JAVA LAB RECORD

TOTAL NO. OF EXPERIMENTS - 18

AMAL NATH M

R3 11

TKM19CS011

Experiment No. 1

26-08-2020

Aim :- Write a JAVA program to print the Fibonacci list upto a given number.

Concepts used :- Class, Logic for Fibonacci.

Algorithm:-

begin Fibonacci

int n=100; t1=0, t2=1

while t1<=n do

print t1

int sum=t1+t2

t1=t2

t2=sum

end while

end Fibonacci

Program code :-

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

		int n=100, t1=0, t2=1;

		System.out.print("Upto "+n+": ");

		while(t1<=n)
		{
			System.out.print(t1+" ");

			int sum=t1+t2;
			t1=t2;
			t2=sum;
		}
	}
}
```

Sample output :-

1. n = 100

- Upto 100: 0 1 1 2 3 5 8 13 21 34 55 89

2. n = 1

- Upto 1: 0 1 1

3. n = 0

- Upto 0: 0

Result :- Fibonacci list upto the given number is printed.

****

Experiment No. 2

26-08-2020

Aim :- Write a JAVA program to display the transpose of a given matrix.

Concepts used :- Class, 2D-array, Logic for transpose.

Algorithm:-

```
begin Transpose
    int row=3, column=3
    int matrix[][] = {{1,2,3},{4,5,6},{7,8,9}}
    int transpose[column][row]
    for i=0 till row do
        for j=0 till column do
            transpose[j][i] = matrix[i][j]
        end for
    end for
    for j=0 till column do
        for i=0 till row do
            print transpose[j][i]
        end for
    end for
end Transpose
```

Program code :-

```
public class Transpose
{
    public static void main(String[] args)
    {
        int row=3, column=3;
        int matrix[][]={{1,2,3},{4,5,6},{7,8,9}};

        //transpose the matrix
        int transpose[][]=new int[column][row];
        for(int i=0;i<row;i++)
            for(int j=0;j<column;j++)
                transpose[j][i]=matrix[i][j];
        //display the transpose
        System.out.println("The transpose is: ");
        for(int j=0;j<column;j++)
```

```
                {
                        for(int i=0;i<row;i++)
                                System.out.print(transpose[j][i]+" ");
                        System.out.println();
                }
        }
}
```

Sample output :-

1. row=3, column=3, matrix[][] = {{1,2,3},{4,5,6},{7,8,9}}

- The transpose is:

  1 4 7

  2 5 8

  3 6 9

2. row=2, column=2, matrix[][] = {{1,2},{3,4}}

- The transpose is:

  1 3

  2 4

3. row=1, column=3, matrix[][] = {{1,2,3}}

- The transpose is:

  1

  2

  3

4. row=3, column=1, matrix[][] = {{1},{2},{3}}

- The transpose is:

  1 2 3

5. row=1, column=1, matrix[][] = {{1}}

- The transpose is:

  1

Result :- Transpose of the given matrix is printed.

****

Experiment No. 3

26-08-2020

Aim :- Write a JAVA program to find the frequency of a given character in a given string.

Concepts used :- Class, String, String length, Accessing characters in a string.

Algorithm:-

begin CharFreq

String str=”hello world !”

char ch=‘l’

int count=0

for i=0 till string_length(str) do

    if str[i] == ch then

        count++

    end if

end for

print count

end CharFreq

Program code :-

```
public class CharFreq
{
    public static void main(String[] args)
    {
        String str = "hello world !";
        char ch = 'l';
        int count = 0;

        for(int i=0; i<str.length(); i++)
            if(str.charAt(i) == ch)
                count++;

        System.out.println("Frequency of '"+ch+"' in \""+str+"\" is "+count);
    }
}
```

Sample output :-

1. str = “hello world !”, ch = ‘l’

- Frequency of ‘l’ in “hello world !” is 3

2. str = “hello world !”, ch = ‘o’

- Frequency of ‘o’ in “hello world !” is 2

3. str = “hello world !”, ch = ‘ ’

- Frequency of ‘ ’ in “hello world !” is 2

4. str = “hello world !”, ch = ‘!’

- Frequency of ‘!’ in “hello world !” is 1

5. str = “hello world 007”, ch = ‘0’

- Frequency of ‘0’ in “hello world 007” is 2

6. str = “Hi, hello world”, ch = ‘H’

- Frequency of ‘H’ in “Hi, hello world” is 1

7. str = “Hi, hello world”, ch1 = ‘h’, ch2 = ‘H’

[Change in code: *if(str.charAt(i) == ch1 || str.CharAt(i) == ch2) { ... }*]

- Frequency of ‘h’ in “Hi, hello world” is 2

Result :- The frequency of a given character in a given string is printed.

****

Experiment No. 4

15/09/2020

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.

Concepts used – Constructor chaining.

Algorithms -

Algorithm deposit(x)

1 accBalance = accBalance + x

Algorithm withdraw(x)

1 accBalance = accBalance – x

Algorithm display()

1 Print accName

2 Print accBalance

3 Print accType

4 Print accNo

Algorithm Bank() //default constructor

1 accName = NIL

2 accBalance = 0

3 accType = NIL

4 accNo = 0

Algorithm Bank(a, b, c, d) //parameterized constructor

1 accName = a

2 accBalance = b

3 accType = c

4 accNo = d

Program code -

```
class Bank
{
	//data

	String accName;
	int accNo;
	String accType;
	double accBalance;

	//methods

	void deposit(double x)
	{
		System.out.println("Amount deposited = "+x+"\n");
		accBalance += x;
		display();
	}

	void withdraw(double x)
	{
		System.out.println("Amount withdrawn = "+x+"\n");
		accBalance -= x;
		display();
	}

	void display()
	{
		System.out.println("Account User = "+accName+"\nAccount Balance =
"+accBalance+"\nAccount type = "+accType+"\nAccount No = "+accNo+"\n");
	}

	void disAmount(double x)
	{
		System.out.println("Amount entered = "+x+"\n");
	}

	//constructors

	Bank(String s, String t, int n, double b) //parameterized
	{
		System.out.println("\nHello User !"); //executed when object is created and
constructor is called
		accName = s;
		accType = t;
		accNo = n;
		accBalance = b;
		display();
```

```
	}

	Bank() //default
	{
		System.out.println("\nHello User !");
		display();
	}

	public static void main(String args[])
	{
		boolean flag;
		double Amount;

		Bank acc1 = new Bank(); //First user is null of account type null with initial balance
0 and account no 0
		Amount = 1000.00; //Amount to be deposited
		flag = true; //Since deposit
		acc1.disAmount(Amount);
		if(flag)
			acc1.deposit(Amount);
		else
			if(acc1.accBalance<Amount)
				System.out.println("Insufficient balance !");
			else
				acc1.withdraw(Amount);

		Bank acc2 = new Bank("Amal", "Savings", 12345, 2500);
		Amount = 500; //Amount to be withdrawn
		flag = false; //Since withdraw
		acc2.disAmount(Amount);
		if(flag)
			acc2.deposit(Amount);
		else
			if(acc2.accBalance<Amount)
				System.out.println("Insufficient balance !");
			else
				acc2.withdraw(Amount);
	}
}
```

Output -

```
Hello User !
Account User = null
Account Balance = 0.0
Account type = null
Account No = 0

Amount entered = 1000.0

Amount deposited = 1000.0

Account User = null
```

Account Balance = 1000.0
Account type = null
Account No = 0

Hello User !
Account User = Amal
Account Balance = 2500.0
Account type = Savings
Account No = 12345

Amount entered = 500.0

Amount withdrawn = 500.0

Account User = Amal
Account Balance = 2000.0
Account type = Savings
Account No = 12345

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

****

Experiment No.5

15/09/2020

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 – Inhertitance, 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
3. Define printSpl method

Class Manager inherits Class Employee

1. Declare additional field department
2. Define constructors for with and without department
3. Define printDept method

Program code -

```
class Employee
{
	String name;
	int age;
	long phno;
	String address;
	double salary;

	void printSalary()
	{
		System.out.println("Salary = "+salary);
	}

	void printDetails()
	{
```

```
		System.out.println("\nName = "+name+"\nAge = "+age+"\nPhone No. = "+phno+"\
nAddress = "+address+"\nSalary = "+salary);
	}

	public static void main(String args[])
	{
		System.out.println("Employee Details :-");
		Officer officer1 = new Officer("Ajith", 19, 8943234482l, "Ajith House, Wayanad",
75325.50);
		Manager manager1 = new Manager("Dinoy", 19, 7501234657l, "Dinoy House,
Kozhikode", 81234.75, "Accounts");
	}
}

class Officer extends Employee
{
	String spl;

	Officer(String name, int age, long phno, String address, double salary, String spl)
	{
		this.name = name;
		this.age = age;
		this.phno = phno;
		this.address = address;
		this.salary = salary;
		this.spl = spl;

		this.printDetails();
		this.printSpl();
	}

	Officer(String name, int age, long phno, String address, double salary) //Since specialization
is optional data
	{
		this.name = name;
		this.age = age;
		this.phno = phno;
		this.address = address;
		this.salary = salary;

		this.printDetails();
	}

	void printSpl()
	{
		System.out.println("Specialization = "+spl);
	}
}

class Manager extends Employee
{
	String dept;
```

```
	Manager(String name, int age, long phno, String address, double salary, String dept)
	{
		this.name = name;
		this.age = age;
		this.phno = phno;
		this.address = address;
		this.salary = salary;
		this.dept = dept;

		this.printDetails();
		this.printDept();
	}

	Manager(String name, int age, long phno, String address, double salary) //Since department is optional data
	{
		this.name = name;
		this.age = age;
		this.phno = phno;
		this.address = address;
		this.salary = salary;

		this.printDetails();
	}

	void printDept()
	{
		System.out.println("Department = "+dept);
	}
}
```

Output -

```
Employee Details :-

Name = Ajith
Age = 19
Phone No. = 8943234482
Address = Ajith House, Wayanad
Salary = 75325.5

Name = Dinoy
Age = 19
Phone No. = 7501234657
Address = Dinoy House, Kozhikode
Salary = 81234.75
Department = Accounts
```

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

****

Experiment No. 6

22/09/2020

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). ● 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 -

Inheritance, static keyword.

ALGORITHM -

Class Employee

1. Declare fields salary and className
2. Define static 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 CODE -

```
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 engg1 = new Engineer();

                engg1.display("Engineer");
                engg1.calcSalary(20000);
        }
}
```

OUTPUT -

```
Name of class is Employee
Salary of employee is 10000.0
Name of class is Engineer
Salary of employee is 20000.0
```

RESULT -

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

****

Experiment No. 7

22/09/2020

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 -

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 CODE -

```
abstract class Shape
{
	abstract int numberOfSides();
}

class Triangle extends Shape
{
	int numberOfSides()
	{
		return 3;
	}
}

class Rectangle extends Shape
```

```
{
	int numberOfSides()
	{
		return 4;
	}
}

class Hexagon extends Shape
{
	int numberOfSides()
	{
		return 6;
	}
}

class NoOfSides
{
	public static void main(String... args)
	{
		Triangle tri1 = new Triangle();
		Rectangle rect1 = new Rectangle();
		Hexagon hex1 = new Hexagon();

		System.out.println("No. of sides in Triangle = "+tri1.numberOfSides());
		System.out.println("No. of sides in Rectangle = "+rect1.numberOfSides());
		System.out.println("No. of sides in Hexagon = "+hex1.numberOfSides());
	}
}
```

OUTPUT -

```
No. of sides in Triangle = 3
No. of sides in Rectangle = 4
No. of sides in Hexagon = 6
```

RESULT -

Data abstraction and method overriding are performed on different shapes.

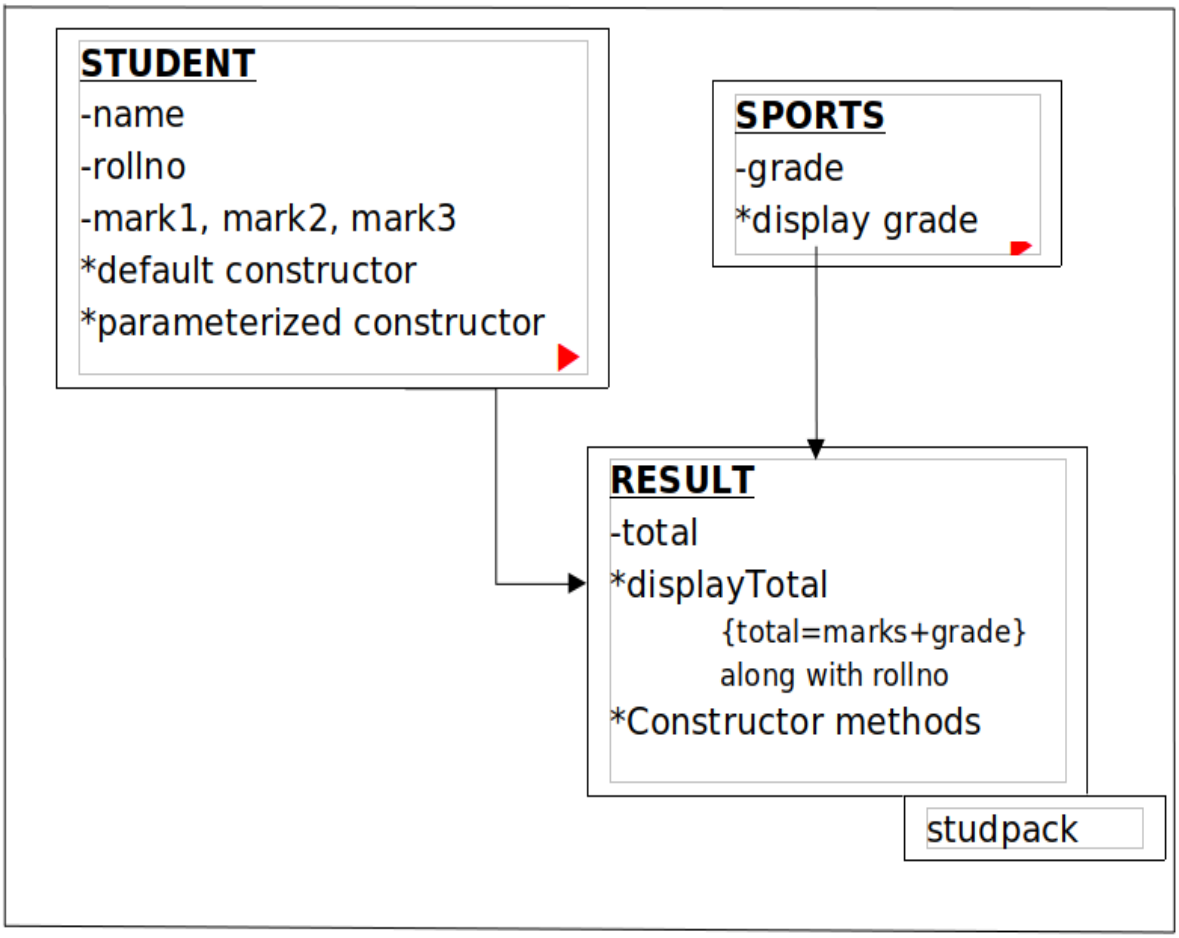
****

Experiment No. 8

26/09/2020

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 -

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 CODE -

PACKAGE -

```
package studpack;

interface Sports
{
	int grade = 100;

	void displayGrade();
}

class Student
{
	String name;
	int rollNo;
	float mark1, mark2, mark3;

	Student()
	{
		name = "NONE";
		rollNo = 0;
		mark1 = mark1 = mark3 = 0;
	}

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

public class Result extends Student implements Sports
{
	float total;

	public void displayGrade()
```

```
	{
		System.out.println("Sports Grade = "+grade);
	}

	public void displayTotal()
	{
		System.out.println("Roll No = "+rollNo);
		displayGrade();
		System.out.println("Total obtained = "+total+"/400\n");
	}

	public Result()
	{
		super();
		total = 0;
	}

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

		total = mark1 + mark2 + mark3 + grade;
	}
}
```

IMPORTING PACKAGE -

```
import studpack.*;

class Hybrid
{
	public static void main(String arg[])
	{
		Result st1 = new Result("Abhiram", 04, 88.75f, 91, 95);
		Result st2 = new Result("Ajith", 07, 89, 90.5f, 96);
		Result st3 = new Result("Emil", 22, 90, 92.5f, 99);
		Result st4 = new Result("Dinoy", 21, 92.5f, 89, 98);
		Result st5 = new Result("Jijo", 29, 90, 90.5f, 97.5f);

		st1.displayTotal();
		st2.displayTotal();
		st3.displayTotal();
		st4.displayTotal();
		st5.displayTotal();
	}
}
```

OUTPUT -

```
Roll No = 4
Sports Grade = 100
```

Total obtained = 374.75/400

Roll No = 7
Sports Grade = 100
Total obtained = 375.5/400

Roll No = 22
Sports Grade = 100
Total obtained = 381.5/400

Roll No = 21
Sports Grade = 100
Total obtained = 379.5/400

Roll No = 29
Sports Grade = 100
Total obtained = 378.0/400

RESULT -

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

****

Experiment No. 9

15/10/2020

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, parseInt() method.

ALGORITHM -

Algorithm tryFinally(index)

1. try block – Print arr[index]
2. catch block – Print ArrayIndexOutOfBoundsException object
3. finally block – Print arr[0] till arr[4]

Algorithm multipleCatch(index)

1. try block – Print arr[index], b[index] and a[index]/a[index]
2. catch block 1 – Print IndexOutOfBoundsException object
3. catch block 2 – Print ArithmeticException object
4. catch block 3 – Print RuntimeException object

Algorithm nestedTry(str, index)

```
1  try –
2          Integer.parseInt(str)
3          try –
4                  Print arr[index]
5                  try –
6                          obj = null
7                          Print obj.arr[index]
8                  catch – Print NullPointerException object
9          catch – Print IndexOutOfBoundsException object
```

10 catch – Print NumberFormatException object

Algorithm throwEg(input[])

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 throwsEg

1 throws FileNotFoundException, IOException

2 Read any file

PROGRAM CODE -

```
import java.io.*;

class Except
{
	static int a[] = {0,1,2,3,4};
	static String b = "hello";
	int i = 0;

	public static void main(String args[]) throws IOException
	{
		tryFinally(5);
		multipleCatch(0);
		nestedTry("1234",1);
		throwExample(args);
		throwsExample(3);
	}


	static void tryFinally(int index)
	{
		System.out.println("Try-Finally example :");

		try {

			System.out.println(a[index]);

		} catch(ArrayIndexOutOfBoundsException e) {

			System.out.println("Exception thrown : "+e);

		} finally {
```

```
                System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]);

        }
}

static void multipleCatch(int index)
{
        System.out.println("\nMultiple catch example :");

        try {

                System.out.println(b.charAt(index));
                System.out.println(a[index]);
                System.out.println(a[index]/a[index]);

        } catch(ArithmeticException e) {

                System.out.println("Exception thrown : "+e);

        } catch(IndexOutOfBoundsException e) {

                System.out.println("Exception thrown : "+e);

        } catch(RuntimeException e) {

                System.out.println("Exception thrown : "+e);

        }
}

static void nestedTry(String str, int index)
{
        System.out.println("\nNested Try example :");

        try {
                int j = Integer.parseInt(str);
                System.out.println(j);

                try {
                        System.out.println(str.charAt(index));
                        System.out.println(a[index]);

                        try {

                                Except obj = new Except();
                                if(index == 0)
                                        obj = null;

                                System.out.println(obj.i);

                        } catch(NullPointerException e) {
```

```
                                System.out.println("Exception thrown : "+e);

                        }

                } catch(IndexOutOfBoundsException e) {

                        System.out.println("Exception thrown : "+e);

                }

        } catch (NumberFormatException e) {

                System.out.println("Exception thrown : "+e);

        }
}

        static void throwExample(String input[])
        {
                System.out.println("\nThrow example :");

                if(input.length != 2)
                {
                        throw new RuntimeException("Input 2 numbers at the command-line !");
                }

                else
                {
                        try {

                        int x = Integer.parseInt(input[0]);
                        int y = Integer.parseInt(input[1]);

                        System.out.println(x+y);

                        } catch(NumberFormatException e){

                        System.out.println("Input 2 numbers !");
                        throw e;

                        }
                }
        }

        static void throwsExample(int index) throws FileNotFoundException,
ArrayIndexOutOfBoundsException
        {
                System.out.println("\nThrows example :");

                System.out.println(a[index]);

                File file = new File ("/home/amal/javalab/Hello.txt");
```

```
                FileReader fr = new FileReader(file);
        }
}
```

SAMPLE OUTPUTS -

1.) java Except 12 24

```
Try-Finally example :
Exception thrown : java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length
5
0 1 2 3 4

Multiple catch example :
h
0
Exception thrown : java.lang.ArithmeticException: / by zero

Nested Try example :
1234
2
1
0

Throw example :
36

Throws example :
3
Exception in thread "main" java.io.FileNotFoundException: /home/amal/javalab/Hello.txt (No such
file or directory)
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
        at java.base/java.io.FileReader.<init>(FileReader.java:75)
        at Except.throwsExample(Except.java:140)
        at Except.main(Except.java:15)
```

2.) java Except 12

```
Try-Finally example :
Exception thrown : java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length
5
0 1 2 3 4

Multiple catch example :
h
0
Exception thrown : java.lang.ArithmeticException: / by zero
```

```
Nested Try example :
1234
2
1
0

Throw example :
Exception in thread "main" java.lang.RuntimeException: Input 2 numbers at the command-line !
        at Except.throwExample(Except.java:108)
        at Except.main(Except.java:14)
```

3.) java Except ab abc

```
Try-Finally example :
Exception thrown : java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length
5
0 1 2 3 4

Multiple catch example :
h
0
Exception thrown : java.lang.ArithmeticException: / by zero

Nested Try example :
1234
2
1
0

Throw example :
Input 2 numbers !
Exception in thread "main" java.lang.NumberFormatException: For input string: "ab"
        at
java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.base/java.lang.Integer.parseInt(Integer.java:652)
        at java.base/java.lang.Integer.parseInt(Integer.java:770)
        at Except.throwExample(Except.java:116)
        at Except.main(Except.java:14)
```

4.) java Except

```
Try-Finally example :
Exception thrown : java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length
5
0 1 2 3 4

Multiple catch example :
h
0
Exception thrown : java.lang.ArithmeticException: / by zero
```

Nested Try example :
1234
2
1
0

Throw example :
Exception in thread "main" java.lang.RuntimeException: Input 2 numbers at the command-line !
	at Except.throwExample(Except.java:108)
	at Except.main(Except.java:14)

5.) java Except 12 36

```
//tryFinally(2);

//multipleCatch(1);

//nestedTry("1234",1);

//throwExample(args);

//throwsExample(3);
```

Try-Finally example :
2
0 1 2 3 4

Multiple catch example :
e
1
1

Nested Try example :
1234
2
1
0

Throw example :
48

Throws example :
3
Exception in thread "main" java.io.FileNotFoundException: /home/amal/javalab/Hello.txt (No such file or directory)
	at java.base/java.io.FileInputStream.open0(Native Method)
	at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
	at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
	at java.base/java.io.FileReader.<init>(FileReader.java:75)
	at Except.throwsExample(Except.java:140)
	at Except.main(Except.java:15)

6.) java Except 12 36

```
//tryFinally(2);

//multipleCatch(1);

//nestedTry("1234",5);

//throwExample(args);

//throwsExample(3);
```

```
Try-Finally example :
2
0 1 2 3 4

Multiple catch example :
e
1
1

Nested Try example :
1234
Exception thrown : java.lang.StringIndexOutOfBoundsException: String index out of range: 5

Throw example :
48

Throws example :
3
Exception in thread "main" java.io.FileNotFoundException: /home/amal/javalab/Hello.txt (No such
file or directory)
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
        at java.base/java.io.FileReader.<init>(FileReader.java:75)
        at Except.throwsExample(Except.java:140)
        at Except.main(Except.java:15)
```

7.) java Except 12 36

```
//tryFinally(2);

//multipleCatch(1);

//nestedTry("1234",0);

//throwExample(args);

//throwsExample(3);
```

Try-Finally example :
2
0 1 2 3 4

Multiple catch example :
e
1
1

Nested Try example :
1234
1
0
Exception thrown : java.lang.NullPointerException

Throw example :
48

Throws example :
3
Exception in thread "main" java.io.FileNotFoundException: /home/amal/javalab/Hello.txt (No such file or directory)
	at java.base/java.io.FileInputStream.open0(Native Method)
	at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
	at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
	at java.base/java.io.FileReader.<init>(FileReader.java:75)
	at Except.throwsExample(Except.java:140)
	at Except.main(Except.java:15)

8.) java Except 12 36

//tryFinally(2);

//multipleCatch(1);

//nestedTry("abc",1);

//throwExample(args);

//throwsExample(3);

Try-Finally example :
2
0 1 2 3 4

Multiple catch example :
e
1
1

Nested Try example :
Exception thrown : java.lang.NumberFormatException: For input string: "abc"

Throw example :
48

Throws example :
3
Exception in thread "main" java.io.FileNotFoundException: /home/amal/javalab/Hello.txt (No such file or directory)
	at java.base/java.io.FileInputStream.open0(Native Method)
	at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
	at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
	at java.base/java.io.FileReader.<init>(FileReader.java:75)
	at Except.throwsExample(Except.java:140)
	at Except.main(Except.java:15)

9.) java Except 12 36

```
//tryFinally(2);

//multipleCatch(3);

//nestedTry("1234",1);

//throwExample(args);

//throwsExample(5);
```

Try-Finally example :
2
0 1 2 3 4

Multiple catch example :
l
3
1

Nested Try example :
1234
2
1
0

Throw example :
48

Throws example :
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
	at Except.throwsExample(Except.java:136)
	at Except.main(Except.java:15)

10.) java Except 12 36

```
//tryFinally(2);

//multipleCatch(3);

//nestedTry("1234",1);

//throwExample(args);

//throwsExample(0);

//Hello.txt file exists
```

```
Try-Finally example :
2
0 1 2 3 4

Multiple catch example :
l
3
1

Nested Try example :
1234
2
1
0

Throw example :
48

Throws example :
0
```

RESULT -

Exception handling is carried out in a Java program.

****

Experiment No. 10

21/10/2020

AIM -

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

CONCEPTS USED -

Input/Output classes.

ALGORITHM -

Algorithm FileHandling

1 import java.io classes
2 throws IOException
3 Initialise FileInputStream object fin on the input file
4 Initialise FileOutputStream object fout on the output file
5 try
6 while((i = fin.read()) != -1)
7 fout.write(i)
8 endwhile
9 catch
10 Print IOException object
11 endtry

PROGRAM CODE -

```
import java.io.*;

class FileHandling
{

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

		try
		{
			FileInputStream fin = new FileInputStream("inputfile.txt");
```

```
                FileOutputStream fout = new FileOutputStream("outputfile.txt");

                int i;

                while((i = fin.read()) != -1)
                {
                        fout.write(i);
                }
        }
        catch (IOException e)
        {
                System.out.println("Exception : " + e);
        }
    }
}
```

OUTPUT -

//inputfile.txt file in the same directory :
Heya !
Hello guys, how are you ?

//outputfile.txt file in the same directory :
Heya !
Hello guys, how are you ?

RESULT -

File handling is carried out in a Java program.

****

Experiment No. 11

21/10/2020

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 -

Scanner class, StringTokenizer class.

ALGORITHM -

Algorithm StringTokens

1 import java.util.Scanner
2 sum = 0
3 Initialise Scanner object in on System.in
4 s = in.nextLine() //read the numbers separated by a space
5 Initialise StringTokenizer object str on s
6 while(str.hasMoreTokens())
7 num = Integer.parseInt(str.nextToken())
8 Print num
9 sum += num
10 endwhile
11 Print sum

PROGRAM CODE -

```
import java.util.*;

class StringTokens
{
	public static void main(String args[])
	{
		int sum = 0;

		System.out.println("Enter the integers to be added separated by a space!");
		Scanner in = new Scanner(System.in);
```

```
                String s = in.nextLine();

                StringTokenizer str = new StringTokenizer(s);

                System.out.println("\nYou entered :");

                while(str.hasMoreTokens())
                {
                        int num = Integer.parseInt(str.nextToken());
                        System.out.println(num);
                        sum += num;
                }

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

OUTPUT -

```
Enter the integers to be added separated by a space!
1 22 333 4444 55555

You entered :
1
22
333
4444
55555

Sum = 60355


Enter the integers to be added separated by a space!
1 2 5 6

You entered :
1
2
5
6

Sum = 14


Enter the integers to be added separated by a space!
12 23 34 45 56 67 78

You entered :
12
23
34
```

45
56
67
78

Sum = 315

Enter the integers to be added separated by a space!

You entered :

Sum = 0

RESULT -

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

****

Experiment No. 12

28/10/2020

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 -

Collections framework.

ALGORITHM -

1 import java.util classes

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 CODE -

```
import java.util.*;

class Dll
{
	public static void main(String args[])
	{
		LinkedList<String> list = new LinkedList<String>();

		list.add("Ajith");
		list.add(0,"Mnakshi");
		list.add("Dinoy");
		list.add("Emil");
		list.add("Jijo");

		System.out.println(list+"\n");
```

```
                list.remove(4);
                list.remove("Emil");

                Iterator itr = list.iterator();

                System.out.println("The list, after deletion, contains:");

                while(itr.hasNext())
                        System.out.println(itr.next());
        }
}
```

OUTPUT -

[Mnakshi, Ajith, Dinoy, Emil, Jijo]

The list, after deletion, contains:
Mnakshi
Ajith
Dinoy

RESULT -

A doubly linked list is implemented using Collections framework.

****

Experiment No. 13

28/10/2020

AIM -

Write a Java program that implements the binary search algorithm.

CONCEPTS USED -

Scanner, Binary Search algorithm.

ALGORITHM -

Algorithm main

1 import java.util.Scanner
2 Intilialise Scanner object in on System.in
3 Read array size, n = in.nextInt()
4 Initialise array arr of size n
5 for i=0 till n //Read the elements
6 arr[i] = in.nextInt()
7 endfor
8 Read search element, key = in.nextInt()
9 binarySearch()

Algorithm binarySearch

1 first = 0, last = arr.length – 1
2 while (first <= last)
3 mid = (first + last) / 2
4 if(key < arr[mid])
5 last = mid – 1
6 else if (key > arr[mid])
7 first = mid + 1
8 else
9 Print “Element found”
10 return

11 endif

12 endwhile

13 Print “Element not found”

PROGRAM CODE -

```
import java.util.*;

class BinarySearch
{
	static void search(int arr[], int X)
	{
		int first = 0;
		int last = arr.length-1;
		int mid = (first + last)/2;

		while(first <= last)
		{
			if(X < arr[mid])
			{
				last = mid - 1;
			}
			else if(X > arr[mid])
			{
				first = mid + 1;
			}
			else
			{
				System.out.println("Element found at index: "+mid);
				break;
			}

			mid = (first + last)/2;
		}

		if(first > last)
			System.out.println("Element not found!");
	}

	public static void main(String args[])
	{
		Scanner in = new Scanner(System.in);

		System.out.println("Enter the no. of elements");
		int n = in.nextInt();

		int arr[] = new int[n];

		System.out.println("Enter the elements in ascending order");
		for(int i=0;i<n;i++)
```

```
                arr[i] = in.nextInt();

            System.out.println("Enter the search element");
            int X = in.nextInt();

            search(arr, X);
        }
}
```

SAMPLE OUTPUTS -

```
Enter the no. of elements
3
Enter the elements in ascending order
11
22
33
Enter the search element
00
Element not found!


Enter the no. of elements
3
Enter the elements in ascending order
11
22
33
Enter the search element
22
Element found at index: 1


Enter the no. of elements
3
Enter the elements in ascending order
11
123
345
Enter the search element
11
Element found at index: 0


Enter the no. of elements
3
Enter the elements in ascending order
123
456
789
```

Enter the search element
789
Element found at index: 2

Enter the no. of elements
4
Enter the elements in ascending order
1
2
3
4
Enter the search element
1
Element found at index: 0

Enter the no. of elements
4
Enter the elements in ascending order
1
2
3
4
Enter the search element
4
Element found at index: 3

Enter the no. of elements
4
Enter the elements in ascending order
1
2
3
4
Enter the search element
2
Element found at index: 1

Enter the no. of elements
4
Enter the elements in ascending order
1
2
3
4
Enter the search element
3
Element found at index: 2

Enter the no. of elements
4
Enter the elements in ascending order
1
2
3
4
Enter the search element
5
Element not found!

RESULT -

Binary Search is performed on an array of integers.

****

Experiment No. 14

04/11/2020

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, Random generator, Math.pow() method.

ALGORITHM -

Class X implements Class Runnable

1 import java.util.Random
2 Initialise Random object rand and static field random
3 Override public void run() method
4 random = rand.nextInt(25)
5 Print random
6 End run()

Class Y implements Class Runnable

1 Override public void run() method
2 if(X.random % 2 == 0)
3 Print Math.pow(X.random, 2)
4 endif
5 End run()

Class Z implements Class Runnable

6 Override public void run() method
7 if(X.random % 2 != 0)
8 Print Math.pow(X.random, 3)
9 endif
10 End run()

Class Main

```
1   for i=0 till 10
2           Thread x = new Thread(new X())
3           Thread y = new Thread(new Y())
4           Thread z = new Thread(new Z())
5           x.sleep(1000) //Inside try-catch block
6           x.start(), y.start(), z.start()
7   endfor
```

PROGRAM CODE -

```
import java.util.Random;

class X implements Runnable
{
        static int random;
        Random rand = new Random();

        public void run()
        {
                random = rand.nextInt(25);
                System.out.println(random);
        }
}

class Y implements Runnable
{
        public void run()
        {
                if(X.random % 2 == 0)
                        System.out.println((int) Math.pow(X.random,2)+"\n");
        }
}

class Z implements Runnable
{
        public void run()
        {
                if(X.random % 2 != 0)
                        System.out.println((int) Math.pow(X.random,3)+"\n");
        }
}

class MultiThread
{
        public static void main(String args[])
        {
                for(int i=0;i<10;i++)
                {
```

```
                Thread objX = new Thread(new X());
                Thread objY = new Thread(new Y());
                Thread objZ = new Thread(new Z());
                try{
                        objX.sleep(1000);
                }catch(Exception e){}
                objX.start();
                objY.start();
                objZ.start();
            }
        }
}
```

SAMPLE OUTPUT -

```
18
324

9
729

10
100

23
12167

13
2197

16
256

10
100

12
144

9
729

2
4
```

RESULT -

Multi-threading is implemented in a Java program.

****

Experiment No. 15

04/11/2020

AIM -

Write a Java program that shows thread synchronization.

CONCEPTS USED -

Multithreading, Thread synchronization.

ALGORITHM -

Class Demo

1 synchronized display(n, t)

1 for i = n to t

2 Print i * 5

3 Thread.sleep(400)

4 enfor

5 End display()

Class X inherits Class Thread

1 Declare field obj (object of class Demo) and define a constructor to initialise obj

2 public void run()

3 obj.display(1, 5)

4 End run()

Class Y inherits Class Thread

5 Declare field obj (object of class Demo) and define a constructor to initialise obj

6 public void run()

7 obj.display(6, 10)

8 End run()

Class Main

1 Initialise Demo obj

2 X x = new X(obj)

3 Y y = new Y(obj)

4 x.start()

5 y.start()

PROGRAM CODE -

```
class Demo
{
	public synchronized void display(int n, int t)
	{

		for(int i=n;i<=t;i++)
		{
			System.out.println(i*5);
			try{
				Thread.sleep(400);
			}catch(Exception e){System.out.println(e);}
		}
	}
}

class X extends Thread
{
	Demo obj;

	X(Demo obj)
	{
		this.obj = obj;
	}

	public void run()
	{
		obj.display(1,5);
	}
}

class Y extends Thread
{
	Demo obj;

	Y(Demo obj)
	{
		this.obj = obj;
	}

	public void run()
	{
		obj.display(6,10);
	}
}
```

```
class ThreadSync
{
	public static void main(String args[])
	{
		Demo obj = new Demo();
		X objX = new X(obj);
		Y objY = new Y(obj);
		objX.start();
		objY.start();
	}
}
```

SAMPLE OUTPUTS -

//Without synchronized

```
5
30
10
35
15
40
20
45
50
25
```

//With synchronized

```
5
10
15
20
25
30
35
40
45
50
```

RESULT -

Thread synchronization is carried out in a Java program.

****

Experiment No. 16

11/11/2020

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 – 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 “Simple Calculator”
3. Initialise JPanel panel1, panel2, inputPanel
4. Set layout managers GridLayout(4,3), GridLayout(6,1), GridBagLayout(), GridBagLayout() for panel, panel1, inputPanel and frame respectively
5. frame.setBounds(825, 0, 350, 400)
6. frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)

//COMPONENTS

7. JTextField result = new JTextField()
8. JButton one = new JButton("1")
9. JButton two = new JButton("2")
10. JButton three = new JButton("3")
11. JButton four = new JButton("4")
12. JButton five = new JButton("5")
13. JButton six = new JButton("6")
14. JButton seven = new JButton("7")
15. JButton eight = new JButton("8")
16. JButton zero = new JButton("0"
17. JButton nine = new JButton("9")
18. JButton AC = new JButton("AC")
19. JButton dot = new JButton(".")
20. JButton plus = new JButton("+")
21. JButton minus = new JButton("-")
22. JButton into = new JButton("*")
23. JButton by = new JButton("/")
24. JButton mod = new JButton("%")
25. JButton equal = new JButton("=")
26. Add buttons from one to dot to panel
27. Add buttons from plus to equal to panel1
28. result.setHorizontalAlignment(JTextField.CENTER)
29. Modify UI using GridBagConstraints object c

```
frame.add(result, c)
Modify UI using c
inputPanel.add(panel, c)
Modify UI using c
inputPanel.add(panel1, c)
Modify UI using c
frame.add(inputPanel, c)
frame.setVisible(true)
panel.setVisible(true)
panel1.setVisible(true)
inputPanel.setVisible(true)
//EVENT HANDLING
//For the input buttons, the event handler is,
        result.setText(result.getText() + "1")          //Button one
//For All Cancel button, the event handler is,
        result.setText("")
//For Equals button, the event handler is,
        exp = result.getText()
        i = 1
        res = 0
        z = exp.charAt(1)
        try
                while(z != '+' && z != '-' && z != '*' && z != '/' && z != '%')
                        i++
                        z = exp.charAt(i)
                endwhile
                x = Float.parseFloat(exp.substring(0,i))
                y = Float.parseFloat(exp.substring(i+1,exp.length()))
                switch(z)
                        case + :
                                res = x+y
                                break
                        case - :
                                res = x-y
                                break
                        case * :
                                res = x*y
                                break
                        case / :
                                if(y == 0)
                                        break
                                res = x/y
                                break
                        case % :
                                res = x%y
                                break
                        endcase
                endswitch
                if(y == 0)
                        result.setText("Not defined!")
                else
                        result.setText(res+"")
```

```
78                  endif
79          catch
80                  result.setText("Enter two operands!")
81          endtry
82  STOP
```

PROGRAM CODE -

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

class Calculator
{
        Calculator()
        {
                //CONTAINERS

                JFrame frame = new JFrame("Simple Calculator");
                JPanel panel = new JPanel();
                JPanel panel1 = new JPanel();
                JPanel inputPanel = new JPanel();

                panel.setLayout(new GridLayout(4,3));
                panel1.setLayout(new GridLayout(6,1));
                inputPanel.setLayout(new GridBagLayout());
                frame.setLayout(new GridBagLayout());
                GridBagConstraints c = new GridBagConstraints();

                frame.setBounds(825,0,350,400);

                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                //COMPONENTS

                JTextField result = new JTextField();
                JButton one = new JButton("1");
                JButton two = new JButton("2");
                JButton three = new JButton("3");
                JButton four = new JButton("4");
                JButton five = new JButton("5");
                JButton six = new JButton("6");
                JButton seven = new JButton("7");
                JButton eight = new JButton("8");
                JButton zero = new JButton("0");
                JButton nine = new JButton("9");
                JButton AC = new JButton("AC");
                JButton dot = new JButton(".");
                JButton plus = new JButton("+");
                JButton minus = new JButton("-");
                JButton into = new JButton("*");
```

```
JButton by = new JButton("/");
JButton mod = new JButton("%");
JButton equal = new JButton("=");

panel.add(one);
panel.add(two);
panel.add(three);
panel.add(four);
panel.add(five);
panel.add(six);
panel.add(seven);
panel.add(eight);
panel.add(nine);
panel.add(AC);
panel.add(zero);
panel.add(dot);
panel1.add(plus);
panel1.add(minus);
panel1.add(into);
panel1.add(by);
panel1.add(mod);
panel1.add(equal);

c.gridx = 0;
c.gridy = 0;
c.fill = GridBagConstraints.BOTH;
c.weightx = 1;
c.weighty = 0.4;
result.setHorizontalAlignment(JTextField.CENTER);
frame.add(result, c);

c.gridx = 0;
c.gridy = 0;
c.weightx = 3;
c.weighty = 1;
inputPanel.add(panel, c);

c.gridx = 1;
c.weightx = 1;
inputPanel.add(panel1, c);

c.gridx = 0;
c.gridy = 1;
frame.add(inputPanel, c);

frame.setVisible(true);
panel.setVisible(true);
panel1.setVisible(true);
inputPanel.setVisible(true);

//EVENT HANDLING
```

```
		one.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent ae){

			if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
				result.setText("");
			result.setText(result.getText() + "1");
		}});
		two.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent ae){

			if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
				result.setText("");
			result.setText(result.getText() + "2");
		}});
		three.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent ae){

			if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
				result.setText("");
			result.setText(result.getText() + "3");
		}});
		four.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent ae){

			if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
				result.setText("");
			result.setText(result.getText() + "4");
		}});
		five.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent ae){

			if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
				result.setText("");
			result.setText(result.getText() + "5");
		}});
		six.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent ae){

			if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
				result.setText("");
			result.setText(result.getText() + "6");
		}});
		seven.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent ae){
```

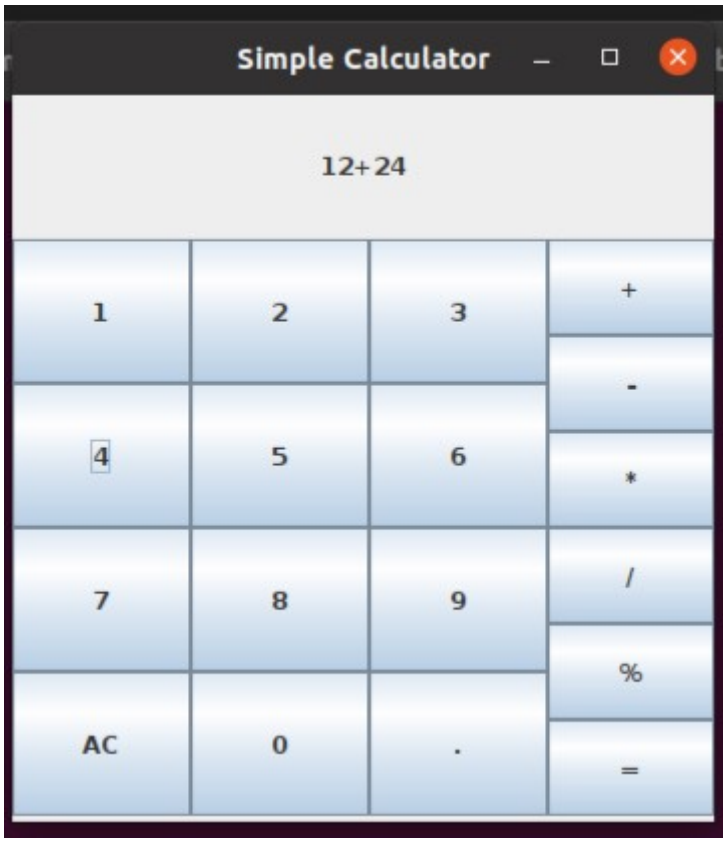
```
                        if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
                                result.setText("");
                        result.setText(result.getText() + "7");
                }});
                eight.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent ae){

                        if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
                                result.setText("");
                        result.setText(result.getText() + "8");
                }});
                nine.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent ae){

                        if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
                                result.setText("");
                        result.setText(result.getText() + "9");
                }});
                zero.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent ae){

                        if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
                                result.setText("");
                        result.setText(result.getText() + "0");
                }});
                dot.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent ae){

                        if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
                                result.setText("");
                        result.setText(result.getText() + ".");
                }});
                plus.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent ae){

                        if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
                                result.setText("");
                        result.setText(result.getText() + "+");
                }});
                minus.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent ae){

                        if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
                                result.setText("");
                        result.setText(result.getText() + "-");
```

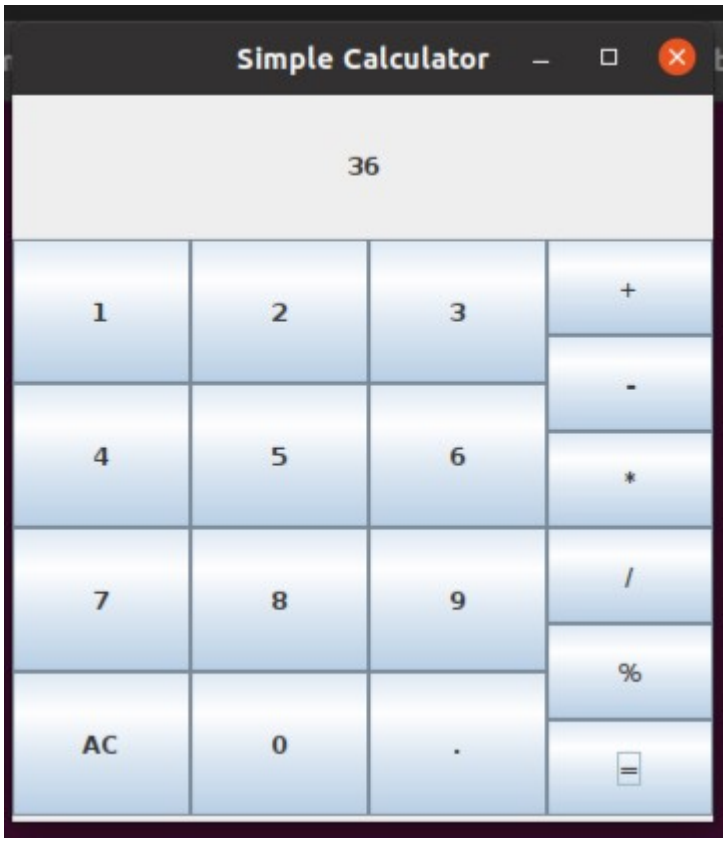
```
		});
		into.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent ae){

			if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
				result.setText("");
			result.setText(result.getText() + "*");
		});
		by.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent ae){

			if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
				result.setText("");
			result.setText(result.getText() + "/");
		});
		mod.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent ae){

			if(result.getText().equals("Enter two operands!") ||
result.getText().equals("Not defined!"))
				result.setText("");
			result.setText(result.getText() + "%");
		});
		AC.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent ae){

			result.setText("");
		});
		result.addKeyListener(new KeyAdapter() {
		public void keyPressed(KeyEvent ae) {

		if (ae.getKeyCode() == KeyEvent.VK_ENTER) {

				String exp = result.getText();
				int i = 1;
				float x, y, res = 0;
				char z = exp.charAt(1);

				try
				{
					while(z != '+' && z != '-' && z != '*' && z != '/' && z != '%')
					{
						i++;
						z = exp.charAt(i);
					}
					x = Float.parseFloat(exp.substring(0,i));
					y = Float.parseFloat(exp.substring(i+1,exp.length()));
					switch(z)
					{
						case '+':
```

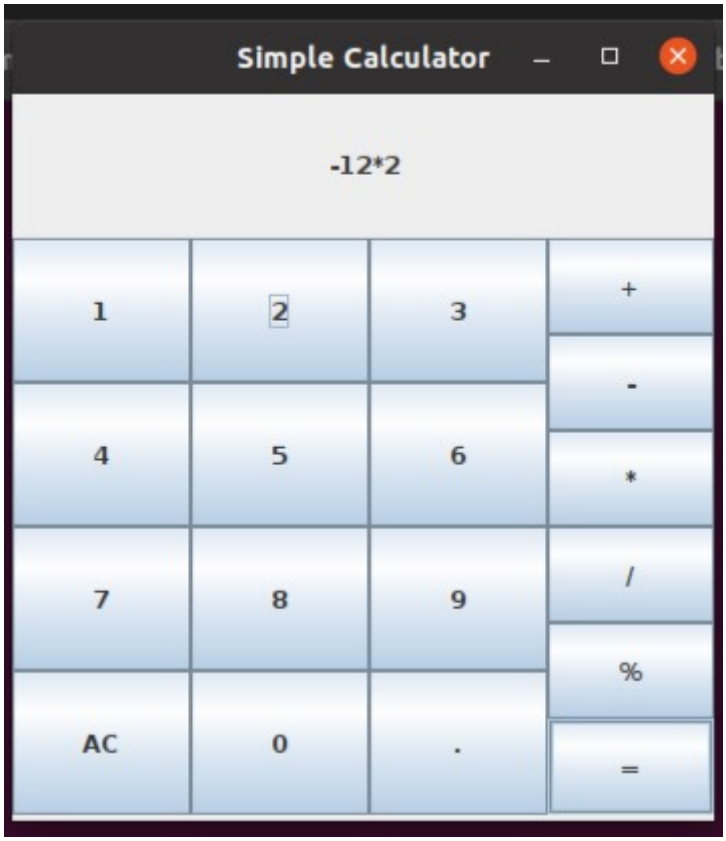
```
                                        res = x+y;
                                        break;
                                case '-':
                                        res = x-y;
                                        break;
                                case '*':
                                        res = x*y;
                                        break;
                                case '/':
                                        if(y == 0)
                                                break;
                                        res = x/y;
                                        break;
                                case '%':
                                        res = x%y;
                                        break;
                        }

                        if(y == 0)
                                result.setText("Not defined!");
                        else if(res == (int)res)
                                result.setText((int)res+"");
                        else
                                result.setText(res+"");

                }catch(Exception e){result.setText("Enter two operands!");}
        }
});
equal.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){

        String exp = result.getText();
        int i = 1;
        float x, y, res = 0;
        char z = exp.charAt(1);

        try
        {
                while(z != '+' && z != '-' && z != '*' && z != '/' && z != '%')
                {
                        i++;
                        z = exp.charAt(i);
                }
                x = Float.parseFloat(exp.substring(0,i));
                y = Float.parseFloat(exp.substring(i+1,exp.length()));
                switch(z)
                {
                        case '+':
                                res = x+y;
                                break;
                        case '-':
                                res = x-y;
```

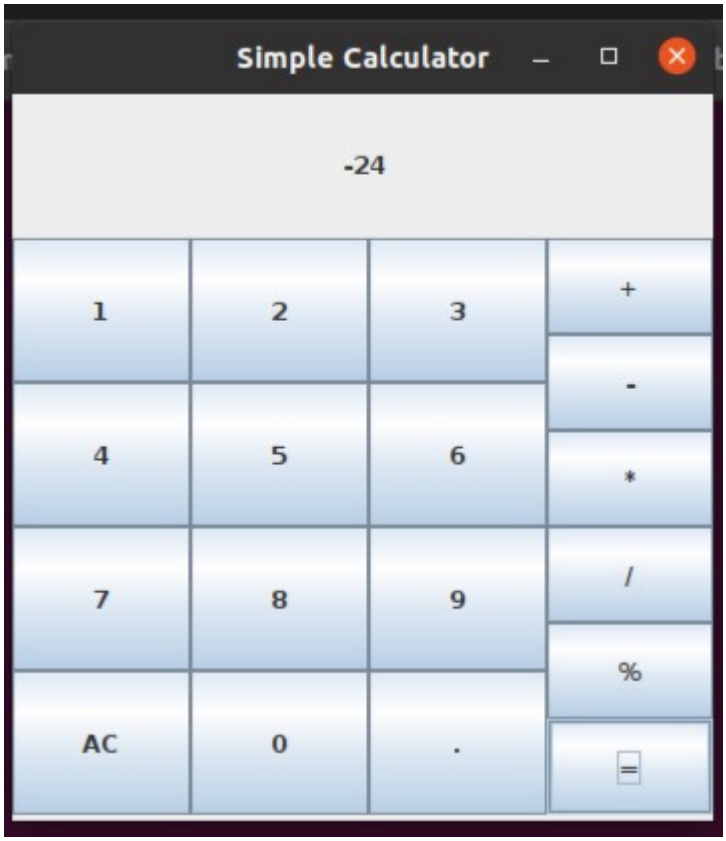
```
                                        break;
                                case '*':
                                        res = x*y;
                                        break;
                                case '/':
                                        if(y == 0)
                                                break;
                                        res = x/y;
                                        break;
                                case '%':
                                        res = x%y;
                                        break;
                                }

                                if(y == 0)
                                        result.setText("Not defined!");
                                else if(res == (int)res)
                                        result.setText((int)res+"");
                                else
                                        result.setText(res+"");

                        }catch(Exception e){result.setText("Enter two operands!");}
                });
        }

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

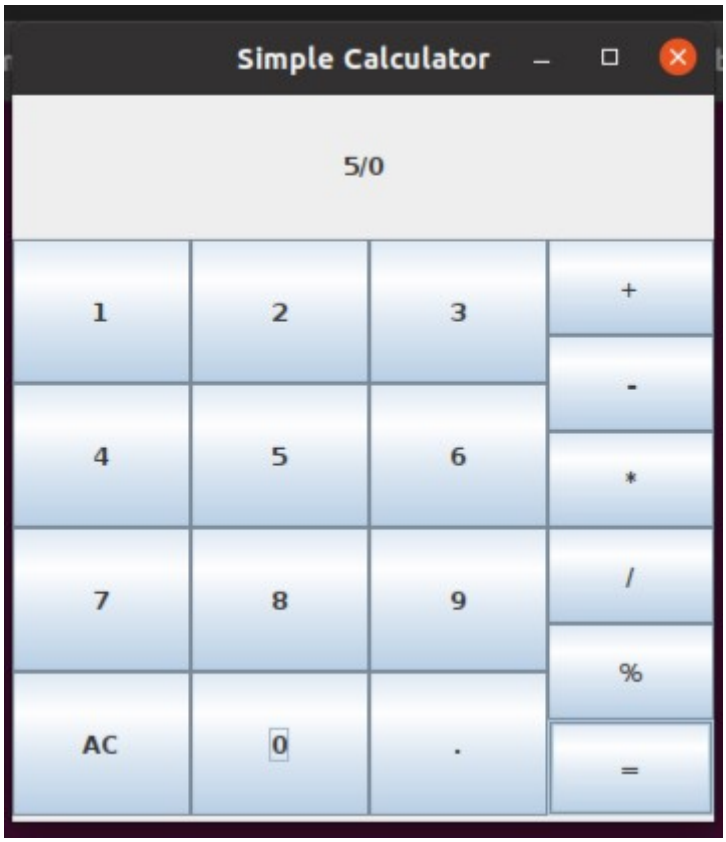
SAMPLE OUTPUTS -

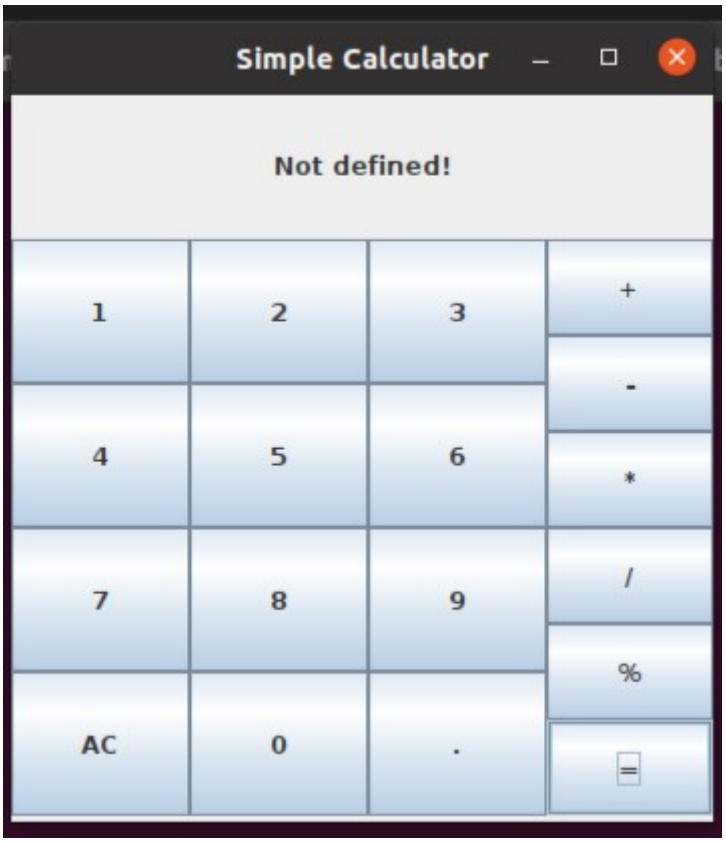


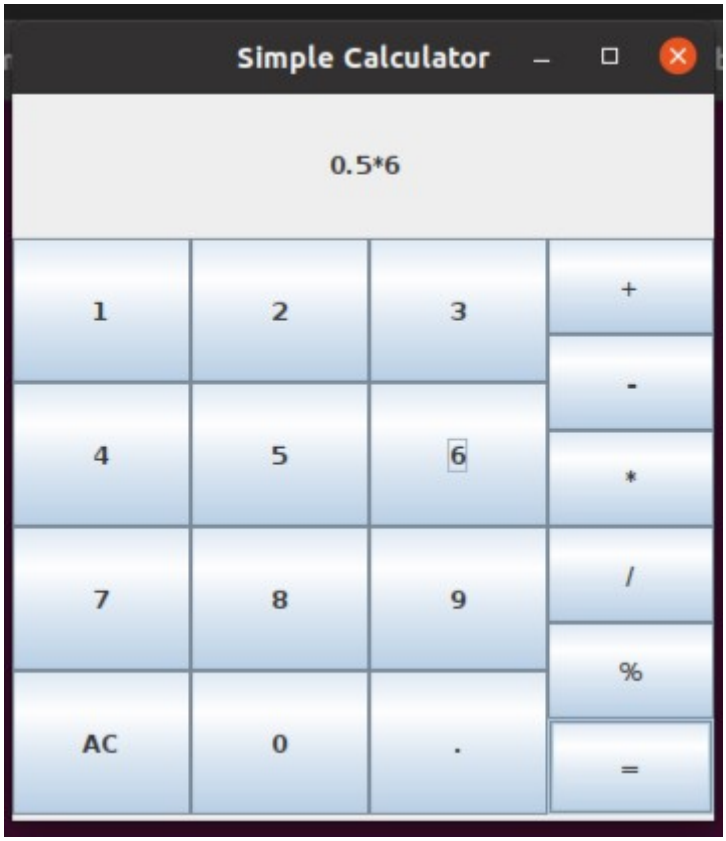


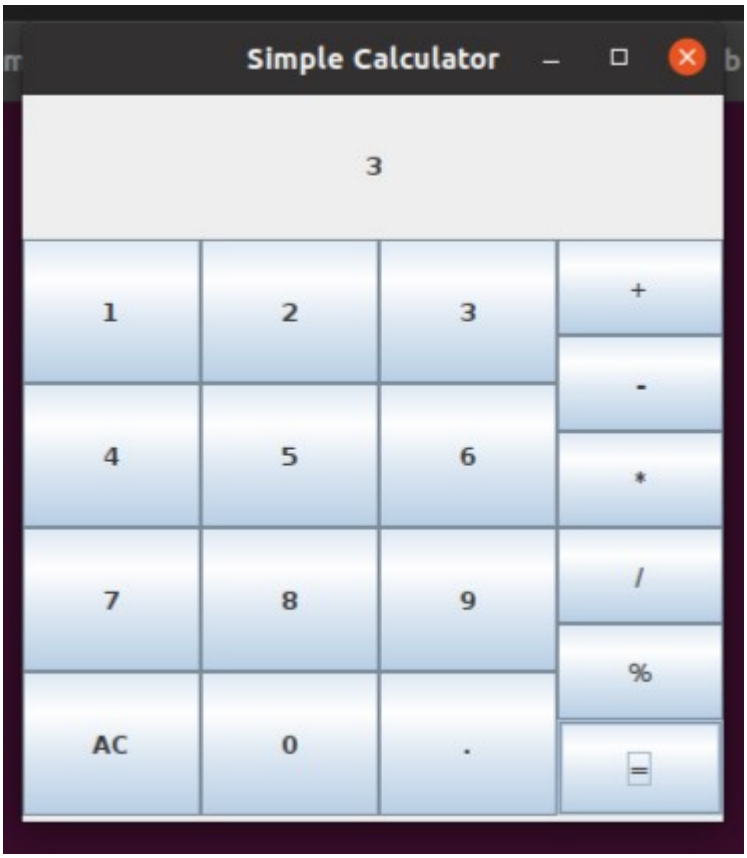

Simple Calculator
-12*2
1
2
3
+
-
4
5
6
*
7
8
9
/
%
AC
0
.
=

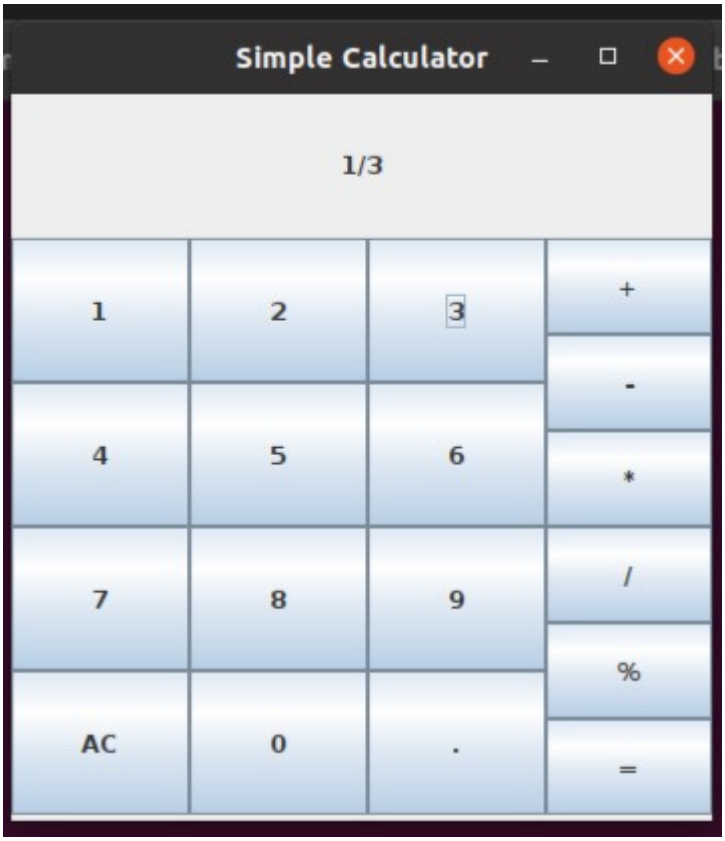

Simple Calculator
-24
1
2
3
+
-
4
5
6
*
7
8
9
/
%
AC
0
.
=

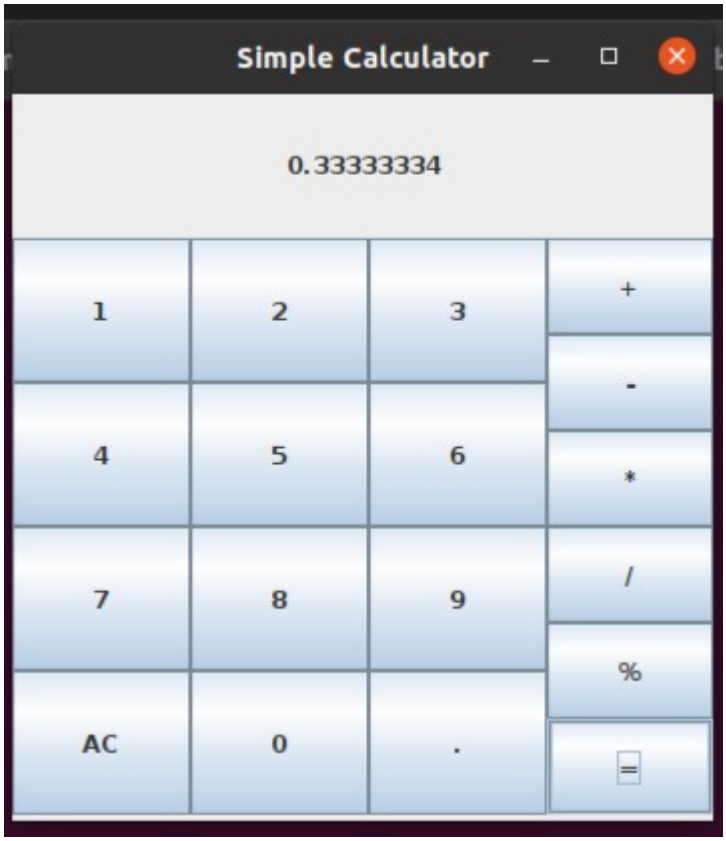

Simple Calculator
5/0
1
2
3
+
-
4
5
6
*
7
8
9
/
%
AC
0
.
=

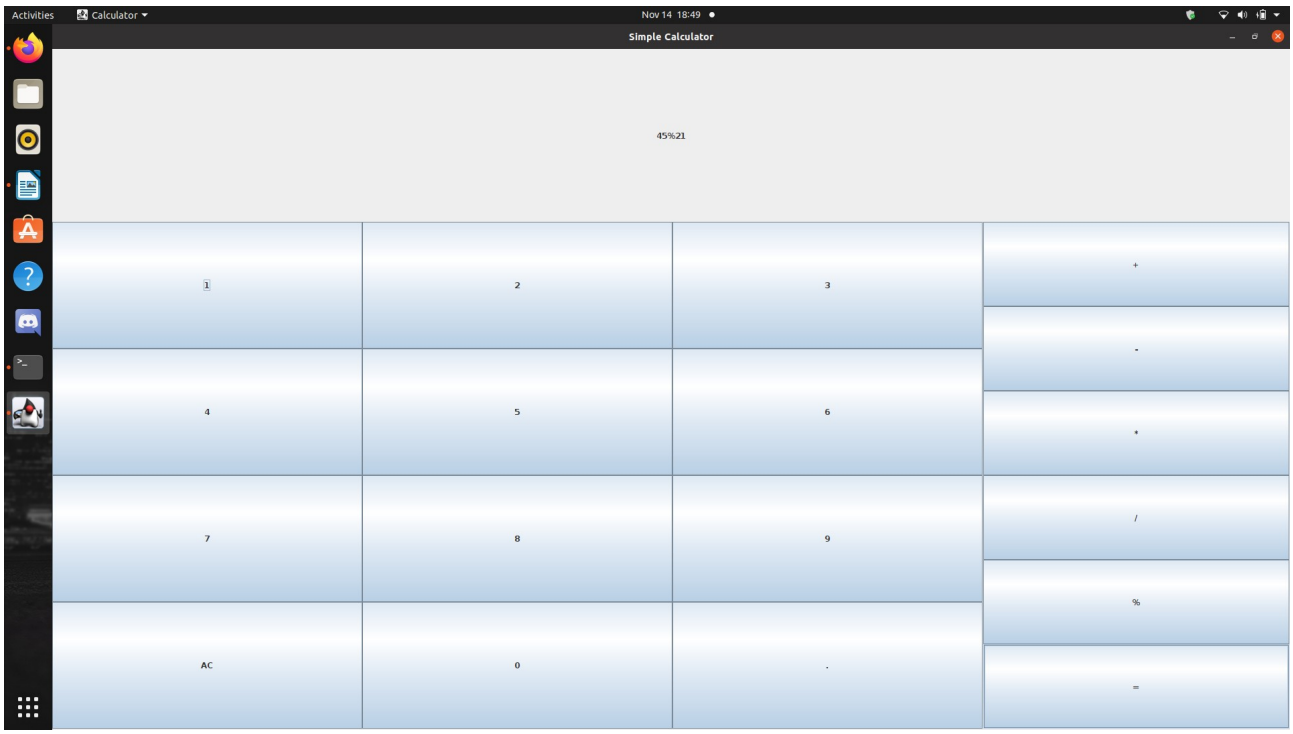

Simple Calculator
Not defined!
1
2
3
+
-
4
5
6
*
7
8
9
/
%
AC
0
.
=

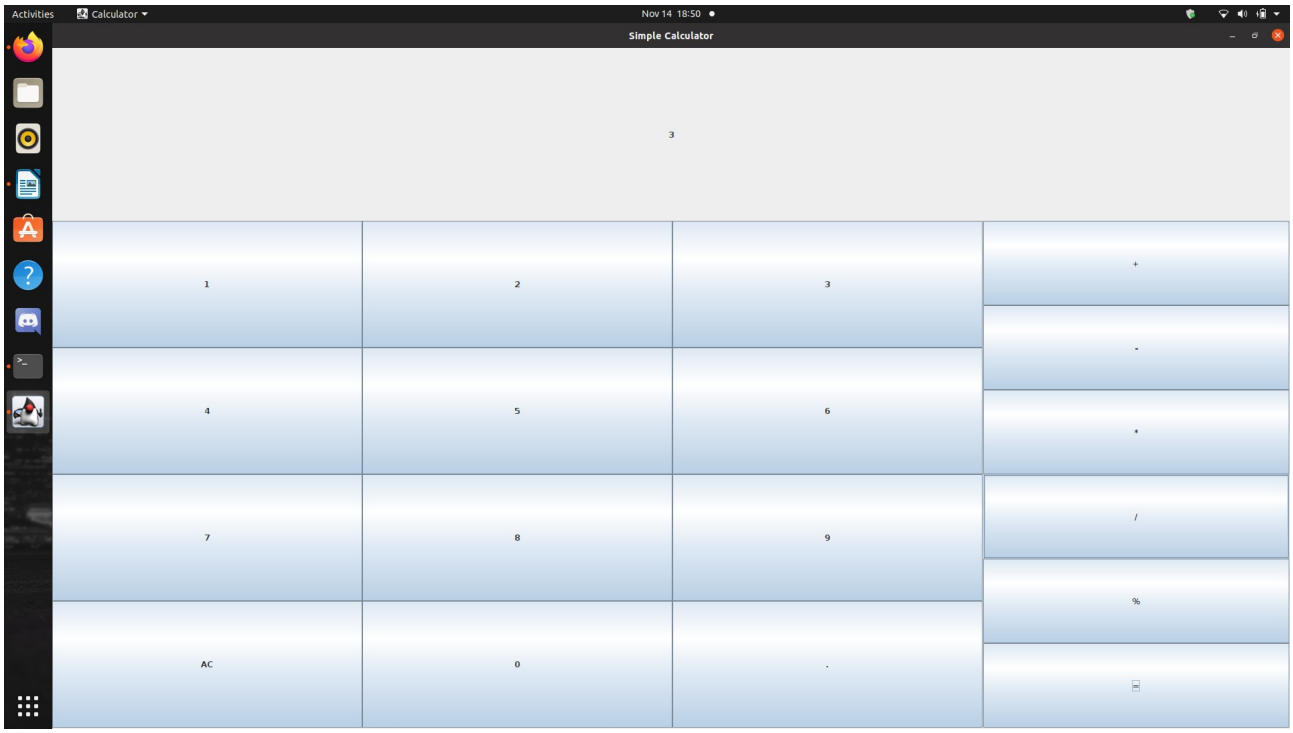

Simple Calculator
0.5*6
1
2
3
+
-
4
5
6
*
7
8
9
/
%
AC
0
.
=


Simple Calculator
3
1
2
3
+
-
4
5
6
*
7
8
9
/
%
AC
0
.
=


Simple Calculator
1/3
1
2
3
+
-
4
5
6
*
7
8
9
/
%
AC
0
.
=


Simple Calculator
0.33333334
1
2
3
+
-
4
5
6
*
7
8
9
/
%
AC
0
.
=





RESULT -

A simple calculator with GUI is created using Java Swing.

****

Experiment No. 17

11/11/2020

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 – 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[] b = new JRadioButton[3]
4. ButtonGroup bg = new ButtonGroup()
5. for i = 0 till 3
6. b[i] = new JRadioButton("")
7. b[i].setBackground(Color.GRAY)
8. bg.add(b[i])
9. frame.add(b[i])
10. endfor
11. frame.setSize(300,130)
12. frame.setLayout(new GridLayout(1,3))
13. frame.setVisible(true)
14. frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
    //Event handler code for RED button
15. b[0].setBackground(Color.RED)
16. b[1].setBackground(Color.GRAY)
17. b[2].setBackground(Color.GRAY)
18. b[0].setText("STOP")
19. b[1].setText("")
20. b[2].setText("")
21. Follow similar procedure for YELLOW and GREEN buttons
22. STOP

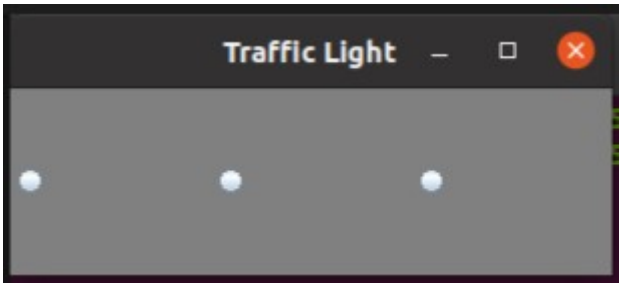
PROGRAM CODE -

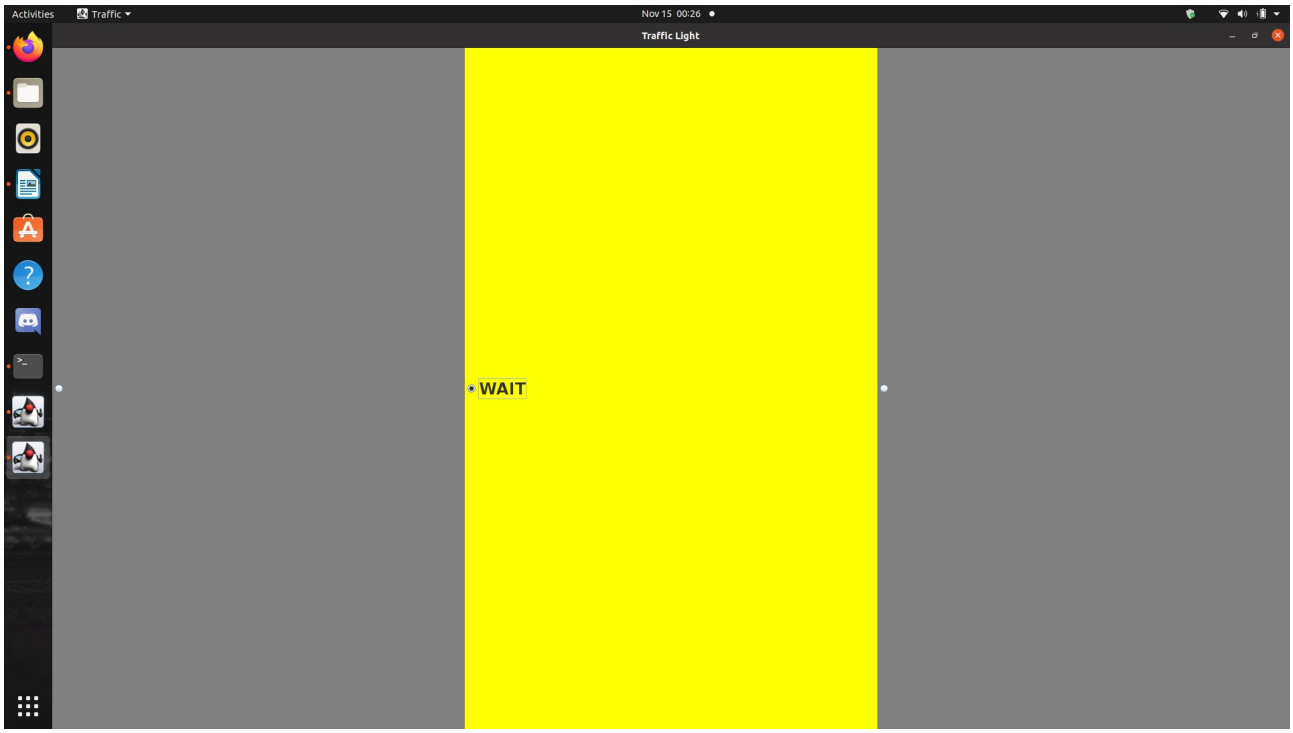
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class Traffic {
```

```
Traffic()
{
	JFrame frame = new JFrame("Traffic Light");
	JRadioButton[] b = new JRadioButton[3];
	ButtonGroup bg = new ButtonGroup();

	for(int i=0;i<3;i++)
	{
		b[i] = new JRadioButton("");
		b[i].setFont(new Font("Sans Serif", Font.BOLD, 24));

	}

	b[0].setBackground(Color.GRAY);
	b[1].setBackground(Color.GRAY);
	b[2].setBackground(Color.GRAY);

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

	frame.add(b[0]);
	frame.add(b[1]);
	frame.add(b[2]);

	frame.setSize(300,130);

	frame.setLayout(new GridLayout(1,3));

	frame.setVisible(true);

	frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

	b[0].addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent ae){

			b[0].setBackground(Color.RED);
			b[1].setBackground(Color.GRAY);
			b[2].setBackground(Color.GRAY);

			b[0].setText("STOP");
			b[1].setText("");
			b[2].setText("");
		}
	});
	b[1].addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent ae){

			b[1].setBackground(Color.YELLOW);
			b[0].setBackground(Color.GRAY);
```

```
                        b[2].setBackground(Color.GRAY);

                        b[1].setText("WAIT");
                        b[0].setText("");
                        b[2].setText("");
                    }
                });
                b[2].addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent ae){

                        b[2].setBackground(Color.GREEN);
                        b[1].setBackground(Color.GRAY);
                        b[0].setBackground(Color.GRAY);

                        b[2].setText("GO");
                        b[1].setText("");
                        b[0].setText("");
                    }
                });
        }

        public static void main(String...args)
        {
                new Traffic();
        }
}
```

SAMPLE OUTPUTS -





RESULT -

A traffic light is stimulated using Java Swing.

****

Experiment No. 18

07/01/2021

AIM -

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

CONCEPTS USED -

Quick Sort algorithm.

ALGORITHM -

```
Algorithm partition(arr, p, r)
    1   i = p – 1
    2   for j = p till r
    3           if (arr[j] <= arr[r])
    4                   if(i++ != j)
    5                           Swap arr[i] and arr[j]
    6                   endif
    7           endif
    8   endfor
    9   if(r != i + 1)
    10          Swap arr[i+1] and arr[r]
    11  endif
    12  return i+1

Algorithm quickSort(arr, p, r)
    1   if(p < r)
    2           q = partition(arr, p, r)
    3           quickSort(arr, p, q-1)
    4           quickSort(arr, q+1, r)
    5   endif
```

PROGRAM CODE -

```
import java.util.Scanner;
import java.util.Arrays;

class QuickSort
{
        static int partition(String[] arr, int p, int r)
        {
                String temp;
                String x = arr[r];
                int i = p - 1;
```

```
		for(int j = p; j < r; j++)
			if(arr[j].toLowerCase().compareTo(x.toLowerCase()) <= 0)
			{
				if(i++ != j)
				{
					temp = arr[i];
					arr[i] = arr[j];
					arr[j] = temp;
				}
			}

		if(r != i+1)
		{
			temp = arr[i+1];
			arr[i+1] = arr[r];
			arr[r] = temp;
		}

		 return i+1;
	}

	static void quickSort(String[] arr, int p, int r)
	{
		if(p<r)
		{
			int q = partition(arr, p, r);
			quickSort(arr, p, q-1);
			quickSort(arr, q+1, r);
		}
	}

	public static void main(String args[])
	{
		Scanner in = new Scanner(System.in);

		System.out.println("Enter the list size");
		int n = in.nextInt();

		String s = in.nextLine();

		String[] arr = new String[n];

		System.out.println("Enter the names");
		for(int i=0; i<n; i++)
			arr[i] = in.nextLine();

		quickSort(arr, 0, n-1);

		System.out.println("Sorted array = "+Arrays.toString(arr));
	}
}
```

SAMPLE OUTPUTS -

1.)
Enter the list size
3
Enter the names
Vishnu
hari
Harikrishnan
Sorted array = [hari, Harikrishnan, Vishnu]

2.)
Enter the list size
4
Enter the names
Mnakshi
Lechu
mili
ameesha
Sorted array = [ameesha, Lechu, mili, Mnakshi]

3.)
Enter the list size
5
Enter the names
ajith
jijo
emil
dinoy
Amal
Sorted array = [ajith, Amal, dinoy, emil, jijo]

4.)
Enter the list size
5
Enter the names
a
b
c
d
e
Sorted array = [a, b, c, d, e]

RESULT -

Quick Sort is performed on a list of names.

****