COLLEGE OF ENGINEERING KIDANGOOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

CST205 OBJECT ORIENTED PROGRAMMING USING JAVA

SEMESTER : 3

ACADEMIC YEAR : 2024-25

FACULTY : GARGI CHANDRABABU

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

EXPERIMENT NO: 01

AIM :

To write a java program to check whether the input string is palindrome or not.

ALGORITHM :

1. Start
2. Declare string variables str and rev, then initialize rev = “ ”
3. Read the string from the user and store it to the variable str
4. Calculate the length of the string and store it in the variable len
5. For the value of i = len -1 to i >=0 do
    a. rev = rev + str.charAt(i)
6. If str and rev are equal
    a. Print the entered string is palindrome
    b. Else print the entered string is not palindrome
7. Stop

PROGRAM CODE:

```
import java.util.Scanner;
public class Palindrome{

      public static void main(String []args) {

            Scanner     sc = new Scanner(System.in);
            System.out.println("Enter the string");
            String str = sc.nextLine();
            int length = str.length();
            String rev = "";
```

```
            for (int i = length-1; i >= 0; i--) {
                  rev = rev+str.charAt(i);
            }
            if(rev.equalsIgnoreCase(str)) {
                  System.out.println("Palindrome");
            }
            else {
                  System.out.println("Not palindrome");
            }
      }

}
```

OUTPUT :

```
Enter the string
Malayalam
Palindrome

Enter the string
Java
Not palindrome
```

EXPERIMENT NO: 02

AIM :

To write a java program to find the frequency of a character in a given string.

ALGORITHM :

1. Start
2. Input the string from the user and store it in variable str.
3. Input the character whose frequency needs to be computed. Store it in variable c
4. Find the length of the string str and store it in variable n.
5. Initialize start index ‘i’ to 0. Initialize the count variable to 0
6. Repeat the steps 6 to 8 until the start index ‘s’ is equal to or greater than n.
7. If str[i] is the same as character in c, then count=count+1
8. Increment s by 1
9. Print the frequency of the character.
10. Stop

PROGRAM CODE :

```
import java.util.Scanner;
class Frequency{
      public static void main (String args[]){
            Scanner s=new Scanner(System.in);
            System.out.print("Enter string : ");
            String str;
            str=s.nextLine();
            System.out.print("enter a letter: ");
            char c=s.next().charAt(0);
            int count=0;
            for(int i=0;i<str.length();i++)
```

```
            {
                  if(c==str.charAt(i)) {
                        count++;
                  }
            }
            System.out.println("Frequency of "+c+" : "+count);
      }
}
```

OUTPUT :

```
Enter string : Hello Good morning
enter a letter: o
Frequency of o : 4
```

EXPERIMENT NO: 03

AIM :

Write a Java program to check whether a given number is prime or not.

ALGORITHM :

1. Start
2. Initialize count = 0
3. Read a number n
4. if(n==1 or n==0), then print " Not a prime number"
5. else if (n<0), then print " Enter +ve number only!"
6. else,
    a. for i=2 to n/2
    b. if (n%i==0), then set c=1 and break the loop
7. if(c==0),then print " It's prime "
8. else, print " It's not a prime "
9. Stop

PROGRAM :

```
import java.util.Scanner;

class Prime {
	public static void main(String[] args) {
		int n, count = 0;
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter the number:");
		n = sc.nextInt();
		if(n==1 || n==0)
			System.out.println("Not Prime");
		else if(n<0)
			System.out.println("Enter a positive number");
		else {
			for (int i = 2;i<=(n/2);i++){
				if(n%i == 0){
					count++;
				}
			}
			if(count == 0) {
```

```
                        System.out.println("Prime");
                }
                else {
                        System.out.print("Not Prime");
                }
        }
    }
}
```

OUTPUT :

Enter the number:1

Not Prime

Enter the number:10

Not Prime

Enter the number:2

Prime

EXPERIMENT NO: 04

AIM :

To write a java program to multiply two matrices.

ALGORITHM :

1. Start
2. Declare variables i, j, k and array c[10][10]
3. Accept the number of rows and columns for the first matrix from the user as m1 and m2
4. An array a with size as m1 and m2 are declared
5. Accept the number of rows and columns for the second matrix from the user as n1 and n2
6. An array b with size as n1 and n2 are declared
7. If m1 and n2 are equal do
    a. Read the elements of first matrix from the user
    b. Read the elements of second matrix from the user
    c. For the value of i<m1 do
        i. For the value of j<n2 do
            1. c[i][j] = 0
            2. For the value of k<n1 do
                a. c[i][j] = c[i][j] + a[i][j] * b[i][j]
    d. Print the OUTPUT matrix c[i][j]
8. If m1 not equal to n2
    a. Print multiplication not possible
9. Stop

PROGRAM CODE :

```
import java.util.Scanner;
class MatrixMultiplication{
      public static void main(String[] args){
        int m1,n1,m2,n2,i,j,k;
          int [][]c = new int[50][50];
          System.out.println("Enter order of 1st matrix : ");
```

```
        Scanner e=new Scanner(System.in);
        Scanner f=new Scanner(System.in);
        m1 = e.nextInt();
        m2 = f.nextInt();
        int a[][] = new int[m1][m1];
        System.out.println("Enter order of 2nd matrix : ");
        Scanner sc=new Scanner(System.in);
        Scanner h=new Scanner(System.in);
        n1 = g.nextInt();
        n2 = h.nextInt();
        int b[][] = new int[n1][n1];
          if(m1 == n2){
                System.out.println("Enter the elements of first matrix : ");
                for(i=0;i<m1;i++) {
                      for(j=0;j<m2;j++) {
                            Scanner s = new Scanner(System.in);
                      a[i][j] = s.nextInt();
                      }
                      }
                System.out.println("Enter the elements of second matrix : ");
                for(i=0;i<n1;i++){
                      for(j=0;j<n2;j++) {
                            Scanner p = new Scanner(System.in);
                      b[i][j] = p.nextInt();
                      }
                      }
                for(i=0;i<m1;i++) {
                      for(j=0;j<n2;j++) {
                            c[i][j] = 0;
                            for(k=0;k<n1;k++) {
                                  c[i][j] = c[i][j]+a[i][k]*b[k][j];
                                  }
                            }
                      }
                System.out.println("\nProduct is : ");
                for(i=0;i<m1;i++){
                      System.out.print("\n");
                      for(j=0;j<m2;j++) {
                            System.out.print(c[i][j]+"\t");
                      }
                }
          }

          else {
                System.out.print("Multiplication not possible");
```

```
            }
        }
}
```

OUTPUT :

```
Enter order of 1st matrix :
3
3
Enter order of 2nd matrix :
3
3
Enter the elements of first matrix :
1
2
3
4
5
6
7
8
9
Enter the elements of second matrix :
9
8
7
6
5
4
3
2
1
Product is :
30   24   18
84   69   54
138  114  90
```

EXPERIMENT NO: 05

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 'print-Salary( )' which prints the salary of the Employee. Two classes 'Officer' and 'Manager' inherit 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. (Exercise to understand inheritance).

ALGORITHM :

1. Create a class ‘Employee’ with given variables and method printSalary() to print the salary.
2. Create two subclasses of class ‘Employee’ named ‘Officer’ and ‘Manager’. Include the variable ‘Specialization’ in ‘Officer’ class and ‘department’ in ‘Manager’ class.
3. Create a class with the main function. Inside the main function do the following steps.
4. Create an object to ‘Officer’ class and assign appropriate value to the variables.
5. Create an object of the ‘Manager’ class and assign appropriate values to the variables.
6. Print the values of the variables using objects.
7. Involves the function ‘printSalary()’ using an object to print salary.
8. Stop

PROGRAM :

```
import java.util.Scanner;
class Employee {
	public String name;
```

```
	public String address;
	public int age;
	public String mob;
	public float salary;
	  public void printsalary() {
		System.out.print("\nSalary of the employee is "+salary);
	  }
}
class Officer extends Employee {
		public String specialization;
	}
class Manager extends Employee {
		public String department;
	}
public class Empinheritance {
	public static void main(String args[]) {
		Officer of = new Officer();
		Manager mn = new Manager();
		Scanner in = new Scanner(System.in);
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter the details of officer\n");
		System.out.print("Enter the name : ");
		of.name = sc.nextLine();
		System.out.print("Enter the address : ");
		of.address = sc.nextLine();
		System.out.print("Enter mobile number : ");
		of.mob = sc.nextLine();
		System.out.print("Enter age : " );
		of.age = in.nextInt();
		System.out.print("Enter specialization : ");
		of.specialization = sc.nextLine();
		System.out.print("Enter salary : ");
		of.salary = in.nextFloat();
		System.out.print("Enter the details of manager\n");
		System.out.print("Enter the name : ");
```

```
            mn.name = sc.nextLine();
            System.out.print("Enter the address : ");
            mn.address = sc.nextLine();
            System.out.print("Enter mobile number : ");
            mn.mob = sc.nextLine();
            System.out.print("Enter age : " );
            mn.age = in.nextInt();
            System.out.print("Enter department : ");
            mn.department = sc.nextLine();
            System.out.print("Enter salary : ");
            mn.salary = in.nextFloat();
            System.out.print("\n\t\t---  Printing the details ---");
            System.out.print("\n\n\t The details of the officer \n\n");
            System.out.print("Name : "+of.name);
            System.out.print("\nAddress : "+of.address);
            System.out.print("\nMobile number : "+of.mob);
            System.out.print("\nspecialization : "+of.specialization);
            of.printsalary();
            System.out.print("\n\n\t The details of the manager \n\n");
            System.out.print("Name : "+mn.name);
            System.out.print("\nAddress : "+mn.address);
            System.out.print("\nMobile number : "+mn.mob);
            System.out.print("\nDepartment : "+mn.department);
            mn.printsalary();
      }
}
```

OUTPUT :

Enter the details of officer

Enter the name : Amal

Enter the address : Kottayam

Enter mobile number : 789065438

Enter age : 34

Enter specialization : DS

Enter salary : 450000

Enter the details of manager

Enter the name : Alan

Enter the address : Pala

Enter mobile number : 87643190765

Enter age : 38

Enter department : IT

Enter salary : 45000

--- Printing the details ---

The details of the officer

Name : Amal

Address : Kottayam

Mobile number : 789065438

specialization : DS

Salary of the employee is 450000.0

The details of the manager

Name : Alan

Address : Pala

Mobile number : 87643190765

Department : IT

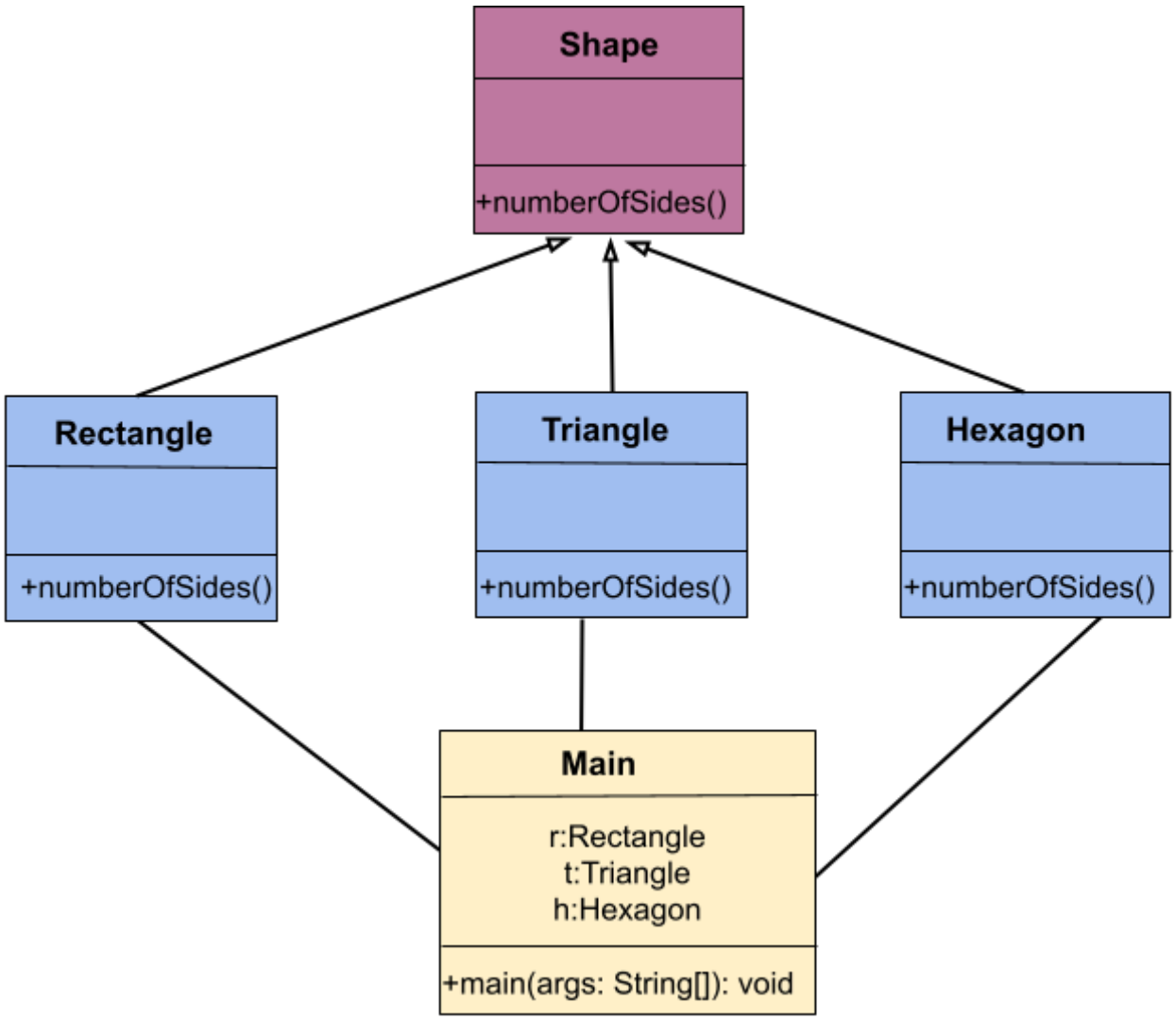
Salary of the employee is 45000.0

EXPERIMENT NO: 06

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. (Exercise to understand polymorphism).

CLASS DIAGRAM :



PROGRAM :

```
abstract class Shape {
	abstract void numberOfSides();
}
class Rectangle extends Shape {
	public void numberOfSides() {
		System.out.println("Number of sides of rectangle : 4");
	}
}
class Triangle extends Shape {
	public void numberOfSides()   {
		System.out.println("Number of sides of triangle : 3");
	}
}
class Hexagon extends Shape {
public void numberOfSides() {
	System.out.println("Number of sides of hexagon : 6");
	}
}
class Main {
public static void main(String args[]) {
	Rectangle r=new Rectangle();
	Triangle t=new Triangle();
	Hexagon h=new Hexagon();
	System.out.print("--------------\n\n\n");
	r.numberOfSides();
	t.numberOfSides();
	h.numberOfSides();
	System.out.print("\n\n\n--------------");
	}
}
```

OUTPUT :

Number of sides of rectangle : 4

Number of sides of triangle : 3

Number of sides of hexagon : 6

EXPERIMENT NO: 07

AIM :

Write a Java program to demonstrate the use of garbage collector.

ALGORITHM :

1. Start
2. Create a class football
3. Create finalize method in class football which prints Garbage collected
4. Create constructor of the class football which prints Object created
5. In main class
    a. Create anonymous object of class football as follows:

        new football();

    b. Create object reference c1 and assign null

        football c1 = new football();

        c1 = null;

    c. Create two object reference of class football and assign a reference to another

        football c2 = new football();

        football c3 = new football();

        c2 = c3;

6. Stop

PROGRAM :

```
public class Football {
	@Override
	protected void finalize() throws Throwable {
		System.out.println("Object destroyed");
	}
```

```
	public Football() {
		System.out.println("Object created");
	}

	public static void main(String[] args) {
		new Football();
		Football c1 = new Football();
		c1 = null;

		Football c2 = new Football();
		Football c3 = new Football();
		c2 = c3;

		System.gc();

	}
}
```

OUTPUT :

Object created

Object created

Object created

Object created

Object destroyed

Object destroyed

Object destroyed

EXPERIMENT NO: 08

AIM :

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

ALGORITHM :

1. Start
2. Main function is invoked
3. In the try function do the following
    a. Accept file name from the user
    b. Create a new file with the file name
    c. If successfully created
        i. Print file created with file name
    d. Else
        i. Print file already exists
4. If an exception occurred catch the exception
5. In another try function do the following
    a. Accept the text from the user as str
    b. Create an object file for FileWriter
    c. Write the string to the file.
    d. Close the file
6. If an exception occurred catch the exception
7. In another try function do the following
    a. Declare a character array ch.
    b. An object rd is created for FileReader
    c. Print the contents in the file
    d. Close the file
8. If an exception occurred catch the exception
9. Stop

PROGRAM :

```
import java.io.*;
import java.util.Scanner;
public class FileHandling {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Enter file name: ");
		String fileName = scanner.nextLine();
		File create = new File(fileName);
		try {
			if(create.createNewFile()) {
				System.out.println("File created successfully "+create.getName());
			}
			else {
				System.out.println("File already Exist");
			}
		} catch (IOException e) {
			e.printStackTrace();
		}
		try {
			System.out.print("\nEnter the text to insert : ");
			String str = scanner.nextLine();
			FileWriter file = new FileWriter("filename");
			file.write(str);
			file.close();
			System.out.print("\nText written successfully");
		} catch(Exception e) {
			e.printStackTrace();
		}
		 try {
			char ch[] = new char[100];
			FileReader rd = new FileReader("filename");
		rd.read(ch);
		System.out.println("\n\nThe contents in the file is \n");
		System.out.println(ch);
		System.out.print("\n\n\n--------------");
		rd.close();
		} catch(Exception e) {
			e.printStackTrace();
		 }
		 scanner.close();
		 }
	}
```

OUTPUT :

Enter file name: sample

File created successfully sample

Enter the text to insert : hello world

Text written successfully

The contents in the file is

hello world

EXPERIMENT NO: 09

AIM :

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

ALGORITHM :

1. Start
2. Main function is invoked
3. Initialize integer variables num and sum
4. Accept the maximum limit from the user
5. Accept the string numbers separated with space
6. Create a string tokenizer object str
7. While (str.hasMoreTokens())
    a. Sum = Sum + Integer.parseInt(str.nextToken()
8. Print the sum
9. Stop

PROGRAM :

```
import java.util.*;

public class StringTokenizerDemo {
      public static void main(String args[]) {
            int num, sum = 0;
            Scanner sc = new Scanner(System.in);
            Scanner st = new Scanner(System.in);
            System.out.print("Enter the limit : ");
            num = sc.nextInt();
            System.out.print("\nEnter "+num+" numbers separated by space : ");
            String list = st.nextLine();
            StringTokenizer str = new StringTokenizer(list," ");
            System.out.println("You entered the following integers :");
```

```
            while(str.hasMoreTokens()) {
                  int element = Integer.parseInt(str.nextToken());
                  System.out.print(element+" ");
                  sum = sum + element;
            }
            System.out.println("\nThe sum of the integers = "+sum);
            System.out.print("\n\n\n--------------");
            }
      }
```

OUTPUT :

Enter the limit : 5

Enter 5 numbers separated by space : 1 2 3 4 5

You entered the following integers :

1 2 3 4 5

The sum of the integers = 15

EXPERIMENT NO: 10

AIM :

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

ALGORITHM :

1. Start
2. Define the `Division` Class:
    a. Create a method `divide(int numerator, int denominator)` that:
        i. Accepts two integers as parameters (`numerator` and `denominator`).
        ii. Throws an `ArithmeticException` if a division by zero occurs.
        iii. Returns the result of `numerator/denominator`.
3. Define the `ExceptionHandlingExample` Class:
    a. In the `main` method:
        i. Initialize a `Scanner` to accept user input.
        ii. Prompt the user to enter a value for the `numerator` and store it in `num1`.
        iii. Prompt the user to enter a value for the `denominator` and store it in `num2`.
4. Use a `try` block:
    a. Attempt to:
        i. Call the `divide` method of the `Division` class, passing `num1` and `num2`.
        ii. Print the result if the division is successful.
5. Handle Exceptions with `catch` blocks:
6. If an `ArithmeticException` is thrown:
    a. Print an error message indicating that division by zero is not allowed.
7. If any other `Exception` is thrown:
    a. Print a message stating an unexpected error occurred.
    b. Print the stack trace to identify the error.
8. Use a `finally` block:
    a. Print "Execution completed. Cleaning up resources..." to indicate that the code has finished executing and resources are being released.
9. Continue Program Execution:

a. Print a message stating "Program continues after try-catch-finally block" to confirm that the program has continued after handling any exceptions.

10. Stop

PROGRAM :

```
package gargi;
import java.util.Scanner;
class Division {
   // Method that throws an exception if division by zero is attempted
   public static double divide(int numerator, int denominator) throws ArithmeticException {
       return numerator / denominator; // May throw ArithmeticException
   }
}
public class ExceptionHandlingExample {
   public static void main(String[] args) {
       try {
            Scanner s = new Scanner(System.in);
            System.out.print("Enter numerator : ");
            int num1 = s.nextInt();
            System.out.print("Enter denominator : ");
           int num2 = s.nextInt();
           double result = Division.divide(num1, num2);
           System.out.println("Result: " + result);
       } catch (ArithmeticException e) {
           System.err.println("Error: Division by zero is not allowed.");
       } catch (Exception e) {
           System.out.println("An unexpected error occurred: " );
           e.printStackTrace();
       } finally {
           System.out.println("Execution completed. Cleaning up resources...");
       }
       System.out.println("Program continues after try-catch-finally block.");
   }
}
```

OUTPUT :

```
Enter numerator : 12
Enter denominator : 1.1
An unexpected error occurred:
java.util.InputMismatchException
	at java.base/java.util.Scanner.throwFor(Scanner.java:964)
	at java.base/java.util.Scanner.next(Scanner.java:1619)
	at java.base/java.util.Scanner.nextInt(Scanner.java:2284)
	at java.base/java.util.Scanner.nextInt(Scanner.java:2238)
	at gargi.ExceptionHandlingExample.main(ExceptionHandlingExample.java:19)
Execution completed. Cleaning up resources...
Program continues after try-catch-finally block.
```

```
Enter numerator : 10
Enter denominator : 0
Error: Division by zero is not allowed.
Execution completed. Cleaning up resources...
Program continues after try-catch-finally block.
```

```
Enter numerator : 12
Enter denominator : 2
Result: 6.0
Execution completed. Cleaning up resources...
Program continues after try-catch-finally block.
```

EXPERIMENT NO: 11

AIM :

Write a Java program that shows how to create a user-defined exception.

ALGORITHM :

1. Start
2. Define the `NegativeException` Class:
    a. Extend `Exception` to create a custom exception `NegativeException`.
    b. Define a constructor that takes a `String` message and passes it to the superclass `Exception`.
3. Define the `NegativeNumberExample` Class:
    a. In the `main` method:
        i. Initialize a `Scanner` to accept user input.
        ii. Prompt the user to enter a number and store it in `number`.
4. Use a `try` block:
    a. Call the `checkForNegative` method with `number` as an argument.
    b. If no exception is thrown, print "You entered: `number`".
5. Define the `checkForNegative` Method:
    a. Accept an integer parameter `number`.
    b. If `number` is negative:
        i. Throw a `NegativeException` with the message "Negative numbers are not allowed."
6. Handle the Exception with a `catch` Block:
    a. Catch the `NegativeException` if thrown.
    b. Print "Caught exception:" followed by the exception's message.
7. Use a `finally` Block:
    a. Close the `Scanner` to release resources, ensuring it executes regardless of whether an exception occurred.

PROGRAM :

```
import java.util.Scanner;

class NegativeException extends Exception {
   public NegativeException(String message) {
       super(message);
   }
}
public class NegativeNumberExample {
   public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);

       System.out.print("Enter a number: ");
       int number = scanner.nextInt();

       try {
           checkForNegative(number);
           System.out.println("You entered: " + number);
       } catch (NegativeException e) {
           System.out.println("Caught exception: " + e.getMessage());
       } finally {
           scanner.close();
       }
   }
   public static void checkForNegative(int number) throws NegativeException {
       if (number < 0) {
           throw new NegativeException("Negative numbers are not allowed.");
       }
   }
}
```

OUTPUT :

Enter a number: -1

Negative numbers are not allowed.

Enter a number: 9

You entered: 9

EXPERIMENT NO: 12

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.

ALGORITHM :

1. Start
2. Initialize the EvenOddThread class, which inherits from Thread.
3. In the run method of EvenOddThread:
4. Loop 10 times to generate 10 random numbers.
5. For each iteration:
    a. Generate a random integer n between 0 and 49.
    b. Print the generated random number.
    c. Check if n is even:
6. If even:
    a. Create an instance of the Square thread with n as the input.
    b. Start the Square thread to calculate and print the square of n.
7. If odd:
    a. Create an instance of the Cube thread with n as the input.
    b. Start the Cube thread to calculate and print the cube of n.
8. Pause for 1 second to allow each thread to complete and to space out the random number generation.
9. Square class:
    a. Initialize with an integer parameter n.
    b. Calculate n * n and store it in a variable sq.
    c. In the run method, print the square of the integer.
10. Cube class:
    a. Initialize with an integer parameter n.
    b. Calculate n * n * n and store it in a variable cube.
    c. In the run method, print the cube of the integer.
11. Main Method:

a. In the MainClass, create an instance of EvenOddThread.
b. Start the EvenOddThread instance to begin the random number generation, square, and cube calculations.

12. Stop

PROGRAM :

```
import java.util.*;

public class EvenOddThread extends Thread {
	Random random = new Random();
	public void run() {
		for(int i=0;i<10;i++) {
			int n = random.nextInt(50);
			System.out.println("Random number : "+n);
			if (n%2==0) {
				Square s = new Square(n);
				s.start();
				System.out.print("Square of "+n);
			}
			else {
				Cube c = new Cube(n);
				c.start();
				System.out.print("Cube of "+n);
			}
			try {
				Thread.sleep(1000);
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
		}
	}
}
class Square extends Thread {
	int sq;
	Square(int n){
		sq = n*n;
	}
	public void run() {
		System.out.print(" : "+sq+"\n");
	}
}
class Cube extends Thread {
	int cube;
```

```
	Cube(int n){
		cube = n*n*n;
	}
	public void run() {
		System.out.print(" : "+cube+"\n");
	}
}
class MainClass {
	public static void main(String[] args) {
		EvenOddThread obj1 = new EvenOddThread();
		obj1.start();
	}
}
```

OUTPUT :

Random number : 29

Cube of 29 : 24389

Random number : 41

Cube of 41 : 68921

Random number : 46

Square of 46 : 2116

Random number : 46

Square of 46 : 2116

Random number : 46

Square of 46 : 2116

Random number : 1

Cube of 1 : 1

Random number : 11

Cube of 11 : 1331

Random number : 45

Cube of 45 : 91125

Random number : 18

Square of 18 : 324

Random number : 4

Square of 4 : 16

EXPERIMENT NO: 13

AIM :

Write a Java program that shows thread synchronization.

ALGORITHM :

1. Start
2. Initialize the Table class:
    a. Define a method printTable(int n):
    b. Mark the method as synchronized to ensure that only one thread can access it at a time.
    c. Print "Multiplication Table of n".
    d. Loop from 1 to 10:
        i. Calculate the product of the loop index i and n.
        ii. Print the multiplication result in the format i x n = (i*n).
        iii. Pause for 1 second to space out the output.
3. Define MyThread1 class:
    a. Extend Thread.
    b. Define a constructor that takes a Table object and assigns it to an instance variable.
    c. Override the run method:
        i. Call printTable(1) on the Table object to print the multiplication table of 1.
4. Define MyThread2 class:
    a. Extend Thread.
    b. Define a constructor that takes a Table object and assigns it to an instance variable.
    c. Override the run method:
        i. Call printTable(2) on the Table object to print the multiplication table of 2.
5. In the main method of the Synchronization class:
    a. Create a single instance of the Table class.
    b. Create two instances of MyThread1 and MyThread2, passing the Table instance to both.
    c. Start both threads:
    d. t1.start() for MyThread1 (printing the multiplication table of 1).
    e. t2.start() for MyThread2 (printing the multiplication table of 2).
6. Stop

PROGRAM :

```
class Table {
	synchronized void printTable(int n){//synchronized method
		System.out.println("Multiplication Table of "+n);
		for(int i=1;i<=10;i++){
			System.out.println(i+" x "+n+" = "+(i*n));
			try {
				Thread.sleep(1000);
				}catch(Exception e){
					System.out.println(e);
				}
			}
		}
}
class MyThread1 extends Thread {
	Table t;
	MyThread1(Table t){
		this.t=t;
	}
	public void run(){
		t.printTable(1);
	}
}
class MyThread2 extends Thread {
	Table t;
	MyThread2(Table t){
		this.t=t;
	}
	public void run(){
		t.printTable(2);
	}
}
public class Synchronization {
	public static void main(String args[]){
		Table obj = new Table();//only one object
		MyThread1 t1=new MyThread1(obj);
		MyThread2 t2=new MyThread2(obj);
		t1.start();
		t2.start();
	}
}
```

OUTPUT :

Multiplication Table of 2

1 x 2 = 2

2 x 2 = 4

3 x 2 = 6

4 x 2 = 8

5 x 2 = 10

6 x 2 = 12

7 x 2 = 14

8 x 2 = 16

9 x 2 = 18

10 x 2 = 20

Multiplication Table of 1

1 x 1 = 1

2 x 1 = 2

3 x 1 = 3

4 x 1 = 4

5 x 1 = 5

6 x 1 = 6

7 x 1 = 7

8 x 1 = 8

9 x 1 = 9

10 x 1 = 10

EXPERIMENT NO: 14

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.*

ALGORITHM :

1. Start
2. Initialize TrafficLight Window:
    a. Set up the main window with title, size, layout, and background color.
    b. Define and add radio buttons (`Red`, `Yellow`, `Green`) to represent traffic light options.
    c. Create a `textArea` to display the current traffic signal action ("STOP!", "WAIT!", or "GO...").
    d. Add radio buttons to a `ButtonGroup` to ensure only one button is selected at a time.
3. Implement Item Listener for Radio Buttons:
    a. Define `itemStateChanged` method to handle button selection events.
    b. If `Red` button is selected:
        i. Set `msg` to "STOP!"
        ii. Set `textArea` color to Red
        iii. Update `x` flag to 1 (Red light active)
        iv. Call `repaint` to update the display
    c. If `Yellow` button is selected:
        i. Set `msg` to "WAIT!"
        ii. Set `textArea` color to Yellow
        iii. Update `y` flag to 1 (Yellow light active)
        iv. Call `repaint` to update the display
    d. If `Green` button is selected:
        i. Set `msg` to "GO..."
        ii. Set `textArea` color to Green

iii. Update `z` flag to 1 (Green light active)

iv. Call `repaint` to update the display

e. Update the `textArea` to display the `msg`.

4. Override the `paint` Method:
   a. Draw the Traffic Light Circles:
      i. Draw three circles for Red, Yellow, and Green lights at fixed positions.
   b. Fill the Active Light:
      i. If `x` flag is set to 1, fill the top circle with Red and reset other circles to White.
      ii. If `y` flag is set to 1, fill the middle circle with Yellow and reset other circles to White.
      iii. If `z` flag is set to 1, fill the bottom circle with Green and reset other circles to White.
   c. Reset flags (`x`, `y`, `z`) after each state update.
5. Run Program:
   a. Launch the main window and display the traffic light interface.
6. End

PROGRAM :

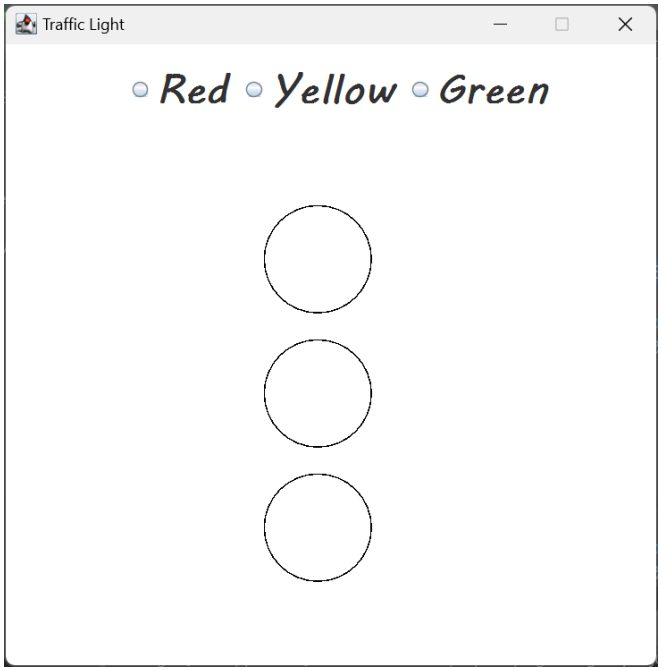
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TrafficLight extends JFrame implements ItemListener {
	JRadioButton r1;
	JRadioButton r2;
	JRadioButton r3;
	JTextArea textArea = new JTextArea();
	String msg;
	ButtonGroup b = new ButtonGroup();
	JPanel panel;
	Font myfont = new Font("MV Boli", Font.BOLD, 30);
	int x=0, y=0, z=0;
	public TrafficLight() {
		panel = new JPanel();
		panel.setBounds(0, 0, 500, 100);
		panel.setBackground(Color.WHITE);
		textArea.setFont(new Font("MV Boli", Font.BOLD, 20));
		textArea.setBackground(Color.BLACK);
		textArea.setEditable(false);
```

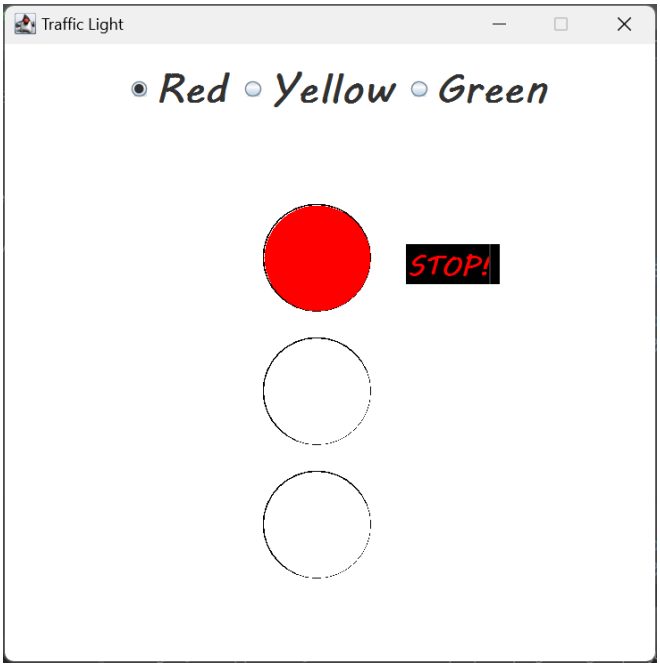
```
        this.setTitle("Traffic Light");
        this.setSize(500, 500);
        this.setLayout(null);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.getContentPane().setBackground(Color.WHITE);
        this.setResizable(false);
        r1 = new JRadioButton("Red");
        r1.setBackground(Color.WHITE);
        r1.setFocusable(false);
        r1.setFont(myfont);
        r2 = new JRadioButton("Yellow");
        r2.setBackground(Color.WHITE);
        r2.setFocusable(false);
        r2.setFont(myfont);
        r3 = new JRadioButton("Green");
        r3.setBackground(Color.WHITE);
        r3.setFocusable(false);
        r3.setFont(myfont);
        r1.addItemListener(this);
        r2.addItemListener(this);
        r3.addItemListener(this);
        this.add(panel);
        panel.add(r1);
        panel.add(r2);
        panel.add(r3);
        this.add(textArea);
        b.add(r1);
        b.add(r2);
        b.add(r3);
        this.setVisible(true);
    }
    public void itemStateChanged(ItemEvent e) {
        if (e.getSource() == r1) {
            if (e.getStateChange() == ItemEvent.SELECTED) {
                msg = "STOP!";
                textArea.setForeground(Color.RED);
                textArea.setBounds(300, 150, 70, 30);
                x = 1;
                repaint();
            }
        }
        if (e.getSource() == r2) {
            if (e.getStateChange() == 1) {
                msg = "WAIT!";
                textArea.setForeground(Color.YELLOW);
```

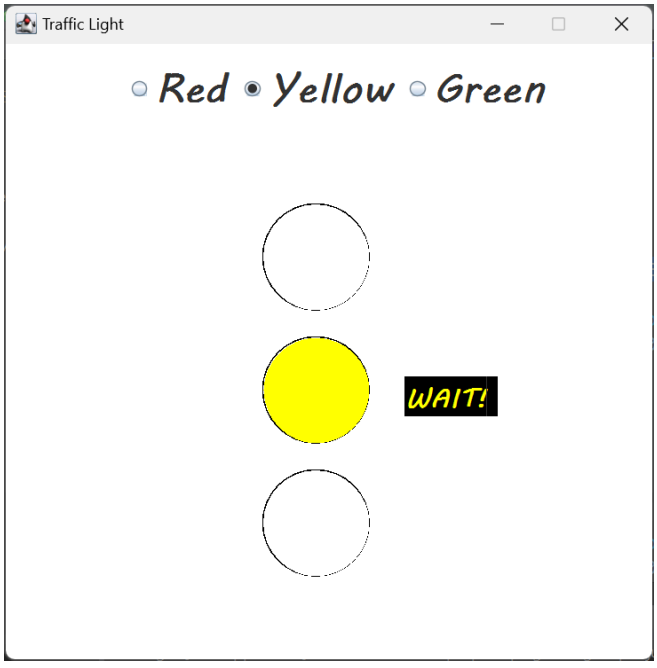
```
                        textArea.setBounds(300, 250, 70, 30);
                        y = 1;
                        repaint();
                  }
            }
            if (e.getSource() == r3) {
                  if (e.getStateChange() == 1) {
                        msg = "GO...";
                        textArea.setForeground(Color.GREEN);
                        textArea.setBounds(300, 350, 60, 30);
                        z = 1;
                        repaint();
                  }
            }
            textArea.setText(msg);
      }
      public void paint(Graphics g) {
            super.paint(g);
            g.drawOval(200, 150, 80, 80);
            g.drawOval(200, 250, 80, 80);
            g.drawOval(200, 350, 80, 80);
            if (x == 1) {
                  g.setColor(Color.RED);
                  g.fillOval(201, 151, 79, 79);
                  g.setColor(Color.WHITE);
                  g.fillOval(201, 251, 79, 79);
                  g.setColor(Color.WHITE);
                  g.fillOval(201, 351, 79, 79);
                  x = 0;
            }
            if (y == 1) {
                  g.setColor(Color.WHITE);
                  g.fillOval(201, 151, 79, 79);
                  g.setColor(Color.YELLOW);
                  g.fillOval(201, 251, 79, 79);
                  g.setColor(Color.WHITE);
                  g.fillOval(201, 351, 79, 79);
                  y = 0;
            }
            if (z == 1) {
                  g.setColor(Color.WHITE);
                  g.fillOval(201, 151, 79, 79);
                  g.setColor(Color.WHITE);
                  g.fillOval(201, 251, 79, 79);
                  g.setColor(Color.GREEN);
```

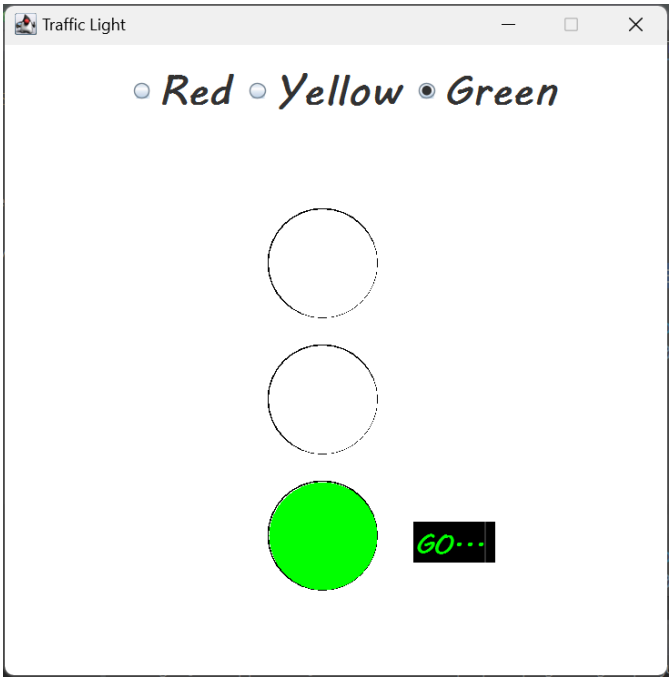
```
                g.fillOval(201, 351, 79, 79);
                z = 0;
            }
        }
        public static void main(String[] args) {
            new TrafficLight();
        }
}
```

OUTPUT :









EXPERIMENT NO: 15

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.

ALGORITHM :

1. Start
2. Initialize GUI Components:
    - Create a JFrame titled "Calculator" with fixed size (420x550).
    - Create a JTextField for displaying the input and results.
    - Create number buttons (0 to 9) and function buttons (+, -, *, /, =, DEL, C, (-), .).
    - Set the font for the buttons and text field.
    - Add buttons to a grid layout (JPanel) for proper arrangement.
3. Set Button Actions:
    - Add action listeners for all buttons to handle user input.
4. Button Action Logic:
    - For number buttons (0-9):
        i. When clicked, append the corresponding number to the textField.
    - For decimal button (.):
        i. When clicked, append a decimal point to the textField if it's not already present.
    - For operation buttons (+, -, *, /):
        i. When clicked, store the current value of textField as num1 (a double) and store the operation as operator.
        ii. Clear the textField to prepare for the next number.
    - For equals button (=):
        i. Retrieve the second number (num2) from textField.
        ii. Perform the operation based on the operator:
            1. If +, add num1 and num2.
            2. If -, subtract num1 and num2.
            3. If *, multiply num1 and num2.

4. If /, divide num1 by num2 (with exception handling for division by zero).

iii. Display the result in textField.

- For clear button (C):
  i. Clear the textField to reset the calculator.
- For delete button (DEL):
  i. Remove the last character from textField.
- For negative button (-):
  i. Convert the current number in textField to its negative counterpart.

5. Handle Division by Zero:
   - If the division operation results in division by zero, show an error message in textField.
6. End

PROGRAM :

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Calculator implements ActionListener {

	JFrame frame;
	JTextField textField;
	JButton[] numberButtons = new JButton[10];
	JButton[] funcButtons = new JButton[9];
	JButton addButton, subButton, mulButton, divButton;
	JButton decButton, equButton, delButton, clrButton, negButton;
	JPanel panel;

	Font myfont = new Font("Ink Free", Font.BOLD, 30);

	double num1=0, num2=0, result=0;
	char operator;

	public Calculator() {
		JFrame frame = new JFrame("Calculator");
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setSize(420, 550);
		frame.setLayout(null);
		textField = new JTextField();
		textField.setBounds(50, 25, 300, 50);
		textField.setFont(myfont);
```

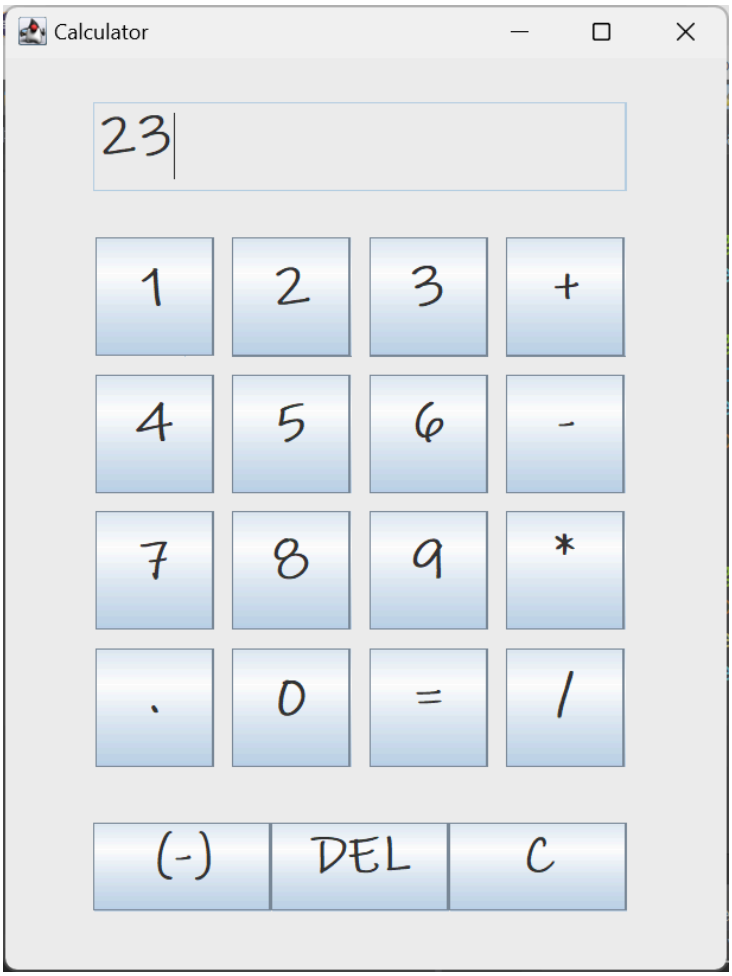
```
        textField.setEditable(false);

        addButton = new JButton("+");
        subButton = new JButton("-");
        mulButton = new JButton("*");
        divButton = new JButton("/");
        decButton = new JButton(".");
        equButton = new JButton("=");
        delButton = new JButton("DEL");
        clrButton = new JButton("C");
        negButton = new JButton("(-)");

        funcButtons[0] = addButton;
        funcButtons[1] = subButton;
        funcButtons[2] = mulButton;
        funcButtons[3] = divButton;
        funcButtons[4] = decButton;
        funcButtons[5] = equButton;
        funcButtons[6] = delButton;
        funcButtons[7] = clrButton;
        funcButtons[8] = negButton;

        for (int i = 0; i < 9; i++) {
              funcButtons[i].addActionListener(this);
              funcButtons[i].setFont(myfont);
              funcButtons[i].setFocusable(false);
        }

        for (int i = 0; i < 10; i++) {
              numberButtons[i] = new JButton(String.valueOf(i));
              numberButtons[i].addActionListener(this);
              numberButtons[i].setFont(myfont);
              numberButtons[i].setFocusable(false);
        }

        negButton.setBounds(50,430,100,50);
        delButton.setBounds(150,430,100,50);
        clrButton.setBounds(250,430,100,50);
        panel = new JPanel();
        panel.setBounds(50, 100, 300, 300);
        panel.setLayout(new GridLayout(4, 4, 10, 10));

        panel.add(numberButtons[1]);
        panel.add(numberButtons[2]);
        panel.add(numberButtons[3]);
```

```
        panel.add(addButton);
        panel.add(numberButtons[4]);
        panel.add(numberButtons[5]);
        panel.add(numberButtons[6]);
        panel.add(subButton);
        panel.add(numberButtons[7]);
        panel.add(numberButtons[8]);
        panel.add(numberButtons[9]);
        panel.add(mulButton);
        panel.add(decButton);
        panel.add(numberButtons[0]);
        panel.add(equButton);
        panel.add(divButton);

        frame.add(panel);
        frame.add(negButton);
        frame.add(delButton);
        frame.add(clrButton);
        frame.add(textField);

        frame.setVisible(true);
    }
    public static void main(String[] args) {
        Calculator calc = new Calculator();
    }
    public void actionPerformed(ActionEvent e) {
        int err = 0;
        for (int i = 0; i < 10; i++) {
            if (e.getSource() == numberButtons[i]) {
                textField.setText(textField.getText().concat(String.valueOf(i)));
            }
        }

        if (e.getSource() == decButton) {
            textField.setText(textField.getText().concat("."));
        }

        if(e.getSource()==addButton) {
            num1 = Double.parseDouble(textField.getText());
            operator ='+';
            textField.setText("");
        }
        if(e.getSource()==subButton) {
            num1 = Double.parseDouble(textField.getText());
            operator ='-';
```

```
            textField.setText("");
        }
        if(e.getSource()==mulButton) {
            num1 = Double.parseDouble(textField.getText());
            operator ='*';
            textField.setText("");
        }
        if(e.getSource()==divButton) {
            num1 = Double.parseDouble(textField.getText());
            operator ='/';
            textField.setText("");
        }
        if(e.getSource()==equButton) {
            num2=Double.parseDouble(textField.getText());

            switch(operator) {
            case'+':
                  result=num1+num2;
                  break;
            case'-':
                  result=num1-num2;
                  break;
            case'*':
                  result=num1*num2;
                  break;
            case'/':
                  try {
                 if (num2 == 0) {
                     err = 1;
                     throw (new ArithmeticException("Error: Division by zero!"));
                 }
                 result = num1 / num2;
               } catch (ArithmeticException error) {
                    textField.setText(error.getMessage());
               }
               break;
               }
               if (err == 0) {
               textField.setText(String.valueOf(result));
               num1=result;
               }
        }
        if(e.getSource()==clrButton) {
            textField.setText("");
        }
```

```
            if(e.getSource()==delButton) {
                  String string = textField.getText();
                  textField.setText("");
                  for(int i=0;i<string.length()-1;i++) {
                        textField.setText(textField.getText()+string.charAt(i));
                  }
            }
            if(e.getSource()==negButton) {
                  double temp = Double.parseDouble(textField.getText());
                  temp*=-1;
                  textField.setText(String.valueOf(temp));
            }

      }
}
```

OUTPUT :



EXPERIMENT NO: 16

AIM :

Write a Java program to display all records from a table using Java Database Connectivity (JDBC).

ALGORITHM :

1. Load and Register Driver
2. Establish the connection between java application and database
3. Create statement object
4. Send and execute SQL query
5. Process Result from ResultSet
6. Close the connection and statement.

PROGRAM :

```
import java.sql.*;
public class Database {
	public static void main(String[] args) throws ClassNotFoundException{
		try{
			//1. Load and Register Driver
			Class.forName("com.mysql.cj.jdbc.Driver");
			//2. Establish the connection between java application and database
			Connection con =   DriverManager.getConnection
					("jdbc:mysql://localhost:3306/studentdb","root","passw");
			System.out.println("***Connected to the database.***\n");
			//3. Creation of statement object
			Statement stmt = con.createStatement();
			//4. Send and execute SQL query
			ResultSet rs = stmt.executeQuery("select * from studentdetails where mark>50");
			//5. Process Result from ResultSet
			while(rs.next()){
				System.out.println("Name = "+rs.getString(1)
						+"\nRollNo =  "+rs.getInt(2)+"\nMark = "+rs.getInt(3));
				System.out.println();
			}
			//6. Close the connection and statement.
```

```
                    stmt.close();
                con.close();
            }
            catch(SQLException e){
                    e.printStackTrace();
            }
        }
}
```

OUTPUT :

Connected to the database.

Name = Gowri

RollNo = 2

Mark = 90

Name = Gopika

RollNo = 3

Mark = 92

Name = Parvathy

RollNo = 4

Mark = 90

EXPERIMENT NO: 17

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.*

ALGORITHM :

1. Start
2. Initialize a doubly linked list `dlist`
3. Display Menu with options:
   - 1: Insert element at beginning
   - 2: Insert element at end
   - 3: Insert element at a specific position
   - 4: Delete a given element
   - 5: Display elements in the list
   - 6: Exit
4. Repeat until user chooses to exit:
   - Prompt the user to choose an option (1-6)
   - Read user input `key`
5. Switch (key):
   - Case 1: Insert element at the beginning
     i. Prompt for element to insert
     ii. Insert element at the beginning of `dlist`
     iii. Display "Successfully inserted"
   - Case 2: Insert element at the end
     i. Prompt for element to insert
     ii. Insert element at the end of `dlist`
     iii. Display "Successfully inserted"
   - Case 3: Insert element at a specific position
     i. Prompt for `position` to insert element
     ii. If `position` is within the range [0, size of `dlist`]:

        - Prompt for element to insert
        - Insert element at specified position in `dlist`
        - Display "Successfully inserted"
    iii. Else, display "Position must be between 0 and the current size of the list"
- Case 4: Delete a specified element
    i. Prompt for element to remove
    ii. If `dlist` contains the element:
        - Remove the first occurrence of the element from `dlist`
        - Display "Successfully removed"
    iii. Else, display "Element not found in the list"
- Case 5: Display elements in the list
    i. If `dlist` is not empty:
        - Iterate through each element in `dlist` and display in format: `element1 <-> element2 <-> ... <-> NULL`
    ii. Else, display "List is empty"
- Case 6: Exit
    i. Display "Exiting program..."
    ii. Close any open resources
    iii. Stop the program
- Default: Display "Invalid option. Please enter a number between 1 and 6."

PROGRAM :

```
import java.util.LinkedList;
import java.util.*;
public class DoublyLinkedList {
	public static void main(String[] args) {
		int element, position;
		LinkedList<Integer> dlist = new LinkedList<Integer>();
		System.out.println("1: Insert element at beginning\n2: Insert element at end"
				+ "\n3: Insert element at position\n4: Delete a given element\n"
				+ "5: Display elements in the list\n6: Exit");
		Scanner sc = new Scanner(System.in);
		do {
			System.out.print("Choose options 1-6: ");
			int key = sc.nextInt();
			switch (key) {
			case 1:
				System.out.print("Enter an element to insert at beginning: ");
				element = sc.nextInt();
```

```
                        dlist.addFirst(element);
                        System.out.println("~~~~Successfully inserted~~~~");
                break;
                case 2:
                        System.out.print("Enter an element to insert at end: ");
                        element = sc.nextInt();
                        dlist.addLast(element);
                        System.out.println("~~~~Successfully inserted~~~~");
                break;
                case 3:
                        System.out.print("Enter a position to insert element: ");
                        position = sc.nextInt();
                        if (position<=dlist.size()) {
                                System.out.print("Enter element: ");
                                element = sc.nextInt();
                                dlist.add(position,element);
                        }
                        else {
                        System.out.println("Enter the position between 0 to "+dlist.size());
                        }
                break;
                case 4:
                        System.out.print("Enter element to remove: ");
                        element = sc.nextInt();
                        if (dlist.contains(element)) {
                                try {
                                     //Removes the first occurrence of an element if it exists.
                                       dlist.remove(Integer.valueOf(element));
                                       System.out.println("~~~~Successfully removed~~~~");
                                } catch (Exception e) {
                                       System.out.println(e.getMessage());
                                }
                        }
                break;

                case 5:
                        System.out.println("~~~~Display elements~~~~");
                        Iterator<Integer> itr = dlist.iterator();
                        while (itr.hasNext()) {
                                System.out.print(itr.next()+" <-> ");
                        }
                        System.out.println("NULL");
                break;
                case 6:
                        System.out.println("Exiting program");
                        sc.close();
                        System.exit(0);
                default:
                        throw new IllegalArgumentException("Unexpected value: " + key);
                }
        } while (true);
```

```
        }
}
```

OUTPUT :

1: Insert element at beginning

2: Insert element at end

3: Insert element at position

4: Delete a given element

5: Display elements in the list

6: Exit

Choose options 1-6: 1

Enter an element to insert at beginning: 1

~~~~Successfully inserted~~~~

Choose options 1-6: 1

Enter an element to insert at beginning: 2

~~~~Successfully inserted~~~~

Choose options 1-6: 1

Enter an element to insert at beginning: 3

~~~~Successfully inserted~~~~

Choose options 1-6: 2

Enter an element to insert at end: 5

~~~~Successfully inserted~~~~

Choose options 1-6: 5

~~~~Display elements~~~~

3 <-> 2 <-> 1 <-> 5 <-> NULL

Choose options 1-6: 4

Enter element to remove: 5

~~~~Successfully removed~~~~

Choose options 1-6: 5

~~~~Display elements~~~~

3 <-> 2 <-> 1 <-> NULL

Choose options 1-6: 4

Enter element to remove: 2

~~~~Successfully removed~~~~

Choose options 1-6: 5

~~~~Display elements~~~~

3 <-> 1 <-> NULL

Choose options 1-6: 6
~~~~

EXPERIMENT NO: 18

AIM :

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

ALGORITHM :

1. Start
2. Define the `quicksort` class
3. Define the `quickSort` function:
4. Input: Array `A`, start index `p`, end index `r`
5. Steps:
    a. If `p < r`:
        i. Set `q = partition(A, p, r)`
        ii. Call `quickSort(A, p, q - 1)` to sort the left part
        iii. Call `quickSort(A, q + 1, r)` to sort the right part
6. Define the `partition` function:
7. Input: Array `A`, start index `p`, end index `r`
8. Steps:
    a. Assign `String x = A[r]` (pivot element)
    b. Set `int i = p - 1` (pointer for elements less than pivot)
    c. For `j = p` to `j <= r - 1` do:
        i. If `A[j].compareToIgnoreCase(x) <= 0`:
            1. Increment `i` by 1 (`i = i + 1`)
            2. Swap `A[i]` with `A[j]`
    d. Swap `A[i + 1]` with `A[r]` to place the pivot in correct position
    e. Return `i + 1` (the pivot index)
9. Define the `main` function:
10. Steps:
    a. Accept the number of names `n` from the user
    b. Initialize array `A` of size `n`

c. For `i = 0` to `i < n`, accept each name and store in `A[i]`
d. Call `quickSort(A, 0, n - 1)` to sort the array
e. Print the sorted elements in array `A`

11. Stop

PROGRAM :

```
import java.util.Scanner;

class QuickSort {

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

       for (int j = p; j <= r - 1; j++) {
           if (A[j].compareToIgnoreCase(pivot) <= 0) {
               i++;

               String temp = A[i];
               A[i] = A[j];
               A[j] = temp;
           }
       }
       String temp = A[i + 1];
       A[i + 1] = A[r];
       A[r] = temp;

       return i + 1;
   }

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

       System.out.println("--------------");
       System.out.print("Enter the number of names: ");
       int n = sc.nextInt();
```

```
        sc.nextLine(); // Consume the newline
        String[] names = new String[n];
        System.out.println("Enter the names:");
        for (int i = 0; i < n; i++) {
            names[i] = sc.nextLine();
        }
        // Sort the names using Quick Sort
        quickSort(names, 0, n - 1);
        // Display sorted names
        System.out.println("\nNames in Ascending Order:");
        for (String name : names) {
            System.out.print(name + " ");
        }
        System.out.println("\n--------------");
        sc.close(); // Close the scanner
    }
}
```

OUTPUT :

Enter the number of names: 5
Enter the names:
GARGI
GAYATHRI
GANGA
ANUGRAHA
APARNA
Names in Ascending Order:
ANUGRAHA APARNA GANGA GARGI GAYATHRI

Gargi Chandrababu

Department of CSE

College of Engineering Kidangoor