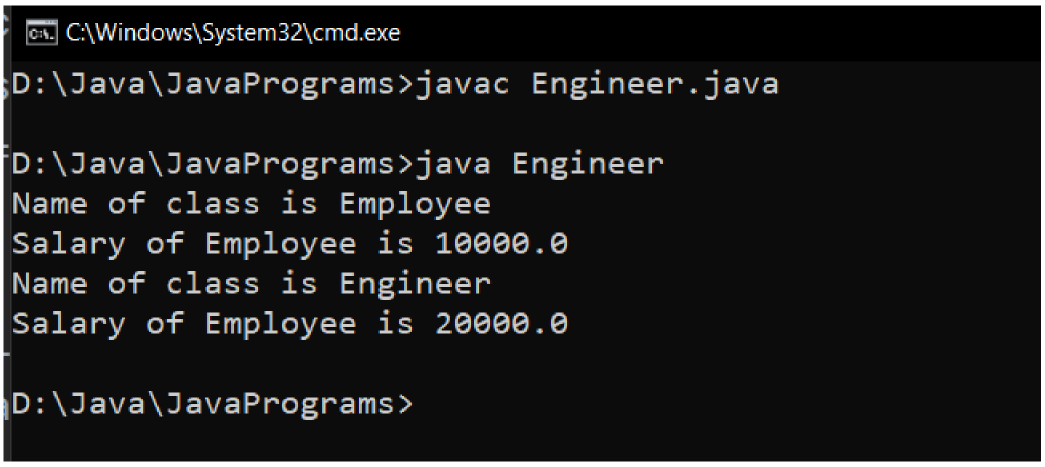
```
Class Employee
   1. Declare fields salary and className

   2. Define display method to print className

   3. Define static calcSalary method to print salary

Class Engineer inherits Class Employee
    1. Call static methods display and calcSalary

    2. Create Engineer object and call methods display and calcSalary on the object
```

Program:

```
class Employee {

  static double salary;
   static String classname;
   static void display(String clsname) {
       classname = clsname;
       System.out.println("Name of class is " + classname);
   }
   static void calcSalary(double sal) {
       salary = sal;
       System.out.println("Salary of Employee is " + sal);
```

```
    }
}
class Engineer extends Employee {

public static void main(String args[]){
        Employee.display("Employee");
        Employee.calcSalary(10000);
        Engineer obj = new Engineer();
        obj.display("Engineer");
        obj.calcSalary(20000);
    }
}
```

Output:



```
D:\Java\JavaPrograms>javac Engineer.java

D:\Java\JavaPrograms>java Engineer
Name of class is Employee
Salary of Employee is 10000.0
Name of class is Engineer
Salary of Employee is 20000.0

D:\Java\JavaPrograms>
```

Result:

Static methods of parent class Employee are accessed with and without creating objects from child class Engineer.

23/09/2020

Experiment:7

Aim:

Write a java program to create an abstract class named Shape that contains an empty method named numberOfSides( ). Provide three classes named Rectangle, Triangle and Hexagon such that each one of the classes extends the class Shape. Each one of the classes contains only the method numberOfSides( ) that shows the number of sides in the given geometrical structures.

Concepts Used:

Inheritance, Method overriding, Data abstraction using abstract keyword.

Algorithm:

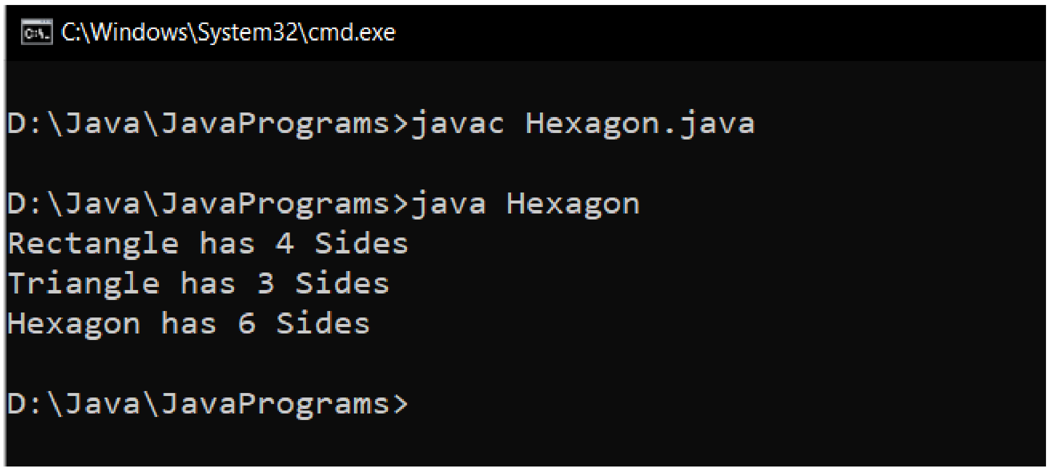
```
Abstract Class Shape
    1. Declare abstract method numOfSides()

Class Triangle inherits Class Shape
    1. Override numOfSides() method and return 3

Class Rectangle inherits Class Shape
    1. Override numOfSides ()method and return 4

Class Hexagon inherits Class Shape
    1. Override numOfSides() method and return 6
```

Program:

```
abstract class Shape{
        abstract void numberOfSides();
}
class Triangle extends Shape{
   void numberOfSides(){
      System.out.println("Triangle has 3 Sides");
   }
}
class Rectangle extends Shape{
   void numberOfSides(){
      System.out.println("Rectangle has 4 Sides");
```

```
    }
}
class Hexagon extends Shape{
    void numberOfSides(){
        System.out.println("Hexagon has 6 Sides");
    }
    public static void main(String[] args){
        Rectangle rect = new Rectangle();
        Triangle tri = new Triangle();
        Hexagon hex = new Hexagon();
        rect.numberOfSides();
        tri.numberOfSides();
        hex.numberOfSides();
    }
}
```

Output:



```
C:\Windows\System32\cmd.exe

D:\Java\JavaPrograms>javac Hexagon.java

D:\Java\JavaPrograms>java Hexagon
Rectangle has 4 Sides
Triangle has 3 Sides
Hexagon has 6 Sides

D:\Java\JavaPrograms>
```

Result:

Data abstraction and method overriding are performed on different shapes.

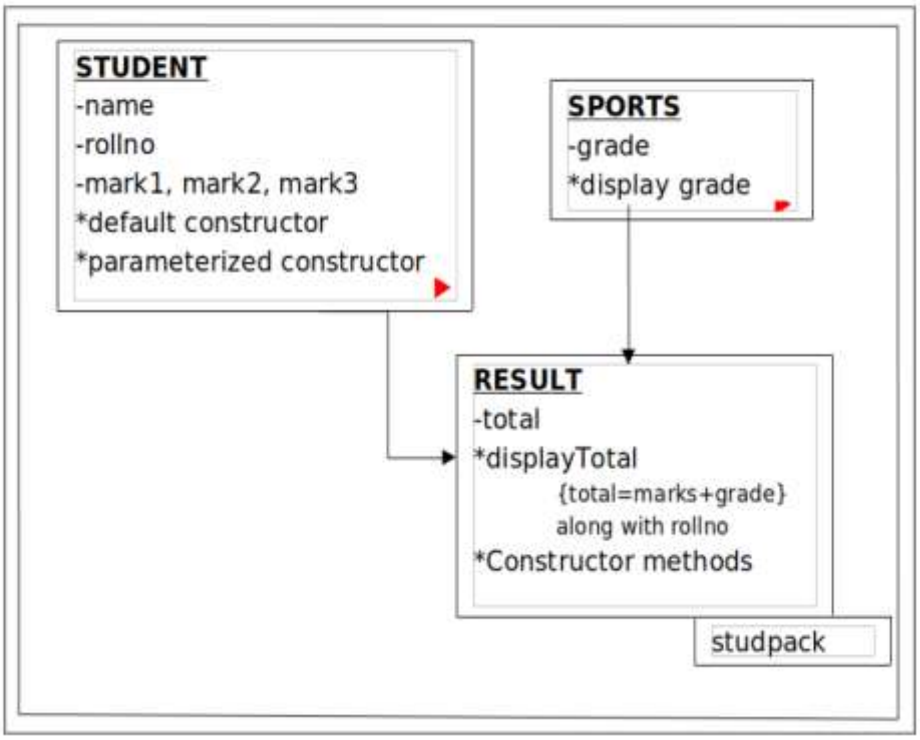
02/10/2020

Experiment:8

Aim:

Create a package ‘studpack’ which incorporates Student class and Sports interface. Create a Result class that extends Student class and implements a sports interface to display the total marks. The details of the classes and interfaces described below. Use appropriate access specifier as per the requirement. (*method, - variable)

Create a java program Hybrid.java that imports Result class from studpack and display the total for 5 students.



Concepts Used:

Package, Interface.

Algorithm:

Package studpack

1. Interface Sports with public static final field grade and public abstract method displayGrade

2. Class Student with fields name, rollNo, mark1, mark2, mark3 and required constructors

3. Class Result inherits Class Student and implements Interface Sports with field total, required constructors, definition for overriden method displayGrade, and displayTotal method

Class Hybrid

1. Import package studpack

2. Create Result object and call displayTotal method on it

Program:

package studpack

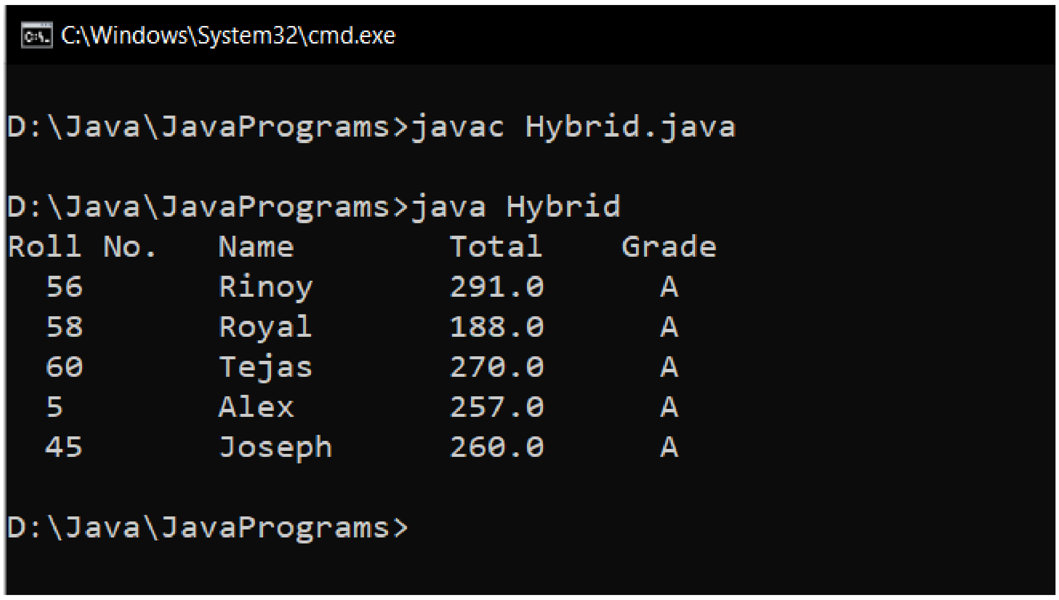
```
package studpack;

class Student {
    String name;
    int rollno;
    float mark1,mark2,mark3;
    Student(){
        mark1 = 0;
        mark2 = 0;
        mark3 = 0;
        rollno = 0;
    }
    Student(String name,int rollno,float mark1,float mark2,float mark3){
        this.name = name;
        this.mark1 = mark1;
        this.mark2 = mark2;
        this.mark3 = mark3;
        this.rollno = rollno;
    }
}

interface Sports{
    char grade='A';
    char displayGrade();
}

public class Result extends Student implements Sports{

    float total;
```

```java
    public char displayGrade(){
        return grade;
    }
    public void displayTotal(){
        total=mark1+mark2+mark3;
        System.out.println("  "+rollno+"     "+name+"     "+total+"     "+displayGrade());
    }
    public Result(String name,int rollno,float mark1,float mark2,float mark3){
        super(name,rollno,mark1,mark2,mark3);
    }
}
```

Hybrid.java

```java
import studpack.*;

public class Hybrid{
    public static void main(String[] args){
         Result S1 =new Result(" Rinoy ",56,97,96,98);
         Result S2 =new Result(" Royal ",58,78,60,50);
         Result S3 =new Result(" Tejas ",60,90,89,91);
         Result S4 =new Result("  Alex  ",05,83,89,85);
         Result S5 =new Result(" Joseph",45,90,81,89);
         System.out.println("Roll No.   Name        Total    Grade");
         S1.displayTotal();
         S2.displayTotal();
         S3.displayTotal();
         S4.displayTotal();
         S5.displayTotal();
    }
}
```

Output:



C:\Windows\System32\cmd.exe

```
D:\Java\JavaPrograms>javac Hybrid.java

D:\Java\JavaPrograms>java Hybrid
Roll No.    Name         Total     Grade
  56        Rinoy        291.0       A
  58        Royal        188.0       A
  60        Tejas        270.0       A
  5         Alex         257.0       A
  45        Joseph       260.0       A

D:\Java\JavaPrograms>
```

Result:

Package studpack is imported to another package and its classes are used there.

17/10/2020

Experiment : 9

Aim:

Write a Java program that shows the usage of try, catch, throws and finally.

(a) Try-Finally example

(b) Multiple catch example (3 catches for a single try)

(c) Nested Try (3 levels of nesting)

(d) Throw an exception when there are no sufficient arguments passed

into command line as input for adding two numbers.

(e) Throws example (for handling two exceptions in a method)

Concept Used:

Exception Handling.

Algorithm:

Algorithm (a)

```
1. try block –Print a/0
2. finally block –Print message
```

Algorithm (b)

```
1. try block –Print arr[size+1], arr[index]/0.
2. catch block 1 – Print IndexOutOfBoundsException object
3. catch block 2 – Print ArithmeticException object
4. catch block 3 – Print Exception object
```

Algorithm (c)

```
1. try –
2. Print arr[index]
3.     try –
4.     Print arr[index]/10
5.         try –
```

```
6.                  Print arr[index]/0
7.                 catch –Print ArithematicException object
8.                 catch –Print Exception object
9.           catch –Print ArithematicException object
10. catch –Print ArrayIndexOutOfBoundsException object
```

Algorithm (d)

```
1. if(input.length != 2)
2.      throw RuntimeException(“Enter two numbers”)
3. else
4.      Print Integer.parseInt(input[0])+Integer.parseInt(input[1])
5. endif
```

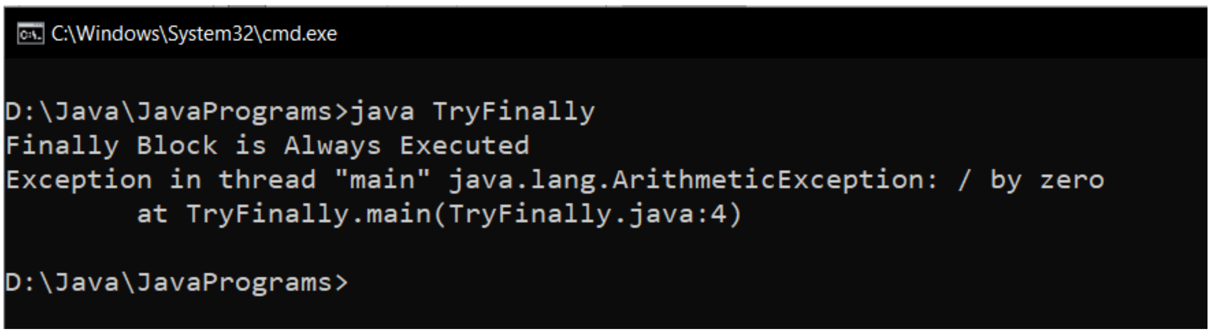
Algorithm (e)

```
1. if(n<10)
2.       throw ArithematicException
3.  else
4.       print Number Greater than 10
```

Program (a): Try-Finally example

```java
public class TryFinally{
   public static void main(String args[]){
      try{
         int a=5/0;
         System.out.println(a);
      }
      finally{
         System.out.println("Finally Block is Always Executed");
      }
   }
}
```

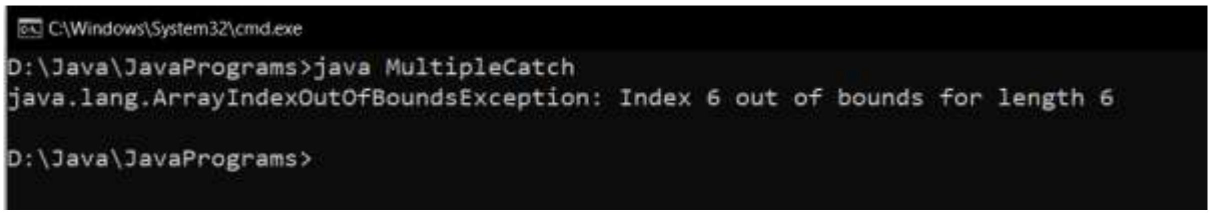
Output (a):


```
C:\Windows\System32\cmd.exe

D:\Java\JavaPrograms>java TryFinally
Finally Block is Always Executed
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at TryFinally.main(TryFinally.java:4)

D:\Java\JavaPrograms>
```

Program (b): Multiple catch example

```
public class MultipleCatch {
   public static void main(String args[]) {
      int[] arr = {67, 20, 39, 2, 10, 18};
      try {
         int res = arr[6];
         int div = arr[2]/0;
         System.out.println(res);
      }
      catch(ArrayIndexOutOfBoundsException exp1){
         System.out.println(exp1);

      }
      catch(ArithmeticException exp2) {
         System.out.println(exp2);
      }

      catch(Exception exp3) {
         System.out.println(exp3);
      }

   }
}
```

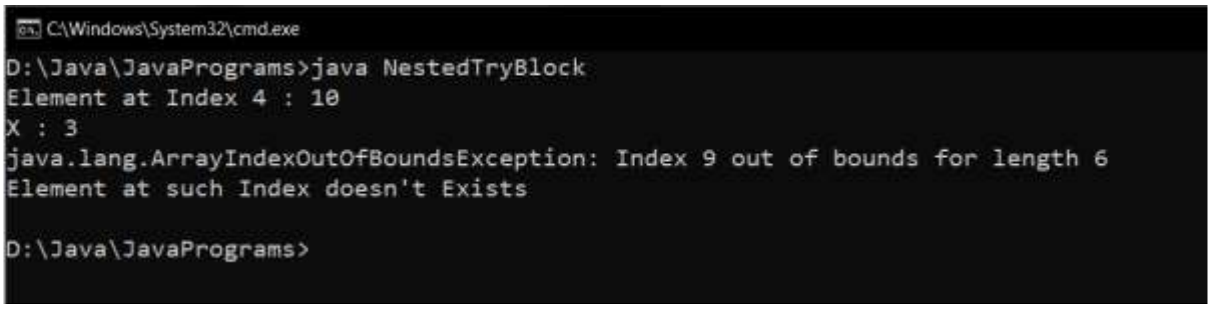
Output (b):


```
C:\Windows\System32\cmd.exe
D:\Java\JavaPrograms>java MultipleCatch
java.lang.ArrayIndexOutOfBoundsException: Index 6 out of bounds for length 6

D:\Java\JavaPrograms>
```

Program (c): Nested Try Block

```
class NestedTryBlock{

   public static void main(String args[]){

      try {

         int[] arr = {67, 20, 39, 2, 10, 18};
         System.out.println("Element at Index 4 : " +arr[4]);
         try {

            int x = arr[2]/ 10;
            System.out.println("X : " +x);
            try {

               int y = arr[9]/ 0;
               System.out.println("Y : " +y);
            }
            catch (ArrayIndexOutOfBoundsException e3) {
               System.out.println(e3);
               System.out.println("Element at such Index doesn't Exists");
            }
            catch (Exception e4) {
               System.out.println(e4);
            }
         }
         catch (ArithmeticException e2) {
            System.out.println(e2);
         }
      }
      catch (ArrayIndexOutOfBoundsException e1) {
         System.out.println(e1);
         System.out.println("Element at such Index doesn't Exists");
      }
   }
}
```

Output (c):

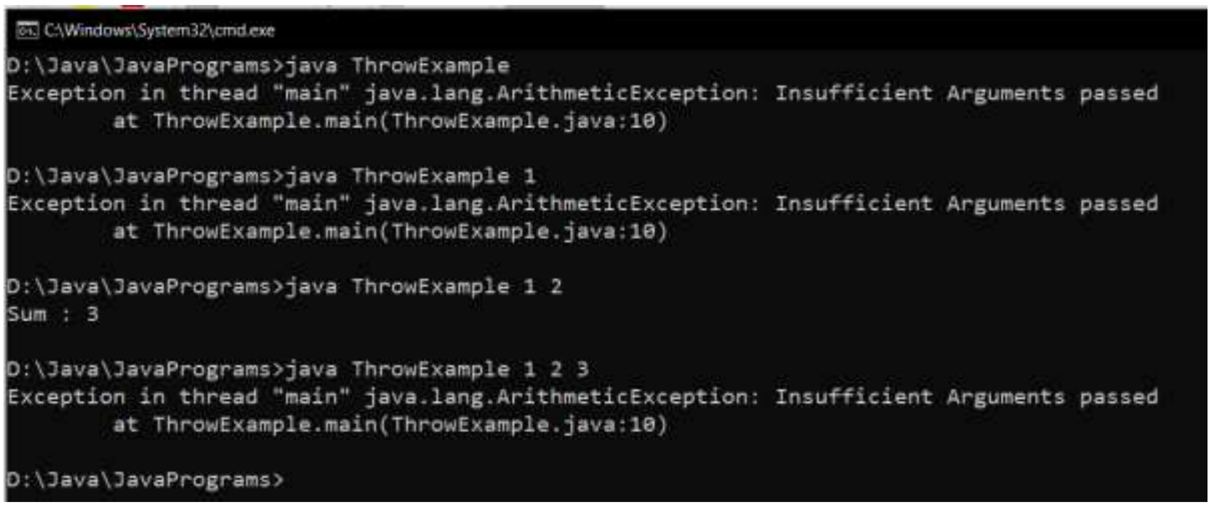


```
D:\Java\JavaPrograms>java NestedTryBlock
Element at Index 4 : 10
X : 3
java.lang.ArrayIndexOutOfBoundsException: Index 9 out of bounds for length 6
Element at such Index doesn't Exists

D:\Java\JavaPrograms>
```

Program (d): Throw an exception when there are no sufficient arguments passed into command line as input for adding two numbers

```
public class ThrowExample {

   public static void main(String[] args) {
      int x=0 ;
       for (String arg:args) {
          x++;
       }

       if (x!=2) {
          throw new ArithmeticException("Insufficient Arguments passed");
       }
       else {
          int sum = Integer.parseInt(args[0])+Integer.parseInt(args[1]);
          System.out.println("Sum : "+ sum);
       }
   }
}
```

Output (d):

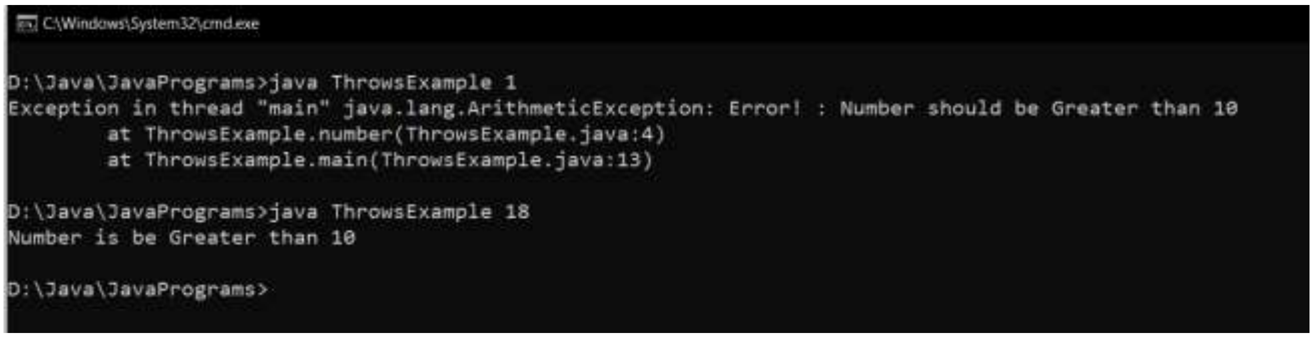


```
D:\Java\JavaPrograms>java ThrowExample
Exception in thread "main" java.lang.ArithmeticException: Insufficient Arguments passed
        at ThrowExample.main(ThrowExample.java:10)

D:\Java\JavaPrograms>java ThrowExample 1
Exception in thread "main" java.lang.ArithmeticException: Insufficient Arguments passed
        at ThrowExample.main(ThrowExample.java:10)

D:\Java\JavaPrograms>java ThrowExample 1 2
Sum : 3

D:\Java\JavaPrograms>java ThrowExample 1 2 3
Exception in thread "main" java.lang.ArithmeticException: Insufficient Arguments passed
        at ThrowExample.main(ThrowExample.java:10)

D:\Java\JavaPrograms>
```

Program (e): Throws example

```
public class ThrowsExample {
  static void number(int x) throws ArithmeticException {
      if (x < 10) {
         throw new ArithmeticException("Error! : Number should be Greater than 10");
      }
      else {
         System.out.println("Number is be Greater than 10");
      }
   }

   public static void main(String[] args) {
      int y=Integer.parseInt(args[0]);
      number(y);
   }
}
```

Output (e):



C:\Windows\System32\cmd.exe

```
D:\Java\JavaPrograms>java ThrowsExample 1
Exception in thread "main" java.lang.ArithmeticException: Error! : Number should be Greater than 10
        at ThrowsExample.number(ThrowsExample.java:4)
        at ThrowsExample.main(ThrowsExample.java:13)

D:\Java\JavaPrograms>java ThrowsExample 18
Number is be Greater than 10

D:\Java\JavaPrograms>
```

Result:

Exception handling is carried out in a Java program.

23/10/2020

Experiment: 10

Aim:

Write a Java program that read from a file and write to file by handling all file related exceptions

Concepts Used:

File Handling.

Algorithm:

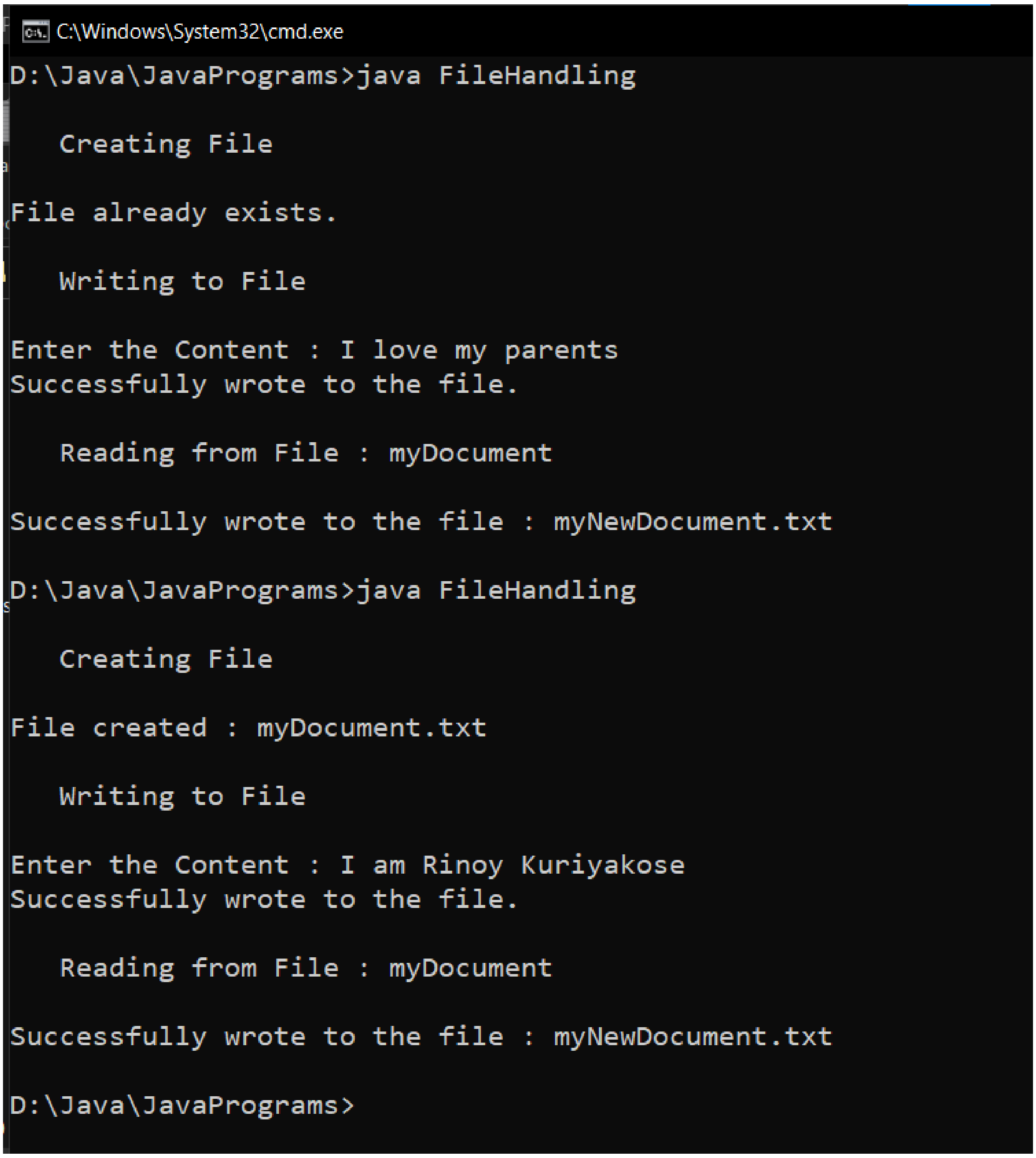
```
Algorithm FileHandling
  1. import java.io.*
  2. throws IOException
  3. Initialise File obj objects to create the  two files
  4. Initialise FileWriter objects on the  two files
  5. Read the input from User(using BufferedReader Class) and write to the file
  6. while((i = fr.read()) != -1) // Reading From the file1
  7.        write (char)i     // Writing to file2
  8. endwhile
```

Program:

```
import java.io.*;

public class FileHandling {
  public static void main(String[] args) throws IOException {
    String content;
    int i;
    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
    System.out.println("\n   Creating File \n");
    try {
      File myObj = new File("myDocument.txt");
```

```java
            if (myObj.createNewFile()) {
                System.out.println("File created : " + myObj.getName());
            } else {
                System.out.println("File already exists.");
            }
        } catch (IOException e1) {
            System.out.println("An error has occurred during File Creation.");
        }
        System.out.println("\n   Writing to File \n");
        try {
            FileWriter myWriter = new FileWriter("myDocument.txt");
            System.out.print("Enter the Content : ");
            content=br.readLine();
            myWriter.write(content);
            myWriter.close();
            System.out.println("Successfully wrote to the file.");
        } catch (IOException e2) {
            System.out.println("An error occurred during writing to File.");
        }
        System.out.println("\n   Reading from File : myDocument\n");
        File newObj = new File("myNewDocument.txt");
        newObj.createNewFile();
        FileWriter newWriter = new FileWriter("myNewDocument.txt");
        FileReader fr=null;
        try
        {
            fr = new FileReader("myDocument.txt");
            while ((i=fr.read())!=-1){
                newWriter.write((char)i);
            }
        }
        catch (FileNotFoundException e3)
        {
            System.out.println("File not found");
        }
        System.out.println("Successfully wrote to the file : myNewDocument.txt");
        newWriter.close();
        fr.close();
    }
}
```

Output:



```
D:\Java\JavaPrograms>java FileHandling

   Creating File

File already exists.

   Writing to File

Enter the Content : I love my parents
Successfully wrote to the file.

   Reading from File : myDocument

Successfully wrote to the file : myNewDocument.txt

D:\Java\JavaPrograms>java FileHandling

   Creating File

File created : myDocument.txt

   Writing to File

Enter the Content : I am Rinoy Kuriyakose
Successfully wrote to the file.

   Reading from File : myDocument

Successfully wrote to the file : myNewDocument.txt

D:\Java\JavaPrograms>
```

Result:

File handling is carried out in a Java program.

23/10/2020

Experiment: 11

Aim:

Write a Java program that reads a line of integers, and then displays each integer, and the sum of all the integers (Use String Tokenizer class of java.util)

Concepts Used:

BufferedReader class, StringTokenizer class.

Algorithm:

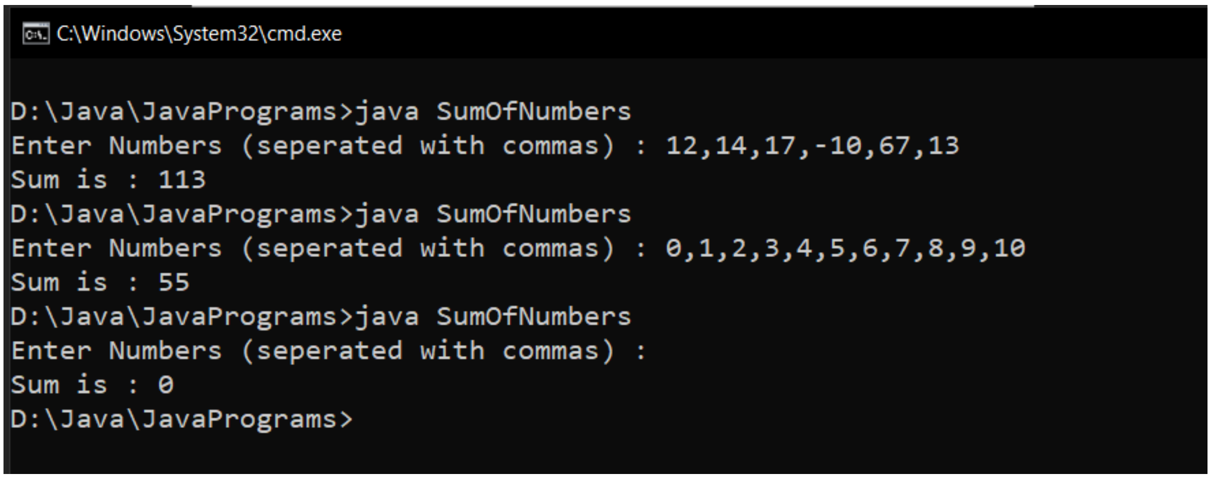
```
Algorithm StringTokens
   1. import java.util.* and import java.io.*
   2. sum = 0
   4. read the numbers separated by comma
   5. Initialise StringTokenizer object st on num
   6. while(st.hasMoreTokens())
   7.      num = Integer.parseInt(st.nextToken())
   8.      sum += num
   9. endwhile
  10. Print sum
```

Program:

```java
import java.util.StringTokenizer;
import java.io.*;
public class SumOfNumbers{
   public static void main(String args[]) throws IOException{
      String num;
      int sum =0;
      BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
      System.out.print("Enter Numbers (seperated with commas) : ");
      num=br.readLine();
      StringTokenizer st = new StringTokenizer(num,",");
```

```
        while (st.hasMoreTokens()) {
          sum=sum+Integer.parseInt(st.nextToken());
        }
        System.out.print("Sum is : "+sum);
    }
}
```

Output:



C:\Windows\System32\cmd.exe

```
D:\Java\JavaPrograms>java SumOfNumbers
Enter Numbers (seperated with commas) : 12,14,17,-10,67,13
Sum is : 113
D:\Java\JavaPrograms>java SumOfNumbers
Enter Numbers (seperated with commas) : 0,1,2,3,4,5,6,7,8,9,10
Sum is : 55
D:\Java\JavaPrograms>java SumOfNumbers
Enter Numbers (seperated with commas) :
Sum is : 0
D:\Java\JavaPrograms>
```

Result:

A line of integers are read from the user and they are printed along with their sum.

31/10/2020

Experiment: 12

Aim:

Write a Java program for the following:
A) Create a doubly linked list of elements.
B) Delete a given element from the above list.
C) Display the contents of the list after deletion.

Concept Used:

Collection Framework.

Algorithm:

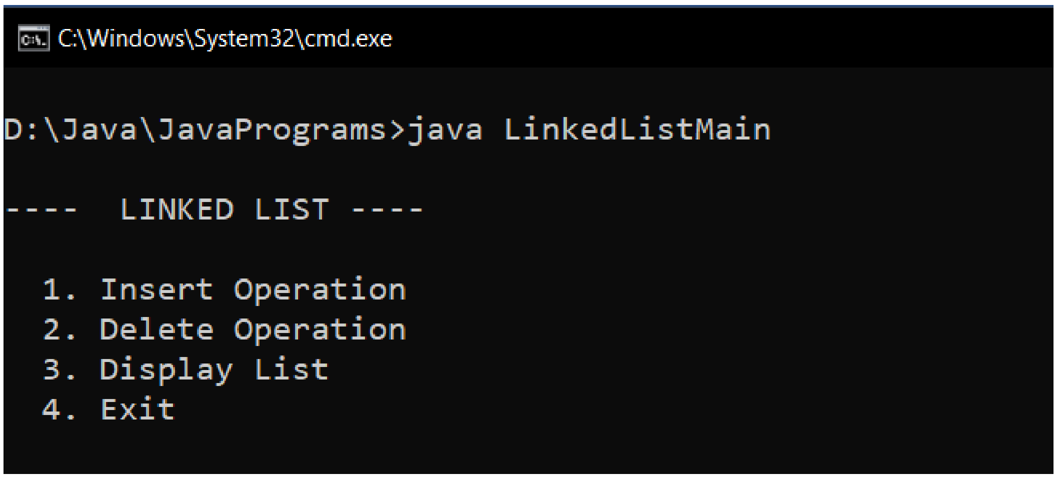
```
1. import java.util .*
2. Initialise LinkedList<String> object list
3. Add members using list.add() method
4. Print list
5. Remove members using list.remove() method
6. Initialise Iterator object itr = list.iterator()
7. while(itr.hasNext()) //Displaying the list
8.     Print itr.next()
9. endwhile
```

Program:

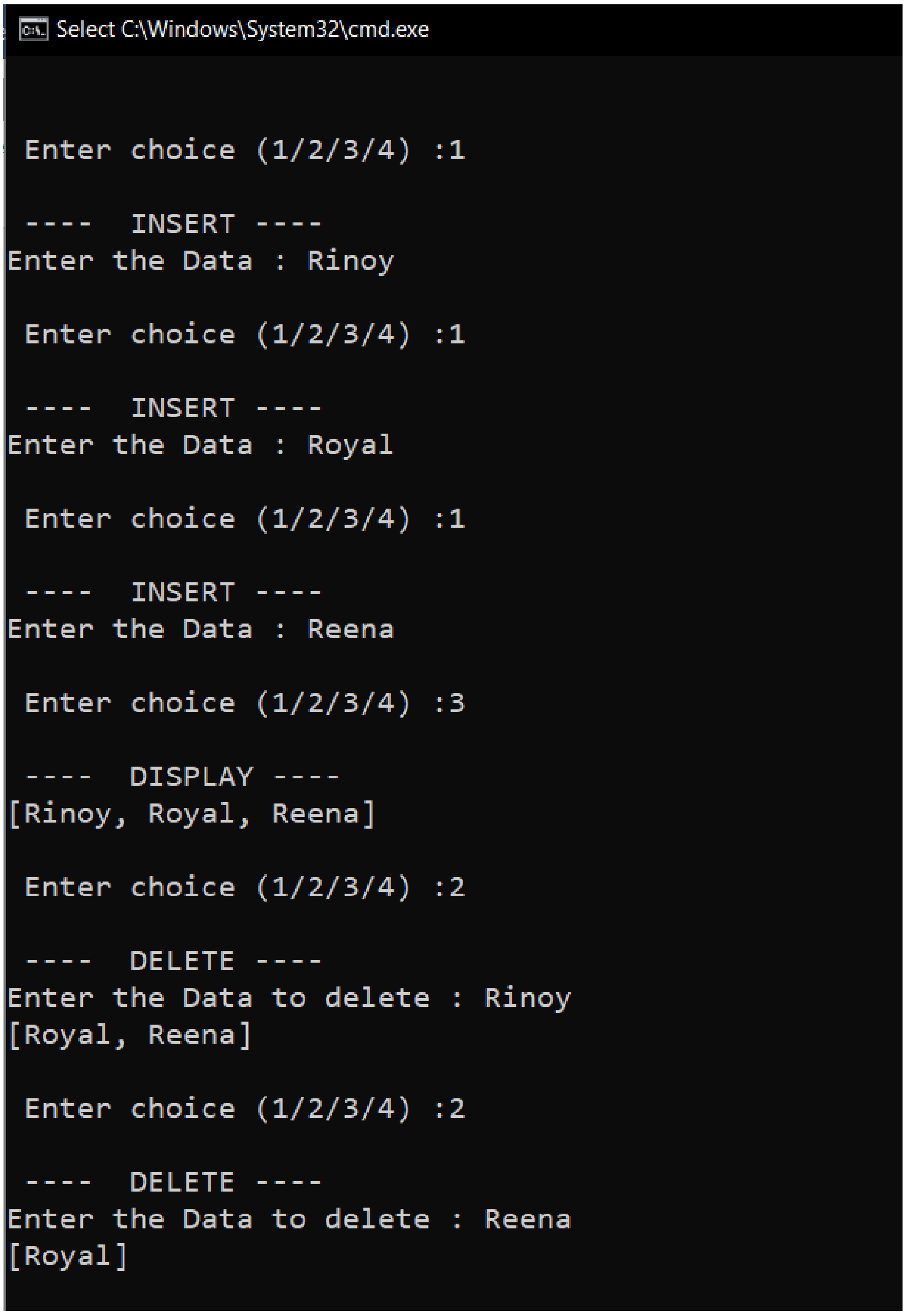
```
import java.util.*;
import java.io.*;

class LinkedListMain {
  public static void main(String[] args) throws IOException{
    LinkedList<String> data = new LinkedList<>();
    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
    int ans=1;
```

```
        System.out.println(“ \n----  LINKED LIST ---- \n”);
        System.out.println(“  1. Insert Operation “);
        System.out.println(“  2. Delete Operation “);
        System.out.println(“  3. Display List “);
        System.out.println(“  4. Exit \n”);
        while(ans!=4) {
          System.out.print(“\n Enter choice (1/2/3/4) :”);
          ans=Integer.parseInt(br.readLine());
          switch(ans){
            case 1 : System.out.println(“\n ----  INSERT ---- “);
                     System.out.print(“Enter the Data : “);
                     data.add(br.readLine());
                     break;
            case 2 : System.out.println(“\n ----  DELETE ----");
                     System.out.print(“Enter the Data to delete : “);
                     data.remove(br.readLine());
                     System.out.println(data);
                     break;
            case 3 : System.out.println(“\n ----  DISPLAY ----");
                     System.out.println(data);
                     break;
            case 4 : break;
            default: System.out.println(“Invalid Choice”);
          }

        }
      }
    }
```

Output:



```
C:\Windows\System32\cmd.exe

D:\Java\JavaPrograms>java LinkedListMain

----  LINKED LIST ----

  1. Insert Operation
  2. Delete Operation
  3. Display List
  4. Exit
```



Select C:\Windows\System32\cmd.exe

```
 Enter choice (1/2/3/4) :1

 ----  INSERT ----
Enter the Data : Rinoy

 Enter choice (1/2/3/4) :1

 ----  INSERT ----
Enter the Data : Royal

 Enter choice (1/2/3/4) :1

 ----  INSERT ----
Enter the Data : Reena

 Enter choice (1/2/3/4) :3

 ----  DISPLAY ----
[Rinoy, Royal, Reena]

 Enter choice (1/2/3/4) :2

 ----  DELETE ----
Enter the Data to delete : Rinoy
[Royal, Reena]

 Enter choice (1/2/3/4) :2

 ----  DELETE ----
Enter the Data to delete : Reena
[Royal]
```

Result:

A doubly linked list is implemented using Collections framework.

31/10/2020

Experiment: 13

Aim:

Write a Java program that implements the binary search algorithm.

Concept Used:

Scanner, Binary Search algorithm.

Algorithm:

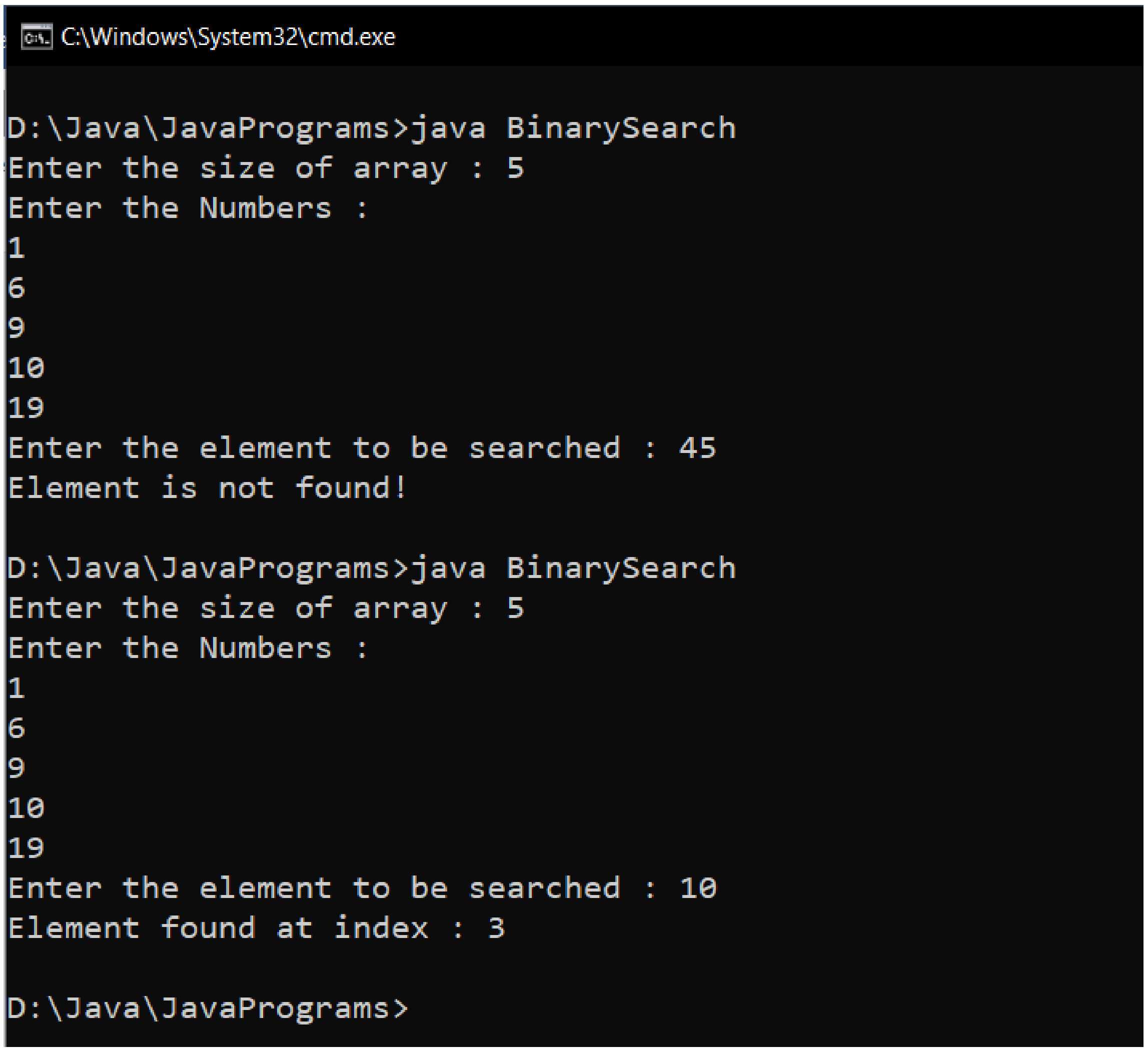
Algorithm BinarySearch

```
1. import java.io.*
2. Read array size,
3. Initialise array arr of size n
4. Read the elements
5. first = 0, last = n –1
6. while (first <= last)
7.     mid = (first + last) / 2
8.          if(search < arr[mid])
9.                 last = mid –1
10.         else if (search > arr[mid])
11.                first = mid + 1
12.        else
13.               Print “Element found”
14.      endif
15. endwhile
16. if(first>last)
17.     print “Element Not Found”
18. endIf
```

Program:

```
import java.io.*;
class BinarySearch{
   public static void main(String args[]) throws IOException {
      BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
      int[] arr = new int[50];
      int size,search;
      System.out.print("Enter the size of array : ");
      size=Integer.parseInt(br.readLine());
      System.out.println("Enter the Numbers : ");
      for(int i=0;i<size;i++){
         arr[i]=Integer.parseInt(br.readLine());
      }
      System.out.print("Enter the element to be searched : ");
      search = Integer.parseInt(br.readLine());
      int first=0;
      int last=size-1;
      int mid = (first + last)/2;
      while( first <= last ){
         if ( arr[mid] < search){
            first = mid + 1;
         }else if ( arr[mid] == search){
            System.out.println("Element found at index : " + mid);
            break;
         }else{
            last = mid-1;
         }
         mid =(first + last)/2;
      }
      if ( first > last ){
         System.out.println("Element is not found!");
      }
   }
}
```

Output:



```
D:\Java\JavaPrograms>java BinarySearch
Enter the size of array : 5
Enter the Numbers :
1
6
9
10
19
Enter the element to be searched : 45
Element is not found!

D:\Java\JavaPrograms>java BinarySearch
Enter the size of array : 5
Enter the Numbers :
1
6
9
10
19
Enter the element to be searched : 10
Element found at index : 3

D:\Java\JavaPrograms>
```

Result:

Binary Search is performed on an array of integers.

07/11/2020

Experiment: 14

Aim:

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

Concept Used:

Multithreading, Random generator.

Algorithm:

Class Thread1 extends Thread

```
1. import java.util.Random
2. Initialise Random object rand and static field random
3. Override public void run() method
4. num = rand.nextInt(100)
5. Print num
6. Thread2 obj2 = new Thread2(num)
7. Thread3 obj3 = new Thread3(num)
8. if(random%2==0)
9.        obj2.start()
10. else
11.        obj3.start()
12.endif
```

Class Thread2 extends Thread

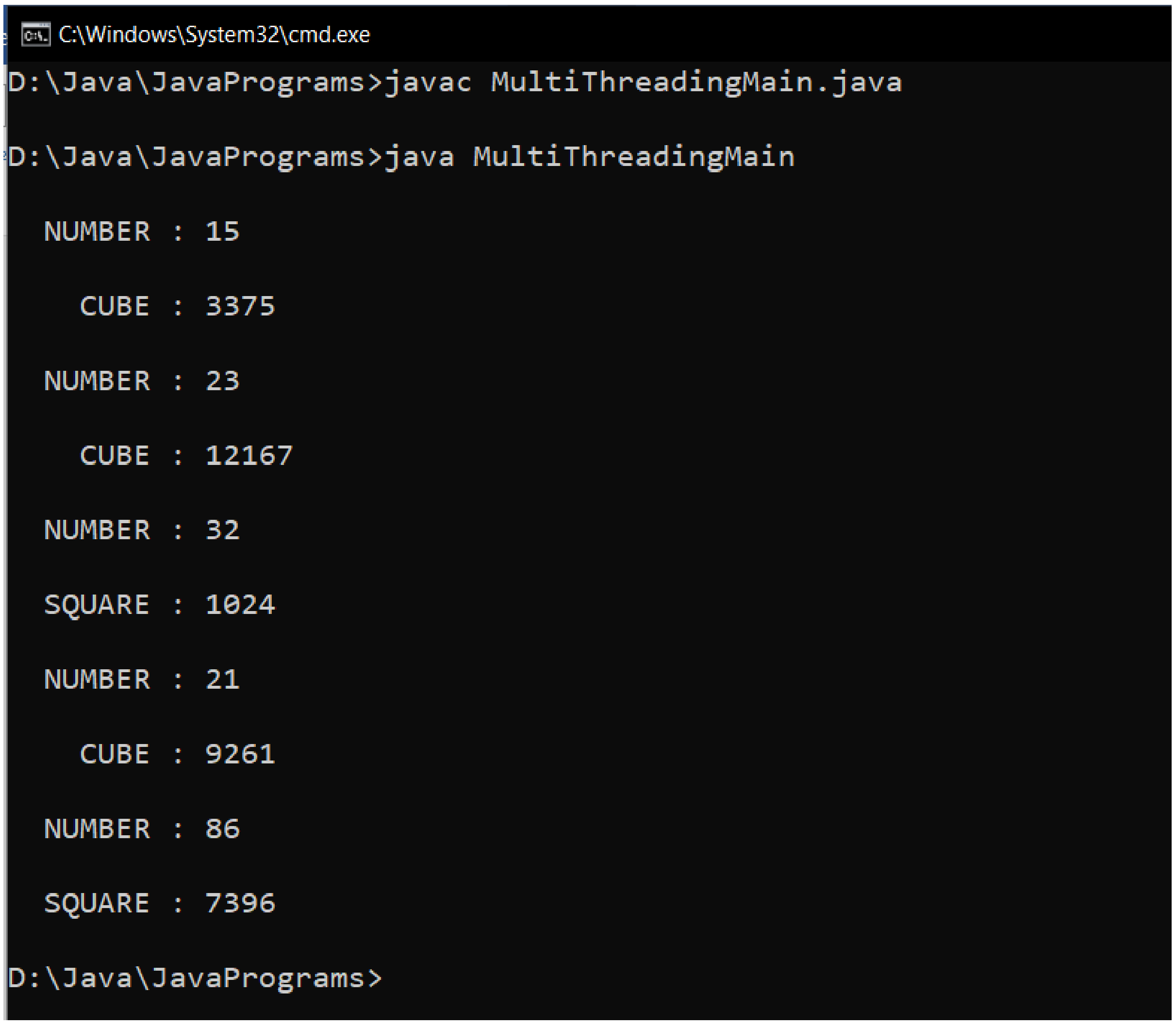
```
1. Override public void run() method
2. Print Math.pow(num, 2)
```

```
   3. End run()

Class Thread3 extends Thread

   1. Override public void run() method

   2. Print Math.pow(num, 3)

   3. End run()

Class ThreadMain

   1. for i=0 till 5

   2.     Thread1 obj1 = new Thread1()

   3.     obj1.start()

   4.     obj1.sleep(1000)

   5. endfor
```

Program:

```java
import java.util.*;
class Thread1 extends Thread

{   int num;
    public void run(){
            Random obj = new Random();
            num=obj.nextInt(100);
            System.out.println("\n  NUMBER : "+num);
            Thread2 obj2 = new Thread2(num);
            Thread3 obj3 = new Thread3(num);
            if(num%2==0){
                obj2.start();
            }else {
                obj3.start();
            }
    }
}
class Thread2 extends Thread{
    int num;
    public void run(){
        System.out.println("\n  SQUARE : "+num*num);
    }
    Thread2(int num) {
        this.num=num;
    }
}
class Thread3 extends Thread
{
    int num;
    public void run(){
        System.out.println("\n    CUBE : "+num*num*num);
    }
    Thread3(int num){
        this.num=num;
    }
}
public class MultiThreadingMain {
    public static void main(String[] args) throws InterruptedException{
        for(int i=0;i<5;i++){
            Thread1 obj1 = new Thread1();
            obj1.start();
            obj1.sleep(1000);
        }
    }
}
```

Output:



C:\Windows\System32\cmd.exe

```
D:\Java\JavaPrograms>javac MultiThreadingMain.java

D:\Java\JavaPrograms>java MultiThreadingMain

  NUMBER : 15

    CUBE : 3375

  NUMBER : 23

    CUBE : 12167

  NUMBER : 32

  SQUARE : 1024

  NUMBER : 21

    CUBE : 9261

  NUMBER : 86

  SQUARE : 7396

D:\Java\JavaPrograms>
```

Result:

Multi-threading is implemented in a Java program.

07/11/2020

Experiment: 15

Aim:

Write a Java program that shows thread synchronization

Concept Used:

Multithreading, Thread synchronization.

Algorithm:

Class Number

```
1. synchronized print(n, )
2. for i = 1 to 5
3.     Print n
4. endfor
5. End print()
```

class Thread1 extends Thread

```
1. Declare field num (object of class Number) and define a constructor to initialise num
2. public void run()
3.     obj.print(1)
4. End run()
```

class Thread1 extends Thread

```
1. Declare field num (object of class Number) and define a constructor to initialise num
2. public void run()
3.     obj.print(2)
4. End run()
```

class ThreadSynchronization

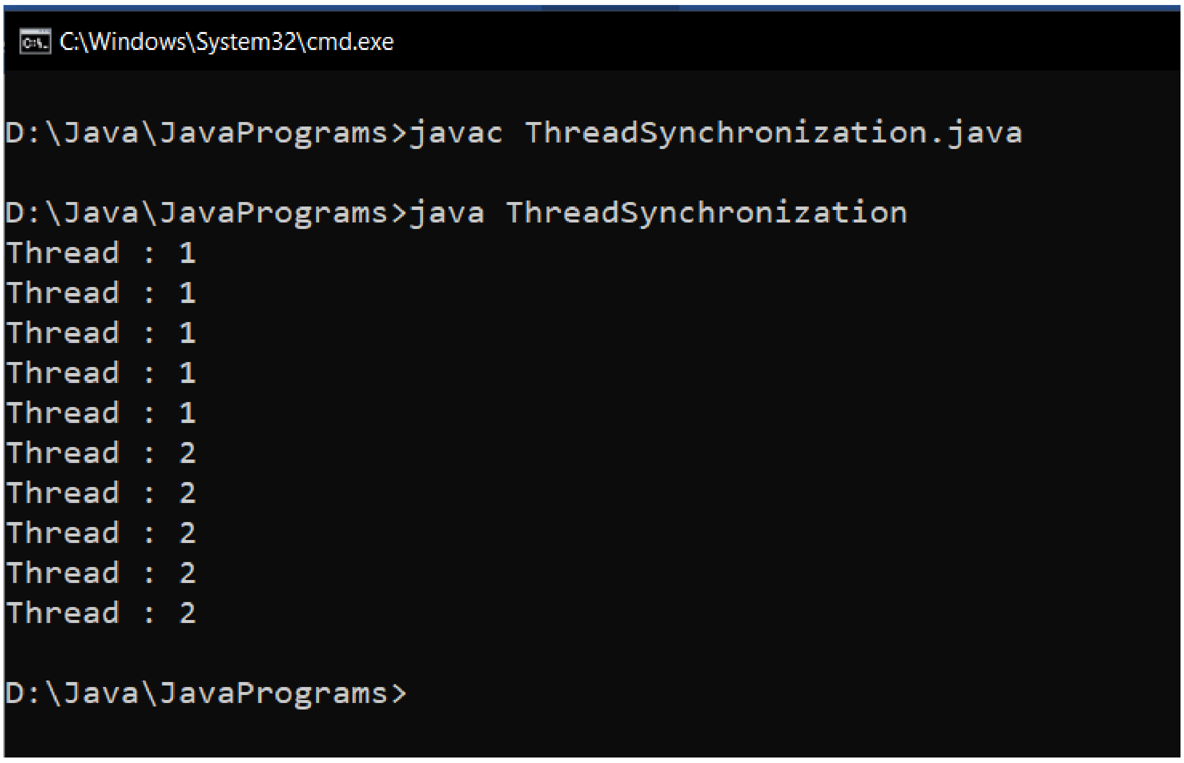
1. Initialise Number obj
2. Thread1 obj1 = new Thread1(obj)
3. Thread2 obj2 = new Thread2(obj);
4. obj1.start();
5. obj2.start();

Program:

```
class Number{
   synchronized void print(int n){
      for(int i=1;i<=5;i++){
         System.out.println("Thread : "+n);
      }
   }
}
class Thread1 extends Thread{
   Number num;
   Thread1(Number num){
      this.num=num;
   }
   public void run(){
      num.print(1);
   }
}
class Thread2 extends Thread{
   Number num;
   Thread2(Number num){
      this.num=num;
   }
   public void run(){
      num.print(2);
   }
}

public class ThreadSynchronization{
   public static void main(String args[]){
      Number obj = new Number();
```

```
        Thread1 obj1 = new Thread1(obj);
        Thread2 obj2 = new Thread2(obj);
        obj1.start();
        obj2.start();
    }
}
```

Output:



C:\Windows\System32\cmd.exe

```
D:\Java\JavaPrograms>javac ThreadSynchronization.java

D:\Java\JavaPrograms>java ThreadSynchronization
Thread : 1
Thread : 1
Thread : 1
Thread : 1
Thread : 1
Thread : 2
Thread : 2
Thread : 2
Thread : 2
Thread : 2

D:\Java\JavaPrograms>
```

Result:

Thread synchronization is carried out in a Java program.

22/11/2020

Experiment: 16

Aim:

Write a Java program that works as a simple calculator. Arrange Buttons for digits and the + - * % operations properly. Add a text field to display the result. Handle any possible exceptions like divide by zero. Use Java Swing

Concept Used:

Java Swing –Graphical User Interface and Event handling.

Algorithm:

```
Algorithm Calculator
   1. Import java.awt, java.awt.event, java.swing and java.util classes
   2. Initialise JFrame frame with title “ Calculator”
   3. Initialise JPanel panel1, panel2
   4. Set layout managers GridLayout(4,5), BorderLayout() for  panel2 and frame respectively
   5. frame.setSize(350,350)
   6. frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
//Components
   7. JTextField input = new JTextField()
   8. JButton b1 = new JButton("1")
   9. JButton b2 = new JButton("2")
  10. JButton b3 = new JButton("3")
  11. JButton b4 = new JButton("4")
  12. JButton b5 = new JButton("5")
  13. JButton b6 = new JButton("6")
  14. JButton b7 = newJButton("7")
  15. JButton b8 = new JButton("8")
  16. JButton b0 = new JButton("0"
  17. JButton b9 = new JButton("9")

  18. JButton b17 = new JButton("AC")

  19. JButton b19 = new JButton(".")

  20. JButton b10 = new JButton("+")

  21. JButton b11 = new JButton("-")

  22. JButton b13 = new JButton("*")

  23. JButton b12 = new JButton("/")

  24. JButton b14 = new JButton("%")

  25. JButton b16 = new JButton("=")

  26. . JButton b15 = new JButton("^")

  27. Add buttons to panel2

  28. input.setHorizontalAlignment(JTextField.CENTER)

  29. panel1.add(input)

  30. frame.add(panel1,BorderLayout.NORTH)

  31. frame.add(panel1,BorderLayout.CENTER)

  32. frame.setVisible(true)

//Event Handling
//For the input buttons, the event handler is,

  33. input.setText(result.getText() + "0") //Button 0

// Same for all other from 1-9

//For AC button, the event handler is,

  34. input.setText("")

//For DEL button, the event handler is,

  35. input.setText(input.setText(input.getText().substring(0, input.getText().length() - 1)

//For Equals button, the event handler is,

  36. inp = input.getText()

  37. i = 0 flag=1
```

```
38. result = 0
39. op = exp.charAt(0)
40. try
41. while(z != '+' && z != '-' && z != '*' && z != '/' && z != '%')
42.        op = exp.charAt(i)
43.        i++
44. endwhile
45. x = Float.parseFloat(exp.substring(0,i-1))
46. y = Float.parseFloat(exp.substring(i,inp.length()))
47. switch(op)
48.     case + : result = x+y
49.               break
50.     case - :  result = x-y
51.               break
52.     case * : result = x*y
53.               break
54.     case / : if(y == 0)
55.                   flag=0   break
56.               result = x/y
57.               break
58.     case %:result = x%y
59.              break
60.     case ^:result = Math.pow(x,y)
61.              break
62. endcase
63. endswitch
64. if(flag==0)
65.       input.setText("Not defined!")
```

```
66. else
67.       input.setText(res+"")
68. endif
69. catch
70. input.setText("Invalid Input")
71. endtry
```

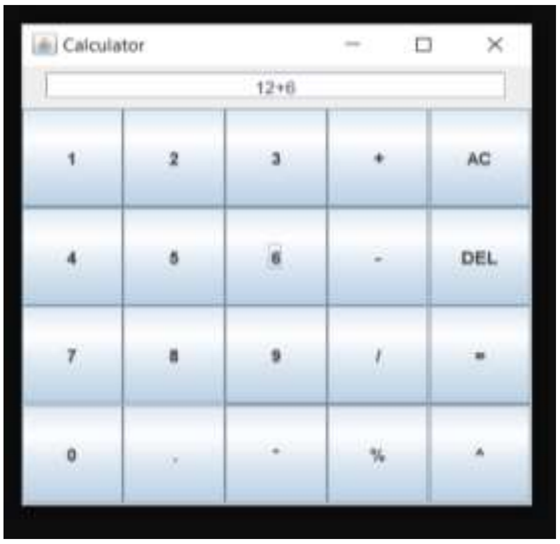
Program:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class CalculatorSwing {
    CalculatorSwing(){
        JFrame frame=new JFrame("Calculator");
        JPanel panel1=new JPanel();
        JPanel panel2=new JPanel();

        frame.setLayout(new BorderLayout());
        panel2.setLayout(new GridLayout(4,5));
        frame.setSize(350,350);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JButton b0=new JButton("0");
        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("+");
        JButton b11=new JButton("-");
        JButton b12=new JButton("/");
        JButton b13=new JButton("*");
        JButton b14=new JButton("%");
        JButton b15=new JButton("^");
JButton b16=new JButton("=");
JButton b17=new JButton("AC");
JButton b18=new JButton("DEL");
JButton b19=new JButton(".");

JTextField input = new JTextField(30);

input.setHorizontalAlignment(JTextField.CENTER);
panel1.add(input);
panel2.add(b1);
panel2.add(b2);
panel2.add(b3);
panel2.add(b10);
panel2.add(b17);

panel2.add(b4);
panel2.add(b5);
panel2.add(b6);
panel2.add(b11);
panel2.add(b18);

panel2.add(b7);
panel2.add(b8);
panel2.add(b9);
panel2.add(b12);
panel2.add(b16);

panel2.add(b0);
panel2.add(b19);
panel2.add(b13);
panel2.add(b14);
panel2.add(b15);
frame.add(panel1,BorderLayout.NORTH);
frame.add(panel2,BorderLayout.CENTER);

frame.setVisible(true);
b0.addActionListener(new ActionListener(){
   public void actionPerformed(ActionEvent ae){
      if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
         input.setText("");
      }
```

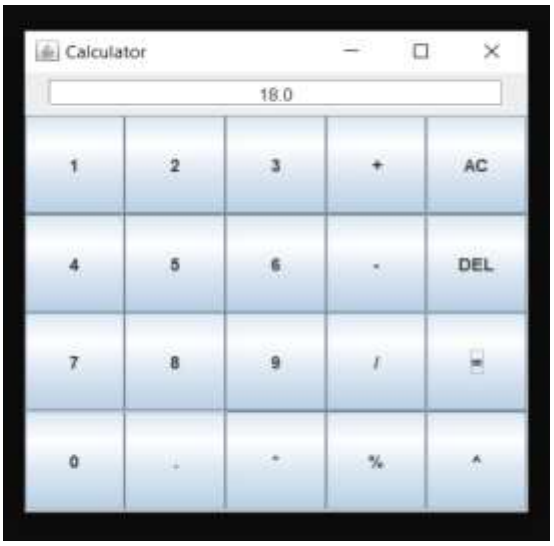
```
            input.setText(input.getText()+"0");
        }});
    b1.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
            if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
                input.setText("");
            }
            input.setText(input.getText()+"1");
        }});
    b2.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
            if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
                input.setText("");
            }
            input.setText(input.getText()+"2");
        }});
    b3.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
            if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
                input.setText("");
            }
            input.setText(input.getText()+"3");
        }});
    b4.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
            if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
                input.setText("");
            }
            input.setText(input.getText()+"4");
        }});
    b5.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
            if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
                input.setText("");
            }
            input.setText(input.getText()+"5");
        }});
    b6.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
            if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
                input.setText("");
        }
        input.setText(input.getText()+"6");
    }});
  b7.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
            input.setText("");
        }
        input.setText(input.getText()+"7");
    }});
  b8.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
            input.setText("");
        }
        input.setText(input.getText()+"8");
    }});
  b9.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
            input.setText("");
        }
        input.setText(input.getText()+"9");
    }});
  b10.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
            input.setText("");
        }
        input.setText(input.getText()+"+");
    }});
  b11.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
            input.setText("");
        }
        input.setText(input.getText()+"-");
    }});
  b12.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
            input.setText("");
          }
          input.setText(input.getText()+"/");
        }});
      b13.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
          if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
            input.setText("");
          }
          input.setText(input.getText()+"*");
        }});
      b14.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
          if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
            input.setText("");
          }
          input.setText(input.getText()+"%");
        }});
      b15.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
          if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
            input.setText("");
          }
          input.setText(input.getText()+"^");
        }});
      b16.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
          String inp = input.getText();
          int i = 0;
          float a, b, result = 0;
          char op = inp.charAt(0);
          try
          {
            while(op != '+' && op != '-' && op!= '*' && op != '/' && op != '%' && op != '^')
            {
              op = inp.charAt(i);
              i++;
            }
            a = Float.parseFloat(inp.substring(0,i-1));
            b = Float.parseFloat(inp.substring(i,inp.length()));
            switch(op)
      {
        case '+': result = a+b;
             break;
        case '-': result = a-b;
             break;
        case '/': try{
               result = a/b;
             }catch(Exception e1){}
             break;
        case '*': result = a*b;
             break;

        case '%': result = a%b;
             break;
        case '^': result = (float) Math.pow(a,b);
             break;
      }
      if(b!=0){
        input.setText(result+"");
      }else{
        input.setText("Not Defined");
      }
    }catch(Exception e2){
      input.setText("Invalid Input");
    }
  }});
b17.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent ae){
    input.setText("");
  }});
b18.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent ae){
    if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
      input.setText("");
    }else {
      try {
        input.setText(input.getText().substring(0, input.getText().length() - 1));
      } catch (Exception e3) {
      }
    }
  }});
```

```
        b19.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent ae){
            if(input.getText().equals("Invalid Input")||input.getText().equals("Not Defined")){
              input.setText("");
            }
            input.setText(input.getText()+".");
          }});
      }

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

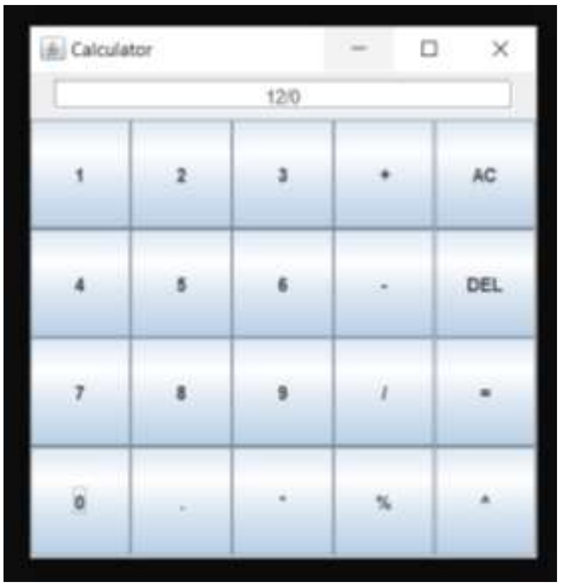
Output:



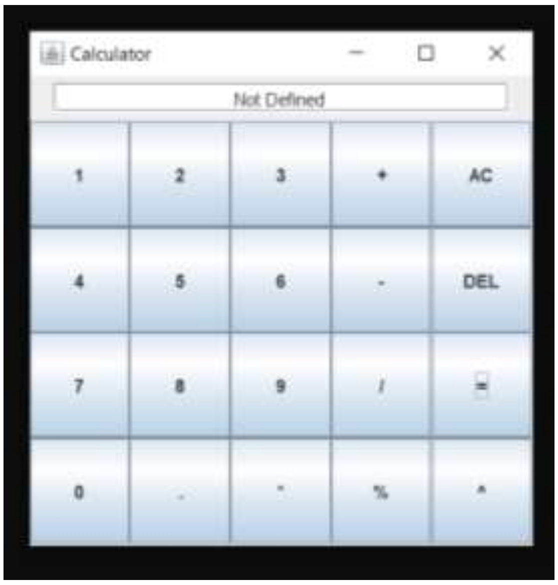
```
12+6
```



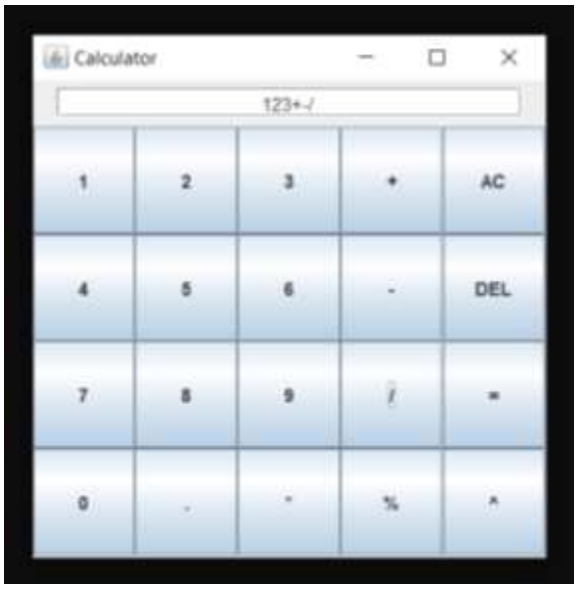
```
18.0
```

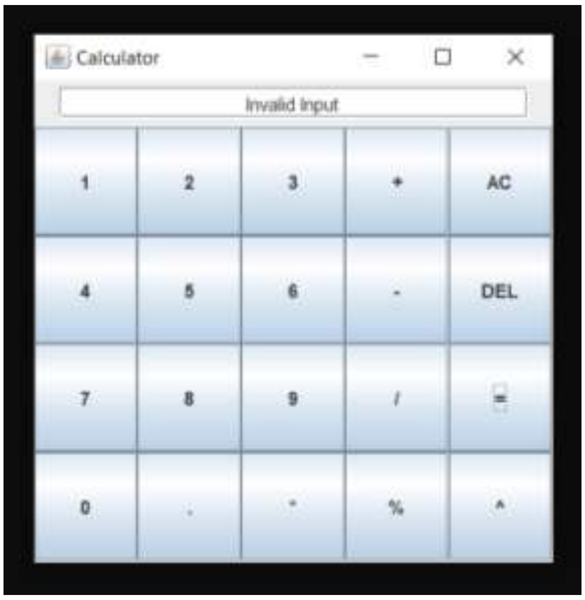


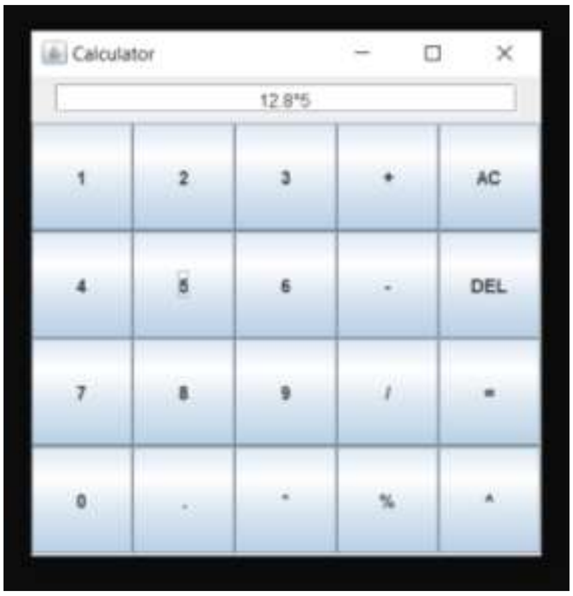
```
12/0
```

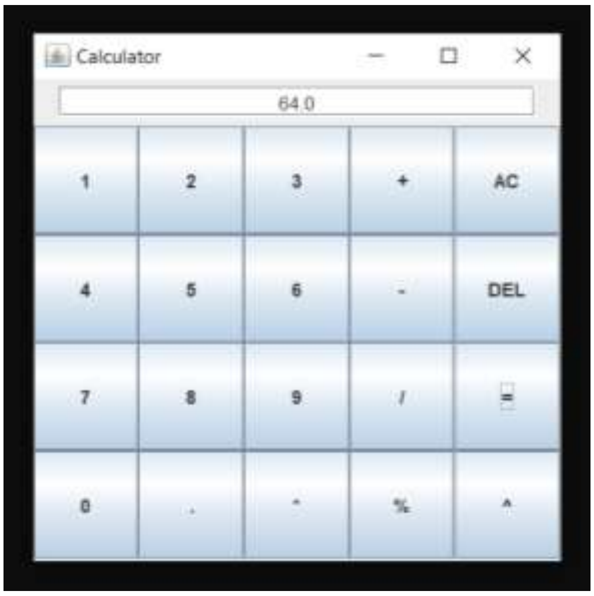


```
Not Defined
```









Result:

A simple calculator with GUI is created using Java Swing.

22/11/2020

Experiment: 17

Aim:

Write a Java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green. When a radio button is selected, the light is turned on, and only one light can be on at a time. No light is on when the program starts.

Concept Used:

Java Swing – GUI and Event Handling.

Algorithm:

```
 1.Import java.awt, java.awt.event and java.swing classes
 2. JFrame frame = new JFrame("Traffic Light")
 3. JRadioButton red = new JradioButton
 4. JRadioButton yellow = new JradioButton
 5. JRadioButton green = new JradioButton
 6. ButtonGroup traffic = new ButtonGroup()
 7. red.setBackground(Color.WHITE)
 8. yellow.setBackground(Color.WHITE)
 9. green.setBackground(Color.WHITE)
10. traffic.add(red)
11. traffic.add(yellow)
12.traffic.add(green)
13. frame.add(red)
14. frame.add(yellow)
15.frame.add(green)
16. frame.setSize(300,100)
17. frame.setLayout(new GridLayout(1,3))
```

18. frame.setVisible(true)

19. frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)

//Event handler code for red button

20. red.setBackground(Color.RED)

21. yellow.setBackground(Color.WHITE)

22. green.setBackground(Color.WHITE)

23. red.setText("STOP")

24. yellow.setText("")

25. green.setText("")

//Event handler code for yellow button

26. red.setBackground(Color.WHITE)

27. yellow.setBackground(Color.YELLOW)

28. green.setBackground(Color.WHITE)

29. red.setText("")

30. yellow.setText("WAIT")

31. green.setText("")

//Event handler code for RED button

32. red.setBackground(Color.WHITE)

33. yellow.setBackground(Color.WHITE)

34. green.setBackground(Color.GREEN)

35. red.setText("")

36. yellow.setText("")

37. green.setText("GO")

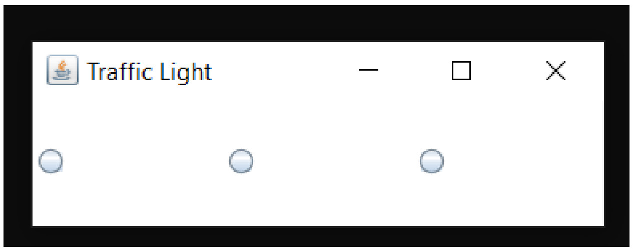
Program:

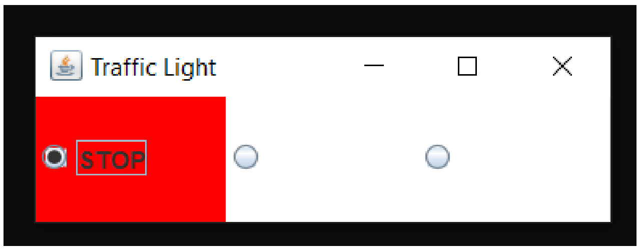
```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

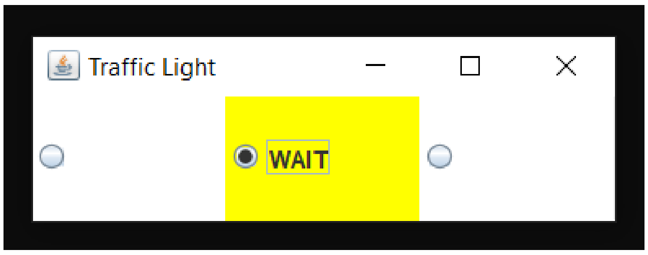
```
public class TrafficLight
{
   TrafficLight() {
      JFrame frame = new JFrame("Traffic Light");
      JRadioButton red = new JRadioButton();
      JRadioButton yellow = new JRadioButton();
      JRadioButton green = new JRadioButton();
      ButtonGroup traffic = new ButtonGroup();
      traffic.add(red);
      traffic.add(yellow);
      traffic.add(green);
      red.setBackground(Color.WHITE);
      yellow.setBackground(Color.WHITE);
      green.setBackground(Color.WHITE);
      frame.add(red);
      frame.add(yellow);
      frame.add(green);
      frame.setSize(300,100);
      frame.setLayout(new GridLayout(1,3));
      frame.setVisible(true);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      red.addActionListener(new ActionListener(){
         public void actionPerformed(ActionEvent ae){
            red.setBackground(Color.RED);
            yellow.setBackground(Color.WHITE);
            green.setBackground(Color.WHITE);
            red.setText("STOP");
            yellow.setText("");
            green.setText("");
         }
      });
      yellow.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent ae){
            red.setBackground(Color.WHITE);
            yellow.setBackground(Color.YELLOW);
            green.setBackground(Color.WHITE);
            red.setText("");
            yellow.setText("WAIT");
            green.setText("");
         }
      });
```

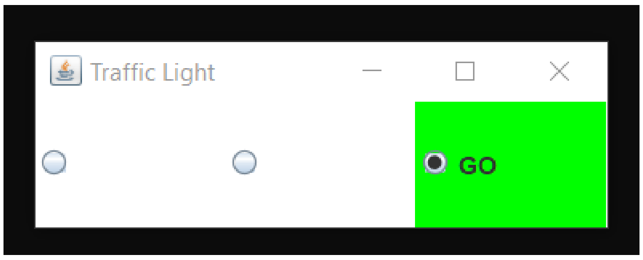
```
        green.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent ae){
            red.setBackground(Color.WHITE);
            yellow.setBackground(Color.WHITE);
            green.setBackground(Color.GREEN);
            red.setText("");
            yellow.setText("");
            green.setText("GO");
          }
        });
    }
    public static void main(String[] args)
    {
        new TrafficLight();
    }
}
```

Output:









Result:

A traffic light is stimulated using Java Swing.

15/01/2021

Experiment: 18

Aim:

Write a Java program that implements a Quick sort algorithm for sorting a list of names in ascending order.

Concept Used:

Quick Sort algorithm

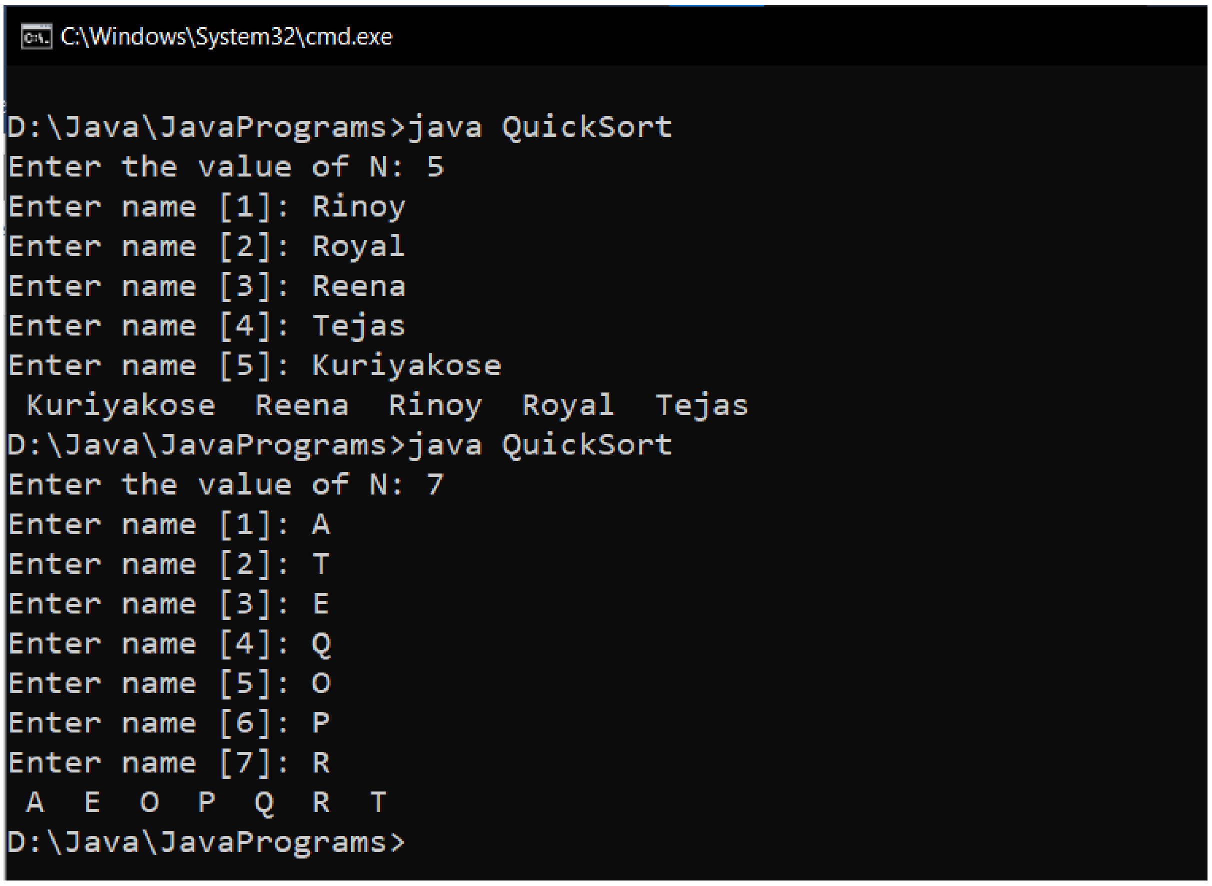
Algorithm:

```
Algorithm Partition(A, p, r)
 START
  1. x = A[r]
  2. i = p-1
  3. for j = p to r
  4.    if (A[j] <= x)
  5.         i = i+1
  6.         if (i != j)
  7.              swap A[i] and A[j]
  8.         endif
  9.    endif
10. endfor
11. if (r != i+1)
12.    swap A[i+1] and A[r]
13. endif
14. return i+1
STOP


Algorithm QuickSort(A, p, r)
START
  1. if (p < r)
  2.    q = Partition(A, p ,r)
  3.    QuickSort(A, p, q-1)
  4.    QuickSort(A, q+1, r)
  5. endif
STOP
```

Program:

```
import java.util.*;
public class QuickSort {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the value of N: ");
        int n = sc.nextInt();
        String arr[] = new String[n];
        sc.nextLine();
        for(int i=0; i<n; i++)
        {
            System.out.print("Enter name [" + (i+1) +"]: ");
            arr[i] = sc.nextLine();
        }
        QuickSort obj = new QuickSort();
        obj.sort(arr, 0, arr.length - 1);
        for (String i : arr) {
            System.out.print(" "+ i + " ");
        }
    }
    int partition(String [] arr, int p, int r) {
        String pivot = arr[r];
        int i = (p - 1);
        String tmp;
        for (int j = p ; j < r ; j++) {
            if (arr[j].compareTo(pivot) < 0) {
                i = i + 1;
                tmp = arr[i];
                arr[i] = arr[j];
                arr[j] = tmp;
            }
        }
        tmp = arr[i+1];
        arr[i+1] = arr[r];
        arr[r] = tmp;
        return i + 1;
    }
    void sort(String[] arr, int p, int r) {
        if (p < r) {
            int q = partition(arr, p, r);
            sort(arr, p, q - 1);
            sort(arr, q + 1, r);
        }
    }
}
```

Output:



```
C:\Windows\System32\cmd.exe

D:\Java\JavaPrograms>java QuickSort
Enter the value of N: 5
Enter name [1]: Rinoy
Enter name [2]: Royal
Enter name [3]: Reena
Enter name [4]: Tejas
Enter name [5]: Kuriyakose
 Kuriyakose  Reena  Rinoy  Royal  Tejas
D:\Java\JavaPrograms>java QuickSort
Enter the value of N: 7
Enter name [1]: A
Enter name [2]: T
Enter name [3]: E
Enter name [4]: Q
Enter name [5]: O
Enter name [6]: P
Enter name [7]: R
 A  E  O  P  Q  R  T
D:\Java\JavaPrograms>
```

Result:

Quick Sort is performed on a list of names.