Experiment 1

Second smallest element in the array

Date Of Submission: 26-08-2020

Aim: Write a Java program to find the second smallest element in an array

Concepts Used: Class, Arrays

Algorithm:

1. Step 1: Start
2. Step 2: pos = 0 //Position of the smallest element
3. Step 3: smallest = arr[pos] //Assume the first element is the smallest
4. Step 4: for i from 1 to arraySize-1 do // Find the smallest element
5. Step 1: if arr[pos] > arr[i] then
6. Step 1: pos = i
7. Step 2: endif
8. Step 5: endFor
9. Step 6: if pos == 0 then
10. Step 1: secondSmallest = 1 // Assume that the second smallest number is the second number
11. Step 7: else
12. Step 1: secondSmallest =0;
13. Step 8: endif
14. Step 10: for i from 0 to arraySize-1 do
15. Step 1: if i == pos then
16. Step 1: continue
17. Step 2: endif
18. Step 3: if arr[i]<arr[secondSmallest] then
19. Step 1: secondSmallest = i
20. Step 4: endif
21. Step 11; Endfor
22. step 12: Stop

Result: The program is successfully compiled and the required output is obtained.

Program Code:

```
/*  Java program to find the second smallest element in an array
 *
*/
class Program1{
```

```
    public static void main(String args[]){
        int[] arr = {-34,2,9,34,12,9,-23,1,4,9,0};
        int arrSize=11;
        int smallest,secondSmallest;
        int i,pos=0;

        smallest=arr[0];

        for(i=1;i<11;i++){
            if(arr[i]<smallest){
                pos=i;
            }
        }

        if(pos!=0){
            secondSmallest=arr[0];
        }
        else{
            secondSmallest=arr[arrSize-1];
        }

        for(i=0;i<11;i++){
            if(i==pos)continue;
                if(arr[i]<secondSmallest){
                    secondSmallest=arr[i];
            }
        }

        System.out.println("Second Smallest element is "+secondSmallest);
    }
}
```

Sample Input
-34,2,9,34,12,9,-23,1,4,9,0

Sample Output:
Second Smallest element is -23

Experiment 2

Program to check whether the given number is prime

Date Of Submission: 26-08-2020

Aim: Write a Java program to check whether the given number is prime or not

Concepts Used: Class

Algorithm:

1. Step 1: Start
2. Step 2: read n // the number to be checked
3. Step 3: flag = 0
4. Step 4; for i from 2 to n/2 do
5. Step 1: if n%i == 0 then
6. Step 1: flag = 1
7. Step 2: break
8. Step 2: endif
9. Step 5: endFor
10. Step 6: if flag == 1 then
11. Step 1: print “The number is prime”
12. Step 7: else:
13. Step 1: print “The number is not prime”
14. Step 8: endif
15. Step 9: Stop

Result: The program is successfully compiled and the required output is obtained.

Program Code:

```
/*
 *Program to check whether a given number is prime or not
 *
 */

class Program2{
    public static void main(String[] args){
        int n,i;
        boolean flag=true;

        n=31; //The number to be checked

        for(i=2;i<n/2;i++){
```

```
            if(n%i==0){
                flag=false;
            }

        }

        if(flag)
            System.out.println("The number "+n+" is prime");
        else
            System.out.println("The number "+n+" is not prime");
    }
}
```

Sample input

31

Sample output:

The number 31 is prime

Experiment 3

Multiplication of Matrices

Date Of Submission: 26-08-2020

Aim: Write a Java program to multiply two given matrices

Concepts Used: 2-D Array, Class

Algorithm:

1. Step 1: Start
2. Step 2: if A.colums == B.rows then
3. Step 1: C.colums = B.columns
4. Step 2: C.rows = A.rows
5. Step 3: for i from 0 to A.rows-1 do
6. Step 1: for j from 0 to B.columns-1 do
7. Step 1: C.array[i][j] = 0
8. Step 2: for k from 0 to B.rows do
9. Step 1: C[i][j] += A[i][k]*B[k][j]
10. Step 3: endfor
11. Step 2: endFor
12. Step 4: endfor
13. Step 3:else
14. Step 1: Print “Matrices can’t be multiplied”
15. Step 4: endif
16. Step 5: Stop

Result: The program is successfully compiled and the required output is obtained.

Program Code

```
/*  Program to multiply two given matrices
 */

class Program3{
    public static void main(String[] args){
        int[][] C;
        int a_rows,a_columns,b_rows,b_columns,c_rows,c_columns;
        int i,j,k;

        a_rows=3;
        a_columns=2;
        int A[][] = {{1,2},{4,5},{9,16}};

        b_rows = 2; b_columns=4;
```

```
        int B[][] = {{3,4,5,6},{4,3,1,0}};

        if(a_columns==b_rows){
            c_rows = a_rows;
            c_columns = b_columns;
            C = new int[a_rows][b_columns];

            for(i=0;i<a_rows;i++){
                for(j=0;j<b_columns;j++){
                    C[i][j]=0;
                    for(k=0;k<a_columns;k++){
                        C[i][j]+=(A[i][k]*B[k][j]);
                    }
                }
            }

            System.out.println("Solution Matrix is : ");
            for(i=0;i<a_rows;i++){
                for(j=0;j<b_columns;j++){
                    System.out.print(C[i][j]+" ");
                }
                System.out.println(" ");
            }

        }

        else{
            System.out.println("Matrix cant be multiplied");
        }
    }
}
```

Sample input:

```
Matrix A = 1,2
           4,5
           9,16
Matrix B =  3,4,5,6
            4,3,1,0
```

Sample output:

```
Solution Marix is:
11 10 7 6
32 31 25 24
91 84 61 54
```

Experiment 4

Class To Represent Bank Account

Date Of Submission: 17-09-2020

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: Class, Constructor Overloading, Polymorphism

Algorithms :

Algorithm : depositAmount(x):

Step 1: Start
Step 2: balance += x
Step 3: Stop

Algorithm :withdrawAmount(x):

Step 1: Start
Step 2: balance -= x
Step 3: Stop

Algorithm : display()

Step 1: Start
Step 2: print name
Step 3: print accountNumber
Step 4: print account type
Step 5: print balance
Step 6: Stop

Algorithm Bank() //Default Constructor

Step 1: start
Step 2: name = “”
Step 3: accountNumber =0
Step 4: accountType = “”
Step 5: balance=0
Step 6: Stop

Algorithm Bank(n,num,type,bal)

Step 1: Start
Step 2: name = n
Step 3: accountNumber = num
Step 4: accountType = type
Step 5: balance=bal
Step 6: Stop

Result: The program was successfully compiled and the required output was obtained

Program:

```
/**
  BankAccount.java
  Created By: Rohit Karunakaran

 **/

class BankAccount{
    String name;
    long accNumber;
    String accType;
    float balance;

    public void withdraw(float amount){
        //You cannot withdraw negative values
        if(amount>=0){

            if(balance-amount > 0){
                balance-=amount;
                System.out.println("Sucessfully withdrawn "+ amount+" from the
account");
            }
            else{
                System.out.println("Error! Insufficient balance in Account");
            }
            System.out.println("Remaining balance in Account = " + balance+"\n");
        }

        else{
            System.out.println("Please enter a valid amount\n");
        }
    }

    public void deposit(float amount){
        //You can't deposit negative values
        if(amount>0){
            balance+=amount;
            System.out.println("Successfully deposited "+amount+" to the account");
            System.out.println("Remaining balance is "+ balance+"\n");
        }
        else{
            System.out.println("Error!!!! Enter a valid value\n");
        }
    }
```

```
public void display(){
    System.out.println("\n=================================================");
    System.out.println("Account Number: " + accNumber);
    System.out.println("Name of the Account Holder: "+ name);
    System.out.println("Account Type: "+ accType);
    System.out.println("Balance: "+ balance);
    System.out.println("=================================================\n");
}
//Default constructor
BankAccount(){
    name = " ";
    accNumber = 0l;
    accType = " ";
    balance = 0.0f;
}

//Paramaterised constructor
BankAccount(String n,float b, long a, String t){
    name = n;
    balance = b;
    accType = t;
    accNumber = a;
}

public static void main(String[] args){
    long a;
    String name;
    String type;
    float cash;

    BankAccount benk = new BankAccount();
    BankAccount benk1 = new BankAccount("Babu",8034.123f,1029343458l,"Savings");

    benk.display();
    benk1.display();

    a = 11066049032l;
    name = "Dhamodharan";
    cash = 2340.0f;
    type = "Checkings";

    benk.name = name;
    benk.accNumber=a;
    benk.balance=cash;
    benk.accType=type;

    benk.withdraw(9000.0f);
    benk1.withdraw(800f);
    benk.deposit(-2000f);

    benk.display();
    benk1.display();
}
```

```
}
```

Sample Output:

```
==================================================
Account Number: 0
Name of the Account Holder:
Account Type:
Balance: 0.0
==================================================


==================================================
Account Number: 1029343458
Name of the Account Holder: Babu
Account Type: Savings
Balance: 8034.123
==================================================

Error! Insufficient balance in Account
Remaining balance in Account = 2340.0

Sucessfully withdrawn 800.0 from the account
Remaining balance in Account = 7234.123

Error!!!! Enter a valid value


==================================================
Account Number: 11066049032
Name of the Account Holder: Dhamodharan
Account Type: Checkings
Balance: 2340.0
==================================================


==================================================
Account Number: 1029343458
Name of the Account Holder: Babu
Account Type: Savings
Balance: 7234.123
==================================================
```

Experiment 5

Employee Class

Date Of Submission: 17-09-2020

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.

Concepts Used: Class, Inheritance, Method Overloading, this keyword

Algorithms:

Class Employee
1. Declare fields name, age, phone, address, salary
2. Define printSalary and printDetails methods

Class Officer inherits Class Employee
1. Declare additional field specialization
2. Define constructors for with and without specialization

Class Manager inherits Class Employee
1. Declare additional field department
2. Define constructors for with and without department

Result: The program is successfully compiled and the required output is obtained

Program:

```
/*  Employee.java
 *
 *  Done By: Rohit Karunakaran
 *
 *  Details On Classes:
 *
 *  Employee:
 *      Attributes:
 *          Name -- String
 *          Age -- int
 *          PhoneNumber -- long
 *          Address -- String
 *          Salary -- float
 *      Methods:
 *          printSalary() -- prints the salary of the employee
 *
 *  Officer - inherited from Employee:
 *      Attributes:
 *          specialization -- String
 *
 *  Manager - inherited from Employee:
```

```
 *      Attributes:
 *          department -- String
 *
 * Task: assign name, age, phoneNo, addr and salary to an office and a manager
 *      by making an object of both and print the same
 * */


class Employee{
    String name;
    int age;
    long phoneNo;
    String addr;
    float salary;

    void displayEmployee(){
        System.out.println("Name: "+this.name);
        System.out.println("Address: "+this.addr);
        System.out.println("Phone Number: "+this.phoneNo);
        System.out.println("Salary : "+this.salary);
    }
       //Function to display the salary of the Employee
    void printSalary(){
        System.out.println("Salary of "+this.name+" is "+ this.salary);
    }

    Employee(String name, int age, long phoneNo, String addr,float salary){
        this.name = name;
        this.age = age;
        this.phoneNo = phoneNo;
        this.addr = addr;
        this.salary = salary;
    }

    //Default Constructor
       Employee(){
        this(" ",0,0l," ",0.0f);
    }

    public static void main(String [] args){
        Manager m1 = new Manager("Radhakrishnan",59,9823148320l,"123, Boulevard of Broken
Dreams",4529.19f,"Research and Development");
        Officer o1 = new Officer("Anto Davis",48, 9847120926l,"Green House Villa,
Pulmaidhanam", 3892.81f,"Corporate Security");
        m1.printSalary();
        o1.printSalary();

        m1.displayEmployee();
        o1.displayEmployee();
    }

}

class Officer extends Employee{
    String spec;

    void displayEmployee(){
        System.out.println("========================================");
        super.displayEmployee();
```

```
        System.out.println("Specialization in "+ this.spec);
        System.out.println("========================================");
        System.out.println("\n");
    }

    Officer(String name, int age, long phoneNo, String addr,float salary,String spec){
        super(name,age,phoneNo,addr,salary);
        this.spec = spec;
    }

    Officer(){
        this(" ",0,0l," ",0.0f," ");
    }
}

class Manager extends Employee{
    String dep;

    void displayEmployee(){
        System.out.println("========================================");
        super.displayEmployee();
        System.out.println("Manager of "+ this.dep);
        System.out.println("========================================");
        System.out.println("\n");
    }

    Manager(String name, int age, long phoneNo, String addr,float salary,String dep){
        super(name,age,phoneNo,addr,salary);
        this.dep= dep;
    }

    Manager(){
        this(" ",0,0l," ",0.0f," ");
    }

}
```

Sample Output:

```
Salary of Radhakrishnan is 4529.19
Salary of Anto Davis is 3892.81
========================================
Name: Radhakrishnan
Address: 123, Boulevard of Broken Dreams
Phone Number: 9823148320
Salary : 4529.19
Manager of Research and Development
========================================


========================================
Name: Anto Davis
Address: Green House Villa, Pulmaidhanam
Phone Number: 9847120926
Salary : 3892.81
Specialization in Corporate Security
========================================
```

Experiment 6

Inheritance

Date of Submission: 25-09-2020

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).

i) display() only prints the name of the class and does not return any value. Ex. " Name of class is Employee."

ii) calcSalary() in Employee displays "Salary of employee is 10000" and calcSalary() in Engineer displays"Salary of employee is 20000."

Concepts Used: Inheritance, Class, Method Overriding

Algorithm:

Class Employee
    algorithm details
        Step 1: Start
        Step 2: print "The name of the class is Employee"
        Step 3: Stop
    algorithm calcSalary
        Step 1: Start
        Step 2: print "The salary of employee is 10000"
        Step 3: Stop
Class Engineer inherits from Class Employee
    algorithm calcSalary
        Step 1: Start
        Step 2: print "The salary of employee is 20000"
        Step 3: Stop

Result: The program was successfully compiled and the required output was obtained.

Program:

```
/* Engineer.java
 *
 * Done By: Rohit Karunakaran
 * */

class Employee
{
    void display()
    {
        System.out.println("Name of the class is Employee");
    }
    void calcSalary()
    {
        System.out.println("Salary of employee is 10000");
    }
}

public class Engineer extends Employee
{
    void calcSalary()
    {
        System.out.println("Salary of employee is 20000");
    }

    public static void main(String[] args)
    {
        Engineer e1 = new Engineer();
        e1.display();
        e1.calcSalary();
    }
}
```

Sample Output:

```
Name of the class is Employee
Salary of employee is 20000
```

Experiment 7

Implementation of Abstract Class

Date of Submission: 25-09-2020

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: Class, Abstract Class, Inheritance

Algorithms:

Abstract Class Shape
    Abstract method numberOfSides

Class Rectangle inherits Shape
    numberOfSides()
        Step 1: Start
        Step 2: Print “The number of sides in a rectangle is 4”
        Step 3: Stop

Class Triangle inherits Shape
    numberOfSides()
        Step 1: Start
        Step 2: Print “The number of sides in a rectangle is 3”
        Step 3: Stop

Class Hexagon inherits Shape
    numberOfSides()
        Step 1: Start
        Step 2: Print “The number of sides in a rectangle is 6”
        Step 3: Stop

Result: The Program was successfully compiled and the required output was obtained.

Program:

```
/* Program2.java
 *
 *  Done By: Rohit Karunakaran
 *
 *  Abstract classes:
 *      Shape
 *  Classes:
 *      Rectangle : extends Shape
 *      Triangle : extends Shape
 *      Hexagon : extends Shape
 * */


abstract class Shape
{
    abstract void numberOfSides();
}

class Rectangle extends Shape
{
    void numberOfSides()
    {
        System.out.println("Number of sides in Rectangle= 4");
    }
}

class Triangle extends Shape
{
    void numberOfSides()
    {
        System.out.println("Number of sides in Triangle= 3");
    }
}

class Hexagon extends Shape
{
    void numberOfSides()
    {
        System.out.println("Number of sides in Hexagon = 6");
    }
}

public class Program2
```

```
{
    public static void main(String args[])
    {
        Rectangle r1 = new Rectangle();
        Triangle t1 = new Triangle();
        Hexagon h1 = new Hexagon();

        r1.numberOfSides();
        t1.numberOfSides();
        h1.numberOfSides();
    }
}
```

Sample Output:

```
Number of sides in Rectangle= 4
Number of sides in Triangle= 3
Number of sides in Hexagon = 6
```

Experiment 8

Packages

Date of Submission: 02-10-2020

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: Interface, Package

Algorithm:

```
1.  Package studpack
2.          Interface Sports
3.              grade: int
4.              displayGrad(): void
5.          Class Student
6.              name: String
7.              rollNo: int
8.              mark1: int
9.              mark2: int
10.             mark3: int
11.         Class Result implements Sports and extends Students
12.             total : int
13.             displayGrade():
14.                 Step1: print grade
15.
16.             displayTotal():
17.                 Step 1: total = mark1+mark2+mark3+grade
18.                 Step 2: print total
19.                 Step 3: print rollNo
20.                 Step 4: stop
21.
22.     Class Hybrid
23.         import studpack.Result
24.         main()
25.             r = new Result()
26.             r.displayTotal()
27.             r.displayGrade()
```

Result: The program is successfully compiled and the required output is obtained.

Program Code:

```
/*  File Name: Student.java
 *
 *  Done by Rohit Karunakaran
 *
 * */
package studpack;

public class Student
{

    String name;
    int rollNo;
    int mark1,mark2,mark3;

    public Student(String name, int rollNo,int mark1,int mark2,int mark3)
    {
        this.name = name;
        this.rollNo = rollNo;
        this.mark1 = mark1;
        this.mark2 = mark2;
        this.mark3 = mark3;
    }

    public Student()
    {
        this("",0,0,0,0);
    }

    public void displayTotal()
    {
        int total = mark1+mark2+mark3;
        System.out.println("Total marks earned by "+this.name+" is : "+total);
    }
}



//Results
/*  File name : Result.java
 *
 *  Done by : Rohit Karunakaran
 *
 */

package studpack;

interface Sports
{
    int grade = 34; //This is public static final
```

```
    void displayGrade();
}

public final class Result extends Student implements Sports
{
    private int total;

    public void displayGrade()
    {
        System.out.println("Grade = "+this.grade);
    }

    public Result(String name, int rollNo, int mark1, int mark2, int mark3)
    {
        super(name,rollNo,mark1,mark2,mark3);

        // total = marks + grade;
        this.total = this.mark1+this.mark2+this.mark3+this.grade;
    }

    public Result()
    {
        this("",0,0,0,0); //Constructor chaining
    }

    public void displayTotal()
    {
        System.out.println("\n=====================================");
        System.out.println("Name of the student is "+this.name);
        System.out.println("Roll No: "+this.rollNo);
        System.out.println("Total = "+this.total);
        System.out.println("=====================================\n");
    }

}


//Hybrid.java
/*  File Name : Hybrid.java
 *
 *  Done By: Rohit Karunakaran
 *
 *  The package studpack contains the classes Result and Students and a Sports
interface
 *
 *  The Result class implements Sports and inherits Student.
 * */
import studpack.Result;
import studpack.Student;

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

```
    {
        String name[] = {"Bhaskaran Pilla","Chandra Mohan","Sivadasan","Mani
Vessel","Narayani"};
        int rollNo[] = {4,11,56,34,41};
        int marks[] = {45,69,23};

        //Create an arry of 5 students
        Student A[]=new Result[5];  //Dynamic Method Dispatch

        //Set the values for the Students
        for(int i = 0;i<5;i++){
            A[i] = new Result(name[i],rollNo[i],marks[i%3]+2*i,marks[i
%3]+3*i,marks[i%3]+4*i);
        }

        //Print out the values for the students
        for(int i = 0;i<5;i++){
            A[i].displayTotal();
        }

    }
}
```

Sample output:

```
=====================================
Name of the student is Bhaskaran Pilla
Roll No: 4
Total = 169
=====================================


=====================================
Name of the student is Chandra Mohan
Roll No: 11
Total = 250
=====================================


=====================================
Name of the student is Sivadasan
Roll No: 56
Total = 121
=====================================


=====================================
Name of the student is Mani Vessel
Roll No: 34
Total = 196
=====================================
```

```
====================================
Name of the student is Narayani
Roll No: 41
Total = 277
====================================
```

Experiment 9

Exception Handling

Date of Submission: 16-10-2020

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)

Concepts used: Exception handling

(a)---Try-Finally example

Algorithm:

Input: command line arguments
Output: Corresponding to the argument given, the required exception is handled

//args is the command line arguments

1. Start
2. a = args.length
3. try
4. b = 3/a
5. end try
6. finally
7. print(“In the finally statement”)
8. endFinally
9. Stop

Program Code:

```
/* File name: TryFinallyExample.java
 *
 *  Done By: Rohit Karunakaran
 * */

public class TryFinallyExample
{
    public static void main(String args[]){
        int a =3;
        int len = args.length;
        try{
            int b = a/len;
        }

        finally{
            System.out.println("In the finally Statement");
            System.out.println("Have a nice Day");
            System.out.println("---------------------------------------\n");
        }
    }
}
```

Sample Input 1:

```
*no input*
```

Sample output 1:

```
In the finally Statement
Have a nice Day
---------------------------------------

Exception in thread "main" java.lang.ArithmeticException: / by zero
      at TryFinallyExample.main(TryFinallyExample.java:12)
```

Sample input 2:

```
hello
```

Sample output 2:

```
In the finally Statement
Have a nice Day
---------------------------------------
```

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

Algorithm

Input: command line arguments

Output: The Exception occured is handled properly

Steps

// args is the command line arguments

1. Start
2. try
3. len = args.length
4. i = Integer.valueOf(args[0])
5. b = Integer.valueOf(args[1])
6. c = i/b
7. endTry
8. catch ArrayIndexOutOfBoundsException e
9. print(“Not sufficent arguments”)
10. endCatch
11.
12. catch NumberFormatException e
13. print(“Numbers are the expected arguments”)
14. endCatch
15.
16. catch ArithmeticException e
17. print (“The second arguement is 0”)
18. endCatch
19. Stop

Program Code

```
/* File Name : MultipleCatchExample.java
 *
 * Done By: Rohit Karunakaran
 *
 * */

public class MultipleCatchExample{

    public static void main(String args[]){
        //3 catches for a single try
        System.out.println("Example for multiple Catch Statements");
        try
        {
            int len = args.length;

            int i = Integer.parseInt(args[0]);
            int b = Integer.parseInt(args[1]);
            int c = i/b;
        }

        catch(NumberFormatException e)
        {
            System.out.println("In the First Catch");
            System.out.println("An Exception has occurred while parsing the command
line input :"+e);
            System.out.println("Expected an integer");
        }

        catch(ArrayIndexOutOfBoundsException e)
        {
            System.out.println("In the Second Catch");
            System.out.println("Expected more than 1 command line argument");
            System.out.println("Exception occured is : "+e);
        }

        catch(ArithmeticException e)
        {
            System.out.println("In the Third Catch");
            System.out.println("An Exception has occured : "+e);
        }
        finally
        {
            //System.out.println("");
            System.out.println("---------------------------------------");
        }
    }

}
```

Sample input:

no input

Sample output:

```
Example for multiple Catch Statements
In the Second Catch
Expected more than 1 command line argument
Exception occured is : java.lang.ArrayIndexOutOfBoundsException: Index 0 out of
bounds for length 0
---------------------------------------
```

Sample input:

3 0

Sample output:

```
Example for multiple Catch Statements
In the Third Catch
An Exception has occured : java.lang.ArithmeticException: / by zero
---------------------------------------
```

Sample input:

3 hello

Sample output:

```
Example for multiple Catch Statements
In the First Catch
An Exception has occurred while parsing the command line
input :java.lang.NumberFormatException: For input string: "hello"
Expected an integer
---------------------------------------
```

(c)---Nested Try (3 levels of nesting)

Algorithm
Input : No input
Output: Corresponding try catch block is executed according to the exception (if occured)

Steps:

```
1.  a = {1,2,5,32,12}
2.  try
3.          b=a[3]
4.          print("In the outermost try block")
5.          try
6.                  print("Inside the inner try block")
7.                  c = 2*a[1]+a[0]-a[2]
8.                  try
9.                          print("Inside the inner most try block")
10.                         e= b/c
11.                 endtry
12.                 catch ArithmeticException e
13.                         print("divide by 0 error")
14.                 endcatch
15.
16.                 finally
17.                         print("In the finally of the inner most try block")
18.                 endfinally
19.         end try
20.
21.         finally
22.                 print("Inside the finally statement of the inner try block")
23.         end finally
24.         d = a[41]
25.         print("Aslong as the size of a is less than 42 this will not be executed")
26. endtry
27.
28. catch ArrayIndexOutOfBoundException
29.         print("The size of array is less")
30.
31. end catch
32.
33. finally
34.         print("In the final block of the outermost try")
35. end finally
```

Program code:

```
/* File name: NestedTryExample.java
 *
 * Done By: Rohit Karunakaran
 */

public class NestedTryExample
{
    static void main(String args[]){
        //3 levels of nesting
        int a[] = {1,2,5,32,12};
        System.out.println("\nExample With Nested Try Statements");
        try
        {
            System.out.println("Inside the Outer Most Try Block");
            System.out.println("================================\n");
            int b = a[3];
            try
            {
                System.out.println("Inside the Inner Try Block");
                System.out.println("================================\n");
                int c = 2*a[1]+a[0]-a[2];
                try
                {
                    System.out.println("Inside the Innermost Try Block");
                    System.out.println("================================\n");
                    int e = b/c;
                }

                catch(ArithmeticException e)
                {
                    System.out.println("An Exception has occured in the innerMost
try Block");
                    System.out.println("An Arithmetic expression error occured :
"+e);
                }

                finally
                {
                    System.out.println("In the final block of the Innermost try
expression");
                    System.out.println("================================\n");
                }
                //System.out.println("After the innermost try bolck Statement");
            }

            finally
            {
                System.out.println("In the final block of the Inner try
expression");
                System.out.println("================================\n");
            }

            int d = a[41]; //Array out of bounds exception
```

```
            System.out.println("I assure you that as long as the size of a is less
than 42,");
            System.out.println("This statement will not be excecuted");
        }

        catch(ArrayIndexOutOfBoundsException e)
        {
            System.out.println("Exception :"+e+" \nhas Occured in the outermost try
block");
        }

        finally
        {
            System.out.println("The Final Block of the Outermost try is complete");
            //System.out.println("==================================\n");
            System.out.println("----------------------------------------\n");
        }


    }
}
```

Sample Output:

```
Example With Nested Try Statements
Inside the Outer Most Try Block
==================================

Inside the Inner Try Block
==================================

Inside the Innermost Try Block
==================================

An Exception has occured in the innerMost try Block
An Arithmetic expression error occured : java.lang.ArithmeticException: / by zero
In the final block of the Innermost try expression
==================================

In the final block of the Inner try expression
==================================

Exception :java.lang.ArrayIndexOutOfBoundsException: Index 41 out of bounds for
length 5
has Occured in the outermost try block
The Final Block of the Outermost try is complete
----------------------------------------
```

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

Algorithm:
Input: Command line arguments (args)
Output: sum of numbers passed as the command line argument or the corresponding exeption

Steps:

1. Start
2. if args.length<2 then
3. throw new ArrayIndexOutOfBoundsException(“No sufficient arguments”)
4. else
5. print Integer.valueOf(args[0]) + Integer.valueOf(args[1])
6. Endif
7. Stop

Program Code:

```
/* File Name: NoSufficientArgument.java
 *
 * Done By: Rohit Karunakaran
 */
public class NoSufficientArgumentForAdding
{
    public static void main(String[] args) //throw Excenption
    {
        System.out.println("In the funcion noSufficientForAddingExample\n");

        try
        {
            if(args.length<2){
                throw new ArrayIndexOutOfBoundsException("Less than 2 command line
arguments found");
            }
            else{
            int a = Integer.parseInt(args[0]);
            int b = Integer.parseInt(args[1]);
            int c = a+b;

            System.out.println("No Exception has occurred. The sum = "+c);
            }
        }
        catch(ArrayIndexOutOfBoundsException e)
        {
            System.out.println(e);
        }
        catch(NumberFormatException e)
        {
```

```
            System.out.println("Pass an Integer as a command line argument");
            System.out.println(e);
        }
    }
}
```

Sample Input 1:
5 4

Sample output 1:
No Exception has occurred. The sum = 9

Sample Input 2:
5

Sample output 2:
java.lang.ArrayIndexOutOfBoundsException: Less than 2 command line arguments found

Sample Input 3:
5 hello

Sample output 3:
Pass an Integer as a command line argument
java.lang.NumberFormatException: For input string: "hello"

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

Algorithm:

1. Start
2. import java.io
3. //main function throws IOException and Arithmetic exception
4. try
5. File f = new File("file.txt")
6. FileReader fr = new FileReader(f)
7. endtry
8. catch IOException e
9. Print "An IOException has occurred"
10. endcatch
11. Stop

Program Code

```
/*File Name: ThrowsExample.java
 *
 *Done By: Rohit Karunakaran
 */

import java.io.*;

public class ThrowsExample
{

    public static void main(String args[]) throws IOException,
FileNotFoundException
    {
        File f = new File("File1.txt");
        FileReader f1 = new FileReader(f);
        System.out.println("The file is opened");
        f1.close();
        f.close();

    }

}
```

Sample output:

```
//Before creating the file
Exception in thread "main" java.io.FileNotFoundException: File1.txt (No such file or
directory)
      at java.base/java.io.FileInputStream.open0(Native Method)
      at java.base/java.io.FileInputStream.open(FileInputStream.java:211)
      at java.base/java.io.FileInputStream.<init>(FileInputStream.java:153)
      at java.base/java.io.FileReader.<init>(FileReader.java:75)
      at ThrowsExample.main(ThrowsExample.java:14)
```

Sample output:

```
//After creating the file
The file is opened
```

Result: The programs were successfully compiled and the required output was obtained

Experiment – 10

File Handling

Date of Submission: 23-10-2020

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

Concepts Used: File Input and output, exception handling

Algorithm:

1. Start
2. import java.io package
3. fileName = “file.txt”
4. File f = new File(fileName)
5. if(!.f.exists()) then
6. f.createNewFile()
7. endif
8.
9. try
10. FileReader fr = new FileReader(file)
11. File copy = File(“copy.txt”)
12. if(!copy.exists()) then
13. copy.createNewFile()
14. endif
15. FileWriter fw = new FileWriter(copy)
16. while i=fr.read() and i!=-1 do
17. fw.write(i)
18. endwhile
19. endtry
20. catch FileNotFoundException e
21. Print “File is not found”
22. endcatch
23. Stop
24.

Result: The program was compiled successfully and the required output was obtained

Program Code:

```
/* Java Program to read and write to a file
 * by: Rohit Karunakaran
 *
 */

import java.io.*;


public class ReadWriteToFile
{
    public static void main(String args[]) throws IOException,
FileNotFoundException
    {

        //BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int i;
        String fileName = "file.txt";
        File originalFile = new File(fileName);

        if(!originalFile.exists()) originalFile.createNewFile();

        try
        {
            FileReader fileReader1 = new FileReader(fileName);
            File copy = new File("copy.txt");

            if(!copy.exists())
            {
                copy.createNewFile();
            }

            try
            {
                FileWriter fileOutput = new FileWriter(copy);

                System.out.println("The String in the old file is : ");
                while ((i=fileReader1.read())!=-1)
                {
                    fileOutput.write((char)i);
                    System.out.print((char)i);
                }
                System.out.println(" ");
                fileOutput.flush();
                fileOutput.close();
            }
            catch (FileNotFoundException e)
            {
                System.out.println("The File is not writable or the file doesnt
exist");
                e.printStackTrace();
            }
            finally
            {
                fileReader1.close();
```

```
            }

            FileReader fileReader2 = new FileReader(copy);

            System.out.println("\nThe contents of the new file is :");
            while((i=fileReader2.read())!=-1)System.out.print((char)i);
            System.out.print("\n");

        }
        catch (FileNotFoundException e)
        {
            System.out.println("File is not found or the file is not readable "+e);
        }
    }
}
```

Sample Output:

```
The String in the old file is :
I am a file and I think there is a copy of me somewhere here


The contents of the new file is :
I am a file and I think there is a copy of me somewhere here
```

Experiment – 11

Console Input and Output

Date of Submission: 23-10-2020

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: String Tokenizer, Console input

Algorithm:
Input: A string containing of digits
Output: Sum of the digits

Steps

1. Start
2. import java.io package and StringTokenizer Class from java.util package
3. sum = 0
4. BufferedReader br = new BufferedReader(new InputStreamReader(System.in))
5. s = br.readLine()
6. StringTokenizer st = new StringTokenizer(s,”0123456789”,true)
7. while(st.hasMoreTokens())do
8. try
9. a = Integer.parseInt(st.nextToken())
10. sum+=a
11. endtry
12. catch NumberFormatException
13. print ”Number expected”
14. endcatch
15. endwhile
16. Stop

Reuslt: The program was successfully compiled and the required output was obtained

Program Code:

```
/* Read interger as a stirng and print it's sum
 * File Name: StringTokenizerExample.java
 *
 * Done By: Rohit Karunakaran
 * */

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

```
public class StringTokenizerExample
{
    public static void main(String args[]) throws IOException
    {
        int sum = 0,a=0;

        System.out.println("Enter an Integer Value");

        String s = new String(new BufferedReader(new
InputStreamReader(System.in)).readLine()); //reads a line from the input

        StringTokenizer st = new StringTokenizer(s,"0123456789",true); //if the
boolean argument is true each of the delimeter is considered as tokens themselves

        System.out.print("The Number entered = ");
        while(st.hasMoreTokens())
        {
            try
            {
                a = Integer.parseInt(st.nextToken());
                System.out.print(a);
            }
            catch(NumberFormatException e)
            {
                System.out.println("Expected an integer value");
                System.out.println("\nHere is the stack trace");

                e.printStackTrace();
            }

            sum+=a;
        }

        System.out.println("\nSum = "+sum);
    }
}
```

Sample Input 1:

```
10000000000000000000000000000000000000000000000000000000009
```

Sample output 1:

```
Enter an Integer Value : 10000000000000000000000000000000000000000000000000000000009

The Number entered = 10000000000000000000000000000000000000000000000000000000009
Sum = 10
```

Sample Input 2:

```
1329
```

Sample output 2:
Enter an Integer Value : 1329

The Number entered = 1329
Sum = 15

Sample input 3:
247298379237

Sample output 3
Enter an Integer Value : 247298379237

The Number entered = 247298379237
Sum = 63

Experiment - 12

Doubly Linked List

Date: 30-10-2020

Aim: Write a Java program for the following:

1) Create a doubly linked list of elements.
2) Delete a given element from the above list.
3) Display the contents of the list after deletion.

Concepts Used: Doubly Linked List

Algorithm:

Class Node
    Data Members
        data:int
        prev : Node
        next :Node

Class DoublyLinkedList
    Data Members
        Header: Node
        size: int
    Methods()
        addNode(): void
        removeNode(): void
        display(): void

Algorithm addNode(int elem):

1. Start
2. Node n = new Node()
3. if n!=NULL then
4. n->data = elem
5. Node n2 = Header
6. while(n2->next!=NULL) do
7. n2 = n2->next
8. endWhile
9. n2->next = n
10. n->next = NULL
11. n->prev = n2
12. endif
13. Stop

Algorithm removeNode(int a)

1. Start

2. if Header->next!=NULL then
3. flag=0
4. Node ptr = Header->next
5. while ptr -> next!=NULL do
6. if ptr->data == a then
7. flag =1
8. endif
9. endwhile
10. if flag == 1 then
11. if(ptr->next!=NULL) then
12. ptr->next->prev = ptr->prev
13. endif
14. ptr->prev->next = ptr->next
15. ptr->prev=NULL
16. ptr->next = NULL
17. else
18. print “element not found”
19. endif
20. endif
21. Stop

Program Code:

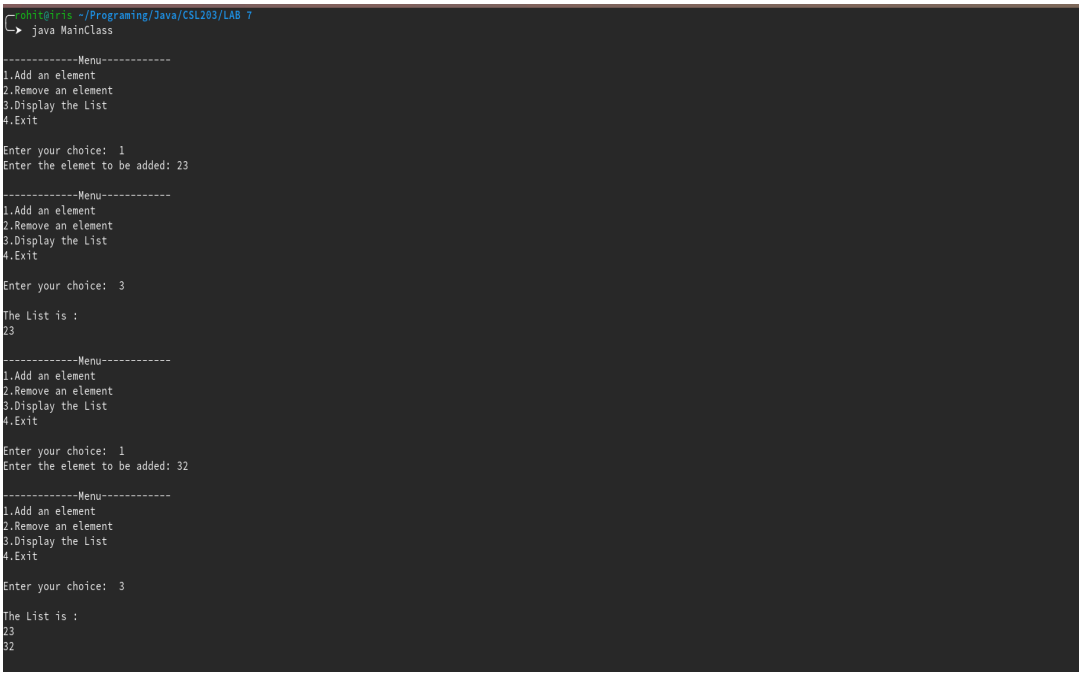
```
/*Doubly Linked List implementation in Java
 * Done By: Rohit Karunakaran
 * */
import java.io.*;

/*Node class for the nodes of the linked list*/
class Node
{
    private int data;
    private Node prev;
    private Node next;

    //Constructors
    public Node(int x,Node next,Node prev)
    {
        data = x;
        this.prev =prev;
        this.next = next;
    }
    public Node(int x){ this(x,null,null); }
    public Node() { this(0); }


    //Gettes and Setters
    public Node getNextNode() { return this.next; }
    public Node getPrevNode() { return this.prev; }
    public int getData() { return this.data; }
```

```
    public void setNextNode(Node n) { this.next=n; }
    public void setPrevNode(Node n) { this.prev=n; }
    public void setNextNode() { this.next=null; }
    public void setPrevNode() { this.prev=null; }
}


/*  Doubly linked list class that contains the relevent functions for
 *  implementation*/

class DoublyLinkedList
{
    private Node header; //The header node
    public int length; //To keep a track of the length of the doubly linked list

    public DoublyLinkedList()
    {
        header = new Node();
        length=0;
    }

    public DoublyLinkedList(int nums[]) //Creates a doubly linked list when an
array of numbers is passed
    {
        this();
        for(int x:nums)
        {
            this.add(x);
        }

    }

    public void add(int x) //add a node to the end of the doubly linked list
    {
        Node ptr=header;
        while(ptr.getNextNode()!=null)
            ptr=ptr.getNextNode();
        Node n = new Node(x);
        n.setPrevNode(ptr);
        n.setNextNode(ptr.getNextNode());
        ptr.setNextNode(n);
        length++;
    }

    public void remove(int x) //remove the node containing the given value if it
exists
    {
        Node ptr=header.getNextNode();
        if(ptr==null)
        {
            System.out.println("The List is empty");
            return;
```

```
        }
        while(ptr!=null)
        {
            if(ptr.getData() == x)
                break;
            ptr=ptr.getNextNode();
        }
        if(ptr!=null)
        {
            //delete node
            if(ptr.getNextNode()!=null)
                ptr.getNextNode().setPrevNode(ptr.getPrevNode());
            ptr.getPrevNode().setNextNode(ptr.getNextNode());
            ptr.setNextNode(null);
            ptr.setPrevNode(null);
            length--;
        }
        else
        {
            System.out.println("No Such element found");
        }

    }


    public void displayList()
    {
        Node ptr=header.getNextNode();
        while(ptr!=null)
        {
            System.out.println(ptr.getData());
            ptr=ptr.getNextNode();
        }
    }
}

public class MainClass
{
    public static void main(String args[]) throws IOException
    {
        DoublyLinkedList dll = new DoublyLinkedList();
        int elem=0;
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in));
        boolean Run = true;

        while(Run)
        {
            System.out.println("\n-------------Menu------------");
            System.out.println("1.Add an element");
            System.out.println("2.Remove an element");
            System.out.println("3.Display the List");
            System.out.println("4.Exit");
            System.out.print("\nEnter your choice:  ");
```

```
            try
            {
                int c = Integer.parseInt(r.readLine());
                switch(c)
                {
                    case 1: //add an element
                            System.out.print("Enter the elemet to be added: ");
                            elem = Integer.parseInt(r.readLine());
                            dll.add(elem);
                            break;
                    case 2: //remove an element
                            System.out.print("Enter the elemet to be deleted: ");
                            elem = Integer.parseInt(r.readLine());
                            dll.remove(elem);
                            break;
                    case 3: //display the list
                            System.out.println("\nThe List is :");
                            dll.displayList();
                            break;
                    case 4:Run = false;
                           break;
                    default: System.out.println("Please Enter a valid input ");
                             break;
                }
            }
            catch (NumberFormatException e)
            {
                System.out.println("Please Enter a integer value ");
                e.printStackTrace();
            }
        }
    }
}
```

Sample input/output:



```
------------Menu------------
1.Add an element
2.Remove an element
3.Display the List
4.Exit

Enter your choice:  1
Enter the elemet to be added: 83

------------Menu------------
1.Add an element
2.Remove an element
3.Display the List
4.Exit

Enter your choice:  2
Enter the elemet to be deleted: 32

------------Menu------------
1.Add an element
2.Remove an element
3.Display the List
4.Exit

Enter your choice:  3

The List is :
23
83

------------Menu------------
1.Add an element
2.Remove an element
3.Display the List
4.Exit

Enter your choice:  2
Enter the elemet to be deleted: 14
No Such element found

------------Menu------------
1.Add an element
2.Remove an element
3.Display the List
4.Exit

Enter your choice:  2
```

```
------------Menu------------
1.Add an element
2.Remove an element
3.Display the List
4.Exit

Enter your choice:  2
Enter the elemet to be deleted: 14
No Such element found

------------Menu------------
1.Add an element
2.Remove an element
3.Display the List
4.Exit

Enter your choice:  2
Enter the elemet to be deleted: 23

------------Menu------------
1.Add an element
2.Remove an element
3.Display the List
4.Exit

Enter your choice:  3

The List is :
83

------------Menu------------
1.Add an element
2.Remove an element
3.Display the List
4.Exit

Enter your choice:  4
┌─rohit@iris ~/Programing/Java/CSL203/LAB 7
└─➤
```

Experiment – 13
Binary Search

Date: 30-10-2020

Aim: Java Program to implement Binary Search algorithm

Concepts Used: Arrays, BinarySearch

Algorithm BinarySearch

Input: the array, the starting index the ending index and the element to be searched
Output: the index of the element, -1 if the element doesn't exist
Steps
Start
if(start<=last) then
    mid = (start+last)/2
    if(a[mid] == elem) then
        return mid
    else if(a[mid]>elem)
        return BinarySearch(a,start, mid,elem)
    else
        return BinarySearch(a,mid+1,last,elem)
    endif
else
    return -1
endif
Stop

Program Code:

```
/* Binary search algorithm implementation in java
 * Done By: Rohit Karunakaran
 **/

import java.util.ArrayList;
import java.util.StringTokenizer;
import java.io.*;

class BinarySearch
{

    //Recursive binary search funtion

    static int binarySearch(ArrayList<Integer> a,int elem,int beg,int last)
    {
        int mid = (beg+last)/2;
        if(beg<=last)
        {
            if(a.get(mid)==elem)
```

```
                return mid;
            else if(a.get(mid)>elem)
                return binarySearch(a,elem,beg,mid);
            else
                return binarySearch(a,elem,mid+1,last);
        }
        else
        {
            return -1; //If the element is not found it will return -1
        }
    }

    public static void main(String args[]) throws IOException
    {

        ArrayList<Integer> arr = new ArrayList<Integer>();
        int elem=0;
        try
        {
            System.out.print("Enter the elements in the array in ascending order in
th form  \"1 2 32 65 75 \" \nwith out the quotes: ");
            BufferedReader br= new BufferedReader(new InputStreamReader(System.in));

            String nums = br.readLine();

            StringTokenizer st = new StringTokenizer(nums," ");

            while(st.hasMoreTokens())
            {
                arr.add(Integer.parseInt(st.nextToken()));
            }

            System.out.print("Enter the element to be searched ");
            elem = Integer.parseInt(br.readLine());
            //arr.sort();
            int index = binarySearch(arr,elem,0,arr.size()-1);

            if(index==-1)
            {
                System.out.println("The element is not found\n");
            }
            else
            {
                System.out.println("The element is found at index "+index);
            }
        }
        catch(NumberFormatException e)
        {
            System.out.println("A Number expected ");
            e.printStackTrace();
        }

    }
}
```

Sample input/output

```
rohit@iris ~/Programing/Java/CSL203/LAB 7
➜  javac BinarySearch.java
rohit@iris ~/Programing/Java/CSL203/LAB 7
➜  java BinarySearch
Enter the elements in the array in ascending order in th form  "1 2 32 65 75 "
with out the quotes: 1 2 32 65 75
Enter the element to be searched 75
The element is found at index 4
rohit@iris ~/Programing/Java/CSL203/LAB 7
➜  java BinarySearch
Enter the elements in the array in ascending order in th form  "1 2 32 65 75 "
with out the quotes: 3 21 38 39 42 47 65 70
Enter the element to be searched 39
The element is found at index 3
rohit@iris ~/Programing/Java/CSL203/LAB 7
➜  java BinarySearch
Enter the elements in the array in ascending order in th form  "1 2 32 65 75 "
with out the quotes: 3 3 3 3 3 3 3 3 3
Enter the element to be searched 3
The element is found at index 4
rohit@iris ~/Programing/Java/CSL203/LAB 7
➜  java BinarySearch
Enter the elements in the array in ascending order in th form  "1 2 32 65 75 "
with out the quotes: 8 12 12 14 18 20 31
Enter the element to be searched 12
The element is found at index 1
rohit@iris ~/Programing/Java/CSL203/LAB 7
➜
```

```
rohit@iris ~/Programing/Java/CSL203/LAB 7
➜  java BinarySearch
Enter the elements in the array in ascending order in th form  "1 2 32 65 75 "
with out the quotes: 1 2 3 4 5 6
Enter the element to be searched 7
The element is not found

rohit@iris ~/Programing/Java/CSL203/LAB 7
➜
```

Experiment – 14

Multi-Threading

Date Of Submission: 6-11-2020

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.

Concepts Used: Multithreading

Algorithm:

Class Square implements Runnable
- Data Members
  - i : int
  - t : Thread
- Methods:
  - Override run method in Runnable interface
    - Start
    - int sq = i*i
    - Print “Square = i”
    - Stop
  - Square (a) //constructor
    - Start
    - i = a
    - t = new Thread(this,”Square”)
    - Stop

Class Cube implements Runnable
- Data Members
  - i : int
  - t : Thread
- Methods:
  - Override run method in Runnable interface
    - Start
    - int cu = i*i*i
    - Print “cube = cu”
    - Stop
  - Cube (a) //constructor
    - Start
    - i = a
    - t = new Thread(this,”Square”)
    - Stop

Class Random implements runnable

Data Members

i: int

t: Thread

Methods:

Random: //constructor

Start

t = new Thread(this,”Random number generator”)

Stop

Override run from Runnable:

Start

i = Math.random()*100

if(i%2==0) then

new Square(i).t.start()

else

new Cube(i).t.start()

endif

Stop

Program Code:

```
/*******************************************************
 * A Multi-threaded Program which has 3 threads,
 * Thread 1: prints a random number every 1 Second;
 * Thread 2: If the number is even then print it's square
 * Thread 3: If the number is odd then print the cube
 * *****************************************************/
/*******************************************************
 * Done By: Rohit Karunakaran
 *******************************************************/


class SquareThread implements Runnable
{
    int i;
    public Thread t;
    public SquareThread(int a)
    {
        t = new Thread(this,"Square Thread");
        i = a;
    }
    public void run()
    {
        int sq = i*i;
        System.out.println("The square of the number "+i+" is "+sq);
    }
```

```
}

class CubeThread implements Runnable
{
    public Thread t;
    int i;
    public CubeThread(int a)
    {
        t = new Thread(this,"Cube Thread");
        i = a;
    }
    public void run()
    {
        int qube = i*i*i;
        System.out.println("The Cube of the number "+i+" is "+qube);
    }
}


class RandomThread implements Runnable
{
    int i;
    public Thread t;

    public RandomThread()
    {
        i = (int)Math.random()*100;
        t = new Thread(this,"Random Number");
    }

    public void run()
    {
        for(int j=0;j<10;j++)
        {
         //Generates a random number from 0-99
            i = (int)(Math.random()*100);
            System.out.println("Random Number : "+i);
            if(i%2==0)
            {
                //Square thread
                new SquareThread(i).t.start();
                //s.t.start();
            }
            else
            {
                //cubeThread
                new CubeThread(i).t.start();
                //c.t.start();
            }
            try
            {
                Thread.sleep(1000);
            }
            catch(InterruptedException e)
            {
                System.out.println("Interrupted");
```

```
            }
        }
    }
}

public class  RandomNumber
{
    public static void main(String args[])
    {
        RandomThread r =new RandomThread();
        r.t.start();

    }

}
```

Sample Input/Output

```
rohit@iris ~/Programing/Java/CSL203/LAB 8
➜  java RandomNumber
Random Number : 71
The Cube of the number 71 is 357911
Random Number : 15
The Cube of the number 15 is 3375
Random Number : 78
The square of the number 78 is 6084
Random Number : 45
The Cube of the number 45 is 91125
Random Number : 11
The Cube of the number 11 is 1331
Random Number : 63
The Cube of the number 63 is 250047
Random Number : 91
The Cube of the number 91 is 753571
Random Number : 47
The Cube of the number 47 is 103823
Random Number : 5
The Cube of the number 5 is 125
Random Number : 90
The square of the number 90 is 8100
rohit@iris ~/Programing/Java/CSL203/LAB 8
➜
```

```
rohit@iris ~/Programing/Java/CSL203/LAB 8
➜  java RandomNumber
Random Number : 34
The square of the number 34 is 1156
Random Number : 33
The Cube of the number 33 is 35937
Random Number : 90
The square of the number 90 is 8100
Random Number : 14
The square of the number 14 is 196
Random Number : 35
The Cube of the number 35 is 42875
Random Number : 5
The Cube of the number 5 is 125
Random Number : 42
The square of the number 42 is 1764
Random Number : 34
The square of the number 34 is 1156
Random Number : 47
The Cube of the number 47 is 103823
Random Number : 35
The Cube of the number 35 is 42875
rohit@iris ~/Programing/Java/CSL203/LAB 8
➜
```

Experiment – 15

Thread Synchronization

Date Of Submission: 6-11-2020

Aim: Write a Java program that shows thread synchronization.

Concepts used: Thread Synchronization

Algorithm:

```
Class Test
    Methods:
        test (String msg)
            Print "["
            Thread.sleep(1000)
            Print "]"
            Stop
Class ThreadInSync implements Runnable
    Data Members
        msg: String
        target : Test
        t : Thread
    Methods
        ThreadInSync (a : Test, s: String)
            Start
            target = a
            msg = s
            t = new Thread(this)
            Stop
        Override run from Runnable interface
            Start
            target.test(msg)
            Thread.sleep(2000)
            synchronized (this) do
                target.test(msg)
            end synchronised
            Stop
Class Main
    Static Method
        main()
            Start
            test t = new test
            ThreadInSync  t1 = new ThreadInSync(t,"Thread")
```

ThreadInSync t2 = new ThreadInSync(t,”synchronized”)
ThreadInSync t1 = new ThreadInSync(t,”Hello”)

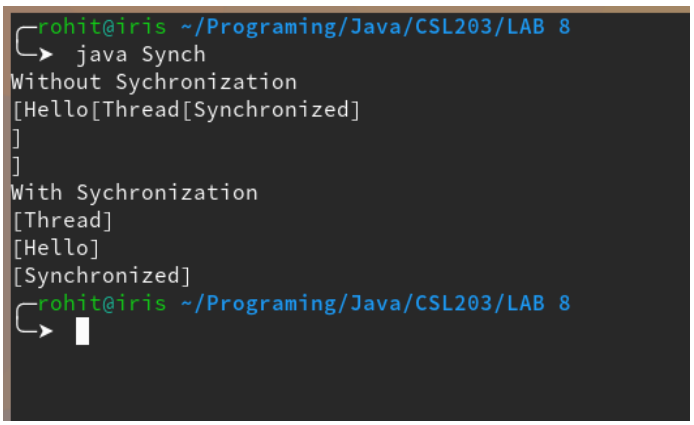
t1.start()
t2.start()
t3.start()
Stop

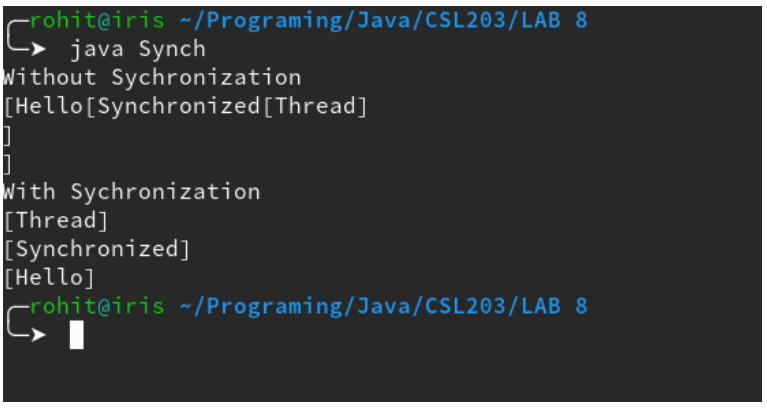
Program Code:

```
/*********************************************
 * Java Program to Demonstrate Thread Synchronization
 * Done By: Rohit Karunakaran
 * *******************************************/


class Test
{
    void test(String msg)
    {
        System.out.print("["+msg);
        try{
            Thread.sleep(1000);

        }
        catch(InterruptedException e)
        {
            System.out.println("Interrupted");
        }
        System.out.println("]");
    }
}

class ThreadsInSync implements Runnable
{
    String msg;
    Test target;
    Thread t;

    public ThreadsInSync(Test targ,String s)
    {
        target = targ;
        msg = s;
        t= new Thread(this);
    }

    public void run()
    {
        target.test(msg);
        try{
            Thread.sleep(2000);
        }
        catch(InterruptedException e)
```

```
        {
            System.out.println("Interrupted");
        }

        synchronized(target){
            target.test(msg);
        }

    }
}

public class Synch
{
    public static void main(String args[])
    {
        Test target = new Test();
        ThreadsInSync ob1 = new ThreadsInSync(target,"Hello");
        ThreadsInSync ob2 = new ThreadsInSync(target,"Synchronized");
        ThreadsInSync ob3 = new ThreadsInSync(target,"Thread");

        System.out.println("Without Sychronization");

        ob1.t.start();
        ob2.t.start();
        ob3.t.start();

        try{
            Thread.sleep(2900);
            System.out.println("With Sychronization");

        }
        catch(InterruptedException e)
        {
            System.out.println("Interrupted");
        }

    }
}
```

Sample Input/Output:





Experiment 16
Calculator using Java Swing

Date of submission: 18-11-2020

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.

Concepts Used: Java Swing and event handling

Algorithm:

Class Calulator

1. Create JFrame with title Calculator and setDefaultCloseOperation to EXIT_ON_CLOSE
2. Set the layout manger for the JFrame to GridBagLayout
3. Create the JLabel to display the result and to show the numbers entered
4. Add the Jlabel to `display` the number entered and the result
5. Add the Jbuttons with the symbols 1,2,3,4,5,6,7,8,9,0 for the numbers and +,-,*,/ for the operations
6. //Event handling
7. For every numberbutton:
8. if the button is pressed, append the character to the Jlabel
9. Done
10.
11. If +,-,*,/ button pressed
12. case + : add = true,sub=false,mul=false,div=false
13. break
14. case - : add = false,sub=true,mul=false,div=false
15. break
16. case * : add = true,sub=false,mul=true,div=false
17. break
18. case / : add = false,sub=false,mul=false,div=true
19. break
20. endcase
21. a = Float.valueOf(display.getText())
22. endFor
23.
24. If ‘=’ button is pressed
25. flag =false
26. if(add or sub or mul or div of mod)
27. b = Float.valueOf(display.getText())
28. case add: res=a+b
29. break

```
30.                     case sub: res = a-b
31.                             break
32.                      case mul: res=a*b
33.                             break
34.                     case div: if b==0
35.                                     flag=true
36.                             else
37.                                     res = a/b
38.                             endif
39.                     endcase
40.                     if flag then
41.                             display.setText(“Divide by 0 error”)
42.                     else
43.                             display.setText(res)
44.                     endif
45. endif
```

Result: The program is successfully compiled and the required output is obtained

Program code:

```
/*****************************************
 * Calculator implementing the funtions +,-,*,/
 * Done By: Rohit Karunakaran
 * ***************************************/
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;


public class Calculator
{
   JFrame jfrm;
   JLabel disp;
   GridBagConstraints c ;
   ActionListener numberButtonPressed;
   ActionListener mathButtonPressed;
   boolean add;
   boolean sub;
   boolean mul;
   boolean div;
   boolean done;

   double calc;

   public Calculator()
   {
      jfrm = new JFrame("Calculator");
```

```
jfrm.setLayout(new GridBagLayout());
c = new GridBagConstraints();
calc = 0;
add=false;
sub=false;
div=false;
mul=false;
done = true;

jfrm.setSize(270,330);
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

disp = new JLabel(String.valueOf(calc));

numberButtonPressed = new ActionListener()
{
   public void actionPerformed(ActionEvent ae)
   {
      JButton callerButton = (JButton)ae.getSource();
      String val = callerButton.getText();
      String displayText = disp.getText();

      if(displayText.equals("ERROR"))
      {
         disp.setText(val);
      }
      else
      {
         if(done ==true){
               disp.setText(String.valueOf(Double.parseDouble(val)));
         }
         else{
               double newVal = Double.parseDouble(displayText)*10+Double.parseDouble(val);
               disp.setText(String.valueOf(newVal));
            }
      }
      done = false;
   }
};

mathButtonPressed = new ActionListener()
{
   public void actionPerformed(ActionEvent ae)
   {
      //done = true;
      JButton b = (JButton) ae.getSource();
      double result = 0.0;
      String val = disp.getText();
      if(!done)
```

```
{
   if(val.equals("ERROR"))
   {
      disp.setText("0.0");
      calc = 0.0;
   }
   else
   {
      if(add||sub||div||mul)
      {
         double operand= Double.parseDouble(val);
         if(add)
         {
            result = calc+operand;
            add = false;
         }
         else if(sub)
         {
            result = calc-operand;
            sub = false;
         }
         else if(mul)
         {
            result = calc*operand;
            mul = false;
         }
         else if(div)
         {
            if(operand!=0.0)
               result = calc/operand;
            else{
               disp.setText("ERROR");
               done = true;
               return;
            }
            div = false;
         }
         calc = result;
         disp.setText(String.valueOf(calc));

      }
      else
      {
         calc = Double.parseDouble(val);
         char op = b.getText().charAt(0);
         switch(op)
         {
            case '+':add=true;break;
            case '*':mul=true;break;
```

```
                    case '/':div=true;break;
                    case '-':sub=true;break;
                }
            }

            }
            done = true;
        }
    }
   };

   c.anchor = GridBagConstraints.FIRST_LINE_START;
   c.fill= GridBagConstraints.VERTICAL;
   c.weightx=0.5;
   c.gridx=0; c.gridy=0;
   c.gridwidth = 3;
   c.ipady = 20;

   jfrm.add(disp,c);
   addButtons();
   jfrm.setVisible(true);
}

private void addButtons()
{
  c.ipady = 10;
  c.gridwidth = 1;
  c.fill= GridBagConstraints.HORIZONTAL;
  c.anchor = GridBagConstraints.LINE_START;
  JButton numbers[] = new JButton[10];
  for(int i = 0;i<10;i++)
  {
      numbers[i] = new JButton(String.valueOf(i));
      numbers[i].addActionListener(numberButtonPressed);
      if(i!=0)
         c.gridx = (i+2)%3;
      else
         c.gridx = 1;
      c.gridy = i%3==0?4-(i/3-1):4-(i/3);
      jfrm.add(numbers[i],c);
  }
  JButton addButton= new JButton("+");
  addButton.addActionListener(mathButtonPressed);

  JButton subButton= new JButton("-");
  subButton.addActionListener(mathButtonPressed);

  JButton mulButton= new JButton("x");
  mulButton.addActionListener(mathButtonPressed);
```

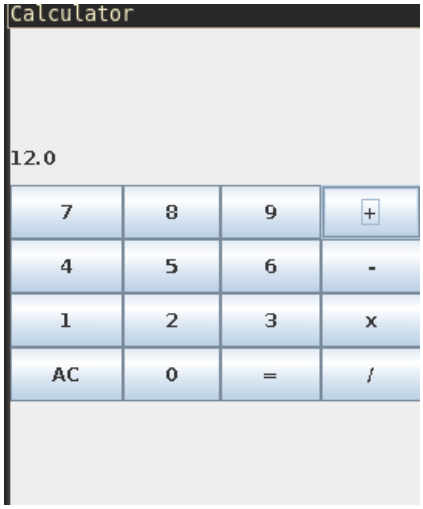
```
JButton divButton= new JButton("/");
divButton.addActionListener(mathButtonPressed);

c.gridx = 3;
c.gridy = 2;
jfrm.add(addButton,c);
c.gridy = 3;
jfrm.add(subButton,c);
c.gridy =4;
jfrm.add(mulButton,c);
c.gridy = 5;
jfrm.add(divButton,c);

JButton equalButton = new JButton("=");
equalButton.addActionListener(new ActionListener()
      {
          public void actionPerformed(ActionEvent ae)
          {
             double result = 0.0;
             String val = disp.getText();
             if(!done)
             {
                if(val.equals("ERROR"))
                {
                   disp.setText("0.0");
                   calc = 0.0;
                }
                else
                {
                   if(add||sub||div||mul)
                   {
                      double operand= Double.parseDouble(val);
                      if(add)
                      {
                         result = calc+operand;
                         add = false;
                      }
                      else if(sub)
                      {
                         result = calc-operand;
                         sub = false;
                      }
                      else if(mul)
                      {
                         result = calc*operand;
                         mul = false;
                      }
                      else if(div)
```

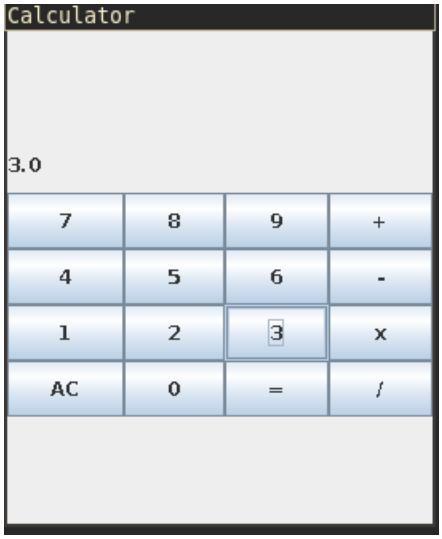
```
                    {
                        if(operand!=0.0)
                            result = calc/operand;
                        else{
                            disp.setText("ERROR");
                            done = true;
                            return;
                            }
                        div = false;
                    }
                    calc = result;
                    disp.setText(String.valueOf(calc));
                }
                else
                {
                    calc = Double.parseDouble(val);
                }

            }
            done = true;
        }
    }
});

    c.gridy=5;
    c.gridx =2;
    jfrm.add(equalButton,c);

    c.gridx = 0;
    JButton clearAll = new JButton("AC");
    clearAll.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent ae)
            {
                add = mul = div = sub = false;
                done = true;
                disp.setText("0.0");
            }
        }
        );
    jfrm.add(clearAll,c);

}


public static void main(String[] args)
{
    SwingUtilities.invokeLater(new Runnable()
        {
```

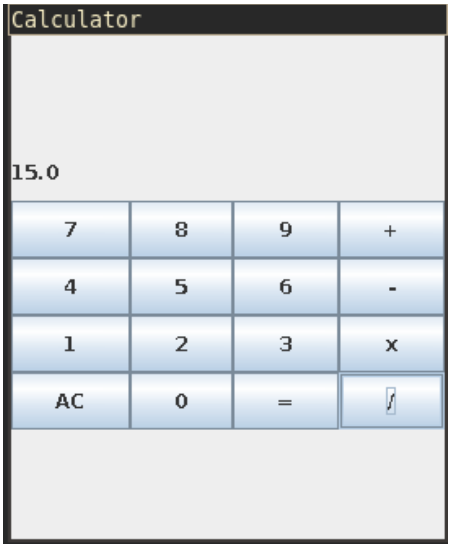
```
                public void run()
                {
                    new Calculator();
                }
            });
    }
}
```

Sample output

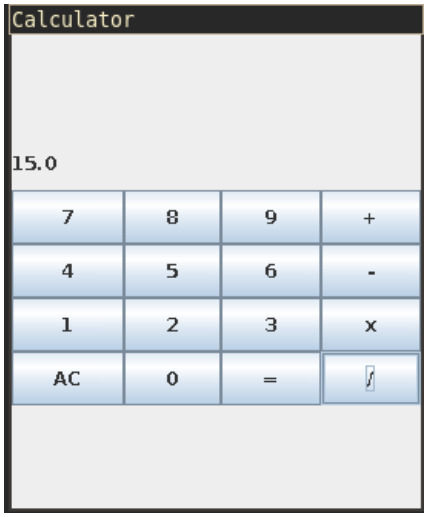
12+3 =15

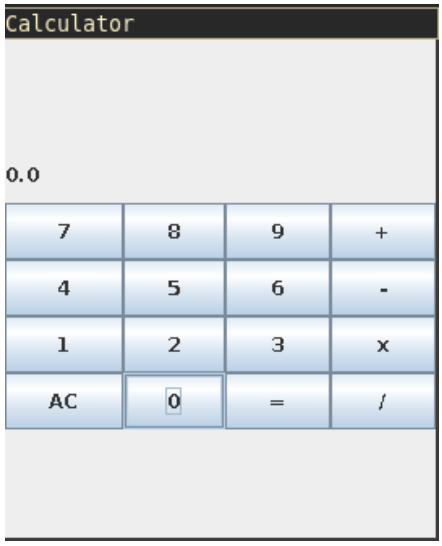


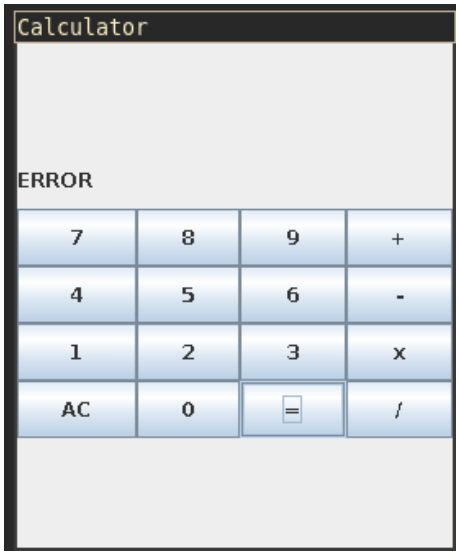




15 / 0 = ERROR







Experiment 17
Traffic Lights using Java Swing

Date of submission: 18-11-2020

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.

Concepts Used: Java Swing, Event handling

Algorithm:

1. Create 3 JradioButtons buttons[] with the values “Red” “Yellow” “Green” for the three buttons
2. buttons[0].setBackground(Color..GRAY)
3. buttons[2].setBackground(Color..GRAY)
4. buttons[1].setBackground(Color..GRAY)
5.
6. add the buttons to the Jframe
7. Create a ButtonGroup bg
8. Add the buttons to bg
9.
10. //Event handling
11. button[0] is pressed
12. set button[1] and button[2] to Grey
13. set button[0] tored
14. button[1] is pressed
15. set button[1] to yellow
16. set button[0] and button[2] to Grey
17. button[2] is pressed
18. set button[2] to green
19. set button[0] and button[1] to Grey

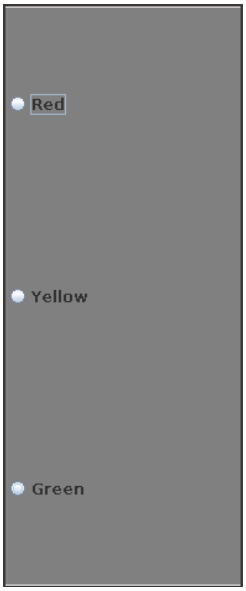
Program Code

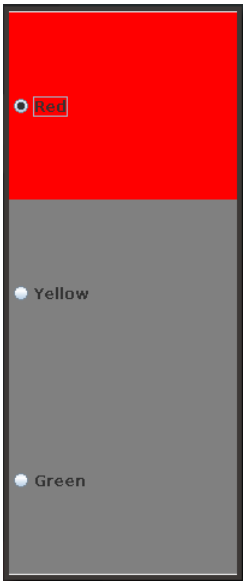
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class TrafficLights{
   TrafficLights(){
      JFrame jfrm = new JFrame();
      jfrm.setLayout(new GridLayout(3,1));
```

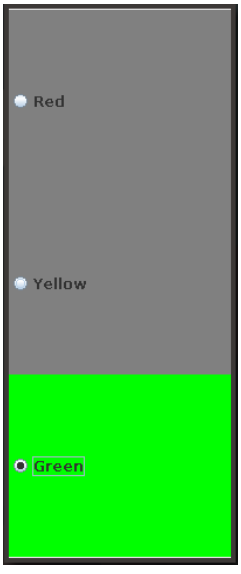
```
    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    jfrm.setSize(200,500);
    JRadioButton signalButtons[] = new JRadioButton[3];
    ButtonGroup bg = new ButtonGroup();

    signalButtons[0]=new JRadioButton("Red");
    signalButtons[0].setBackground(Color.GRAY);;

    signalButtons[1]=new JRadioButton("Yellow");
    signalButtons[1].setBackground(Color.GRAY);;

    signalButtons[2]=new JRadioButton("Green");
    signalButtons[2].setBackground(Color.GRAY);;

    bg.add(signalButtons[0]);
    bg.add(signalButtons[1]);
    bg.add(signalButtons[2]);

    jfrm.add(signalButtons[0]);
    jfrm.add(signalButtons[1]);
    jfrm.add(signalButtons[2]);

    signalButtons[0].addActionListener(new ActionListener(){
       public void actionPerformed(ActionEvent e){
          signalButtons[0].setBackground(Color.RED);
          signalButtons[1].setBackground(Color.GRAY);;
          signalButtons[2].setBackground(Color.GRAY);;
       }
    });

    signalButtons[1].addActionListener(new ActionListener(){
       public void actionPerformed(ActionEvent e){
          signalButtons[0].setBackground(Color.GRAY);
          signalButtons[1].setBackground(Color.YELLOW);;
          signalButtons[2].setBackground(Color.GRAY);;
       }
    });

    signalButtons[2].addActionListener(new ActionListener(){
       public void actionPerformed(ActionEvent e){
          signalButtons[0].setBackground(Color.GRAY);
          signalButtons[1].setBackground(Color.GRAY);;
          signalButtons[2].setBackground(Color.GREEN);;
       }
    });
    jfrm.setVisible(true);

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

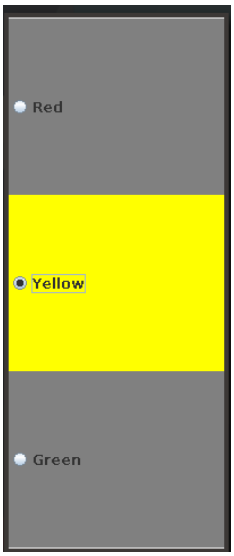
```
        new TrafficLights();
    }

}
```

Sample output









Experiment 18

Quick Sort

Date of Submission 13-01-2021

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

Concepts Used: Arrays, Quick Sorting

Algorithm Partition(arr, start,pivot)

Steps:
Start
i = start
j = start-1
 while i<pivot do
	if arr[i]<arr[pivot]then
		j++
		swap(arr[i],arr[j])
	endif
	i++
endwhile
j = j+1
if(j!=pivot)
	swap(arr[pivot],arr[j])
endif
stop

Algorithm QuickSort(arr,start,end)

Steps:
Start
if start<end
	p = Partition(arr,start,end)
	Quicksort(arr,start,pivot-1)
	Quicksort(arr,pivot+1,end)
endif

Program code:

```
import java.util.Scanner;

class QuickSort{
    public static void quickSort(String arr[], int s,int e){
        if(s<e){
            int q = partition(arr,s,e);
            quickSort(arr,s,q-1);
            quickSort(arr,q+1,e);
        }
    }

    static int partition(String arr[], int s, int pivot){
        String x = arr[pivot];
        int i=s-1,j=s;
        String temp;
        for(;j<pivot;j++)
        {
            if(arr[j].compareTo(x)<=0){ //arr[j] <= arr[pivot] the switch
                i++;
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        temp = arr[i+1];
        arr[i+1]=arr[pivot];
        arr[pivot] = temp;
        return i+1;
    }

    public static void main(String args[]){
        String[] arr = new String[100];
        int i = 0;

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of elements: ");
        int n = sc.nextInt();
        sc.nextLine();
        System.out.print("\nEnter the names to be sorted: ");

        while(i<n && sc.hasNextLine()){
                arr[i++] = sc.nextLine();
        }

        quickSort(arr,0,i-1);
        System.out.print("The sorted array is : ");
        for(int x=0;x<i;x++){
            System.out.print(arr[x]+" ");
```

```
        }
        System.out.println("");
    }
}
```

Program output:

```
..rograming/Java/CSL203/LAB 10❯ javac QuickSort.java
..rograming/Java/CSL203/LAB 10❯ java QuickSort
Enter the number of elements: 4

Enter the names to be sorted: Rahul
Ravi
Revathy
Rohini
The sorted array is : Rahul Ravi Revathy Rohini
..rograming/Java/CSL203/LAB 10❯ java QuickSort
Enter the number of elements: 2

Enter the names to be sorted: Hector
Eduardo
The sorted array is : Eduardo Hector
..rograming/Java/CSL203/LAB 10❯ java QuickSort
Enter the number of elements: 5

Enter the names to be sorted: Saul
Walter
Skylar
Hank
Hamlin
The sorted array is : Hamlin Hank Saul Skylar Walter
..rograming/Java/CSL203/LAB 10❯
```