EXERCISE 1

Date: 26/02/2024

AIM: -

Define a class 'product' with data members pcode, pname and price. Create 3 objects of the class and find the product having the lowest price.

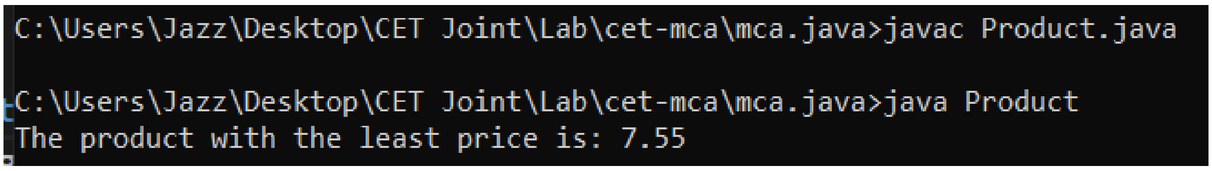
ALGORITHM: -

1. Define a class named "Product".
2. Declare a private instance variable named "price" of type double.
3. Create a constructor method named "Product" that takes a parameter "price" of type double and assigns it to the "price" instance variable.
4. Define a method named "getPrice" which returns the value of the "price" instance variable.
5. Define a static method named "comparePrices" that takes two parameters of type "Product" named "product1" and "product2", compares their prices, and returns the product with the smaller price.
6. Inside the main method:
    a. Create three instances of the "Product" class named "product1", "product2", and "product3" with prices 10.55, 15.75, and 7.55 respectively.
    b. Call the "comparePrices" method twice with "product2" and "product3" as arguments, and then pass the result and "product1" as arguments to another call of "comparePrices". Assign the result to a variable named "leastProduct".
    c. Print a message indicating the product with the least price, using the "getPrice" method of "leastProduct".

PROGRAM: -

```
public class Product{
  private double price;


  public Product(double price) {
    this.price = price;
  }


  public double getPrice() {
    return price;
  }
  public static Product comparePrices(Product product1, Product product2) {
    return product1.getPrice() < product2.getPrice() ? product1 : product2;
  }


  public static void main(String[] args) {
    Product product1 = new Product(10.55);
    Product product2 = new Product(15.75);
    Product product3 = new Product(7.55);


    Product leastProduct = Product.comparePrices(product1, Product.comparePrices(product2,
product3));
    System.out.println("The product with the least price is: " + leastProduct.getPrice());
  }
}
```

OUTPUT: -



```
C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>javac Product.java

C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>java Product
The product with the least price is: 7.55
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 2

Date: 26/02/2024

AIM: -

Read 2 matrices from the console and perform matrix addition.

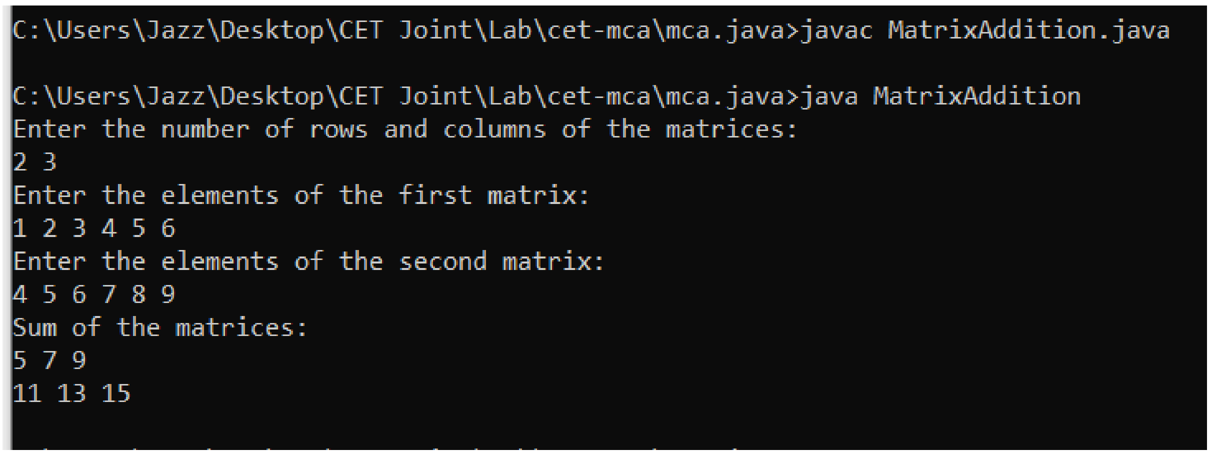
ALGORITHM: -

1. Create a class named "MatrixAddition" with a public static void main(String[] args) method.
2. Inside the main method, instantiate a Scanner object named "scanner" for user input.
3. Prompt the user to enter the number of rows and columns of the matrices.
4. Read the user input for rows and columns and store them in variables "rows" and "cols" respectively.
5. Create three 2D arrays named "matrix1," "matrix2," and "sum" with dimensions specified by "rows" and "cols."
6. Prompt the user to enter the elements of the first matrix and use nested loops to fill the "matrix1" array.
7. Prompt the user to enter the elements of the second matrix and use nested loops to fill the "matrix2" array.
8. Use nested loops to calculate the sum of the matrices and store the result in the "sum" array.
9. Display a message indicating the start of the output: "Sum of the matrices:"
10. Use nested loops to iterate through the "sum" array and print each element followed by a space.
11. After each row, print a newline character to move to the next row.
12. Close the Scanner object to release resources.

PROGRAM: -

```
 import java.util.Scanner;


public class MatrixAddition {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of rows and columns of the matrices:");
        int rows = scanner.nextInt();
        int cols = scanner.nextInt();
        int[][] matrix1 = new int[rows][cols];
        int[][] matrix2 = new int[rows][cols];
        int[][] sum = new int[rows][cols];

        System.out.println("Enter the elements of the first matrix:");
        for (int i = 0; i < rows; i++)
            for (int j = 0; j < cols; j++)
                matrix1[i][j] = scanner.nextInt();

        System.out.println("Enter the elements of the second matrix:");
        for (int i = 0; i < rows; i++)
            for (int j = 0; j < cols; j++)
                matrix2[i][j] = scanner.nextInt();

        for (int i = 0; i < rows; i++)
            for (int j = 0; j < cols; j++)
                sum[i][j] = matrix1[i][j] + matrix2[i][j];

System.out.println("Sum of the matrices:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++)
                System.out.print(sum[i][j] + " ");
            System.out.println();
        }
        scanner.close();
    }
}
```

OUTPUT: -



```
C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>javac MatrixAddition.java

C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>java MatrixAddition
Enter the number of rows and columns of the matrices:
2 3
Enter the elements of the first matrix:
1 2 3 4 5 6
Enter the elements of the second matrix:
4 5 6 7 8 9
Sum of the matrices:
5 7 9
11 13 15
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 3

Date: 26/02/2024

AIM: -

Add complex numbers.

ALGORITHM: -

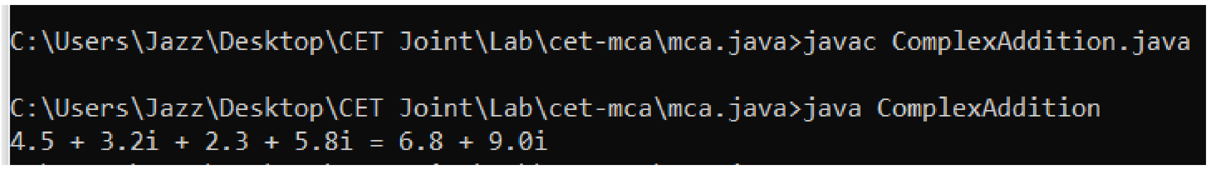
1. Define a class named "Complex" with instance variables "real" and "imaginary" of type double.
2. Implement a constructor "Complex" that takes two parameters "r" and "i" representing the real and imaginary parts of the complex number respectively.
3. Define a method "add" in the "Complex" class to add two complex numbers. It takes a Complex object "c" as a parameter, adds the real and imaginary parts separately, and returns a new Complex object representing the sum.
4. Implement a method "display" in the "Complex" class to print the complex number in the form "real + imaginary*i".
5. Define a public class "ComplexAddition" with the main method.
6. Inside the main method:
    a. Create two Complex objects "num1" and "num2" with the provided real and imaginary parts.
    b. Add "num1" and "num2" using the "add" method, and store the result in a Complex object "sum".
    c. Display "num1", "num2", and "sum" using the "display" method to print their respective complex representations along with the plus and equals signs.

PROGRAM: -

```
class Complex {

  double real;

  double imaginary;

  Complex(double r, double i) {

    real = r;

    imaginary = i;

  }

  Complex add(Complex c) {

    double realPart = this.real + c.real;

    double imaginaryPart = this.imaginary + c.imaginary;
```

```
        return new Complex(realPart, imaginaryPart);
    }
    void display() {
        System.out.print(real + " + " + imaginary + "i");
    }
}
public class ComplexAddition {
    public static void main(String[] args) {
        Complex num1 = new Complex(4.5, 3.2);
        Complex num2 = new Complex(2.3, 5.8);

        Complex sum = num1.add(num2);

        num1.display();
        System.out.print(" + ");
        num2.display();
        System.out.print(" = ");
        sum.display();
    }
}
```

OUTPUT: -



```
C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>javac ComplexAddition.java

C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>java ComplexAddition
4.5 + 3.2i + 2.3 + 5.8i = 6.8 + 9.0i
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 4

Date: 26/02/2024

AIM: -

Read a matrix from the console and check whether it is symmetric or not.

ALGORITHM: -

1. Import the Scanner class from the java.util package.
2. Define a class named "SymmetricMatrixCheck".
3. Inside the class, define the main method.
4. Create a Scanner object named "scanner" to read user input from the console.
5. Prompt the user to input the number of rows and columns for the matrix.
6. Read the input for rows and columns and store them in variables "rows" and "cols" respectively.
7. Create a 2D array named "matrix" with dimensions specified by "rows" and "cols".
8. Prompt the user to input the elements of the matrix and use nested loops to fill the "matrix" array.
9. Initialize a boolean variable "isSymmetric" to true.
10. Check if the number of rows is not equal to the number of columns, if so, set "isSymmetric" to false.
11. If the number of rows equals the number of columns:
    a. Use nested loops to compare each element of the matrix with its corresponding element across the diagonal.
    b. If any pair of corresponding elements does not match, set "isSymmetric" to false and break out of the loop.
12. If "isSymmetric" is still true after the comparisons, print "The matrix is symmetric." Otherwise, print "The matrix is not symmetric."
13. Close the Scanner object to release resources.

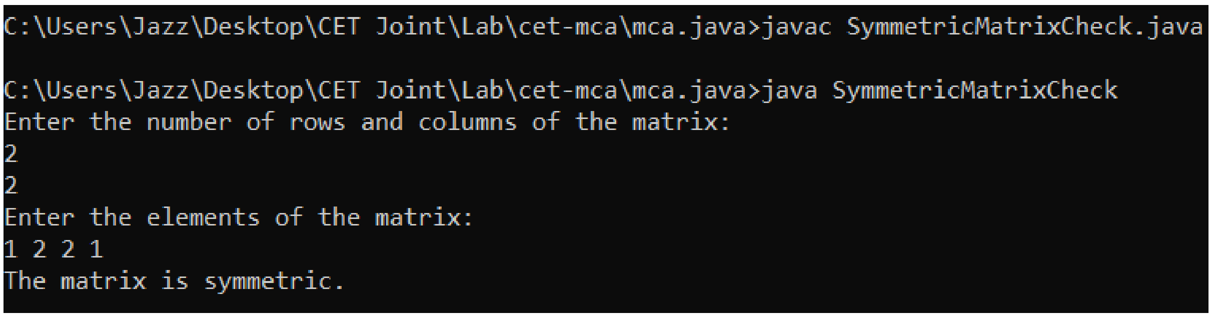
PROGRAM: -

```
import java.util.Scanner;


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

```
System.out.println("Enter the number of rows and columns of the matrix:");
int rows = scanner.nextInt();
int cols = scanner.nextInt();

int[][] matrix = new int[rows][cols];

System.out.println("Enter the elements of the matrix:");
for (int i = 0; i < rows; i++)
   for (int j = 0; j < cols; j++)
      matrix[i][j] = scanner.nextInt();

boolean isSymmetric = true;
if (rows != cols) {
   isSymmetric = false;
} else {
   for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
         if (matrix[i][j] != matrix[j][i]) {
            isSymmetric = false;
            break;
         }
      }
      if (!isSymmetric) {
         break;
      }
   }
}

if (isSymmetric) {
```

```
            System.out.println("The matrix is symmetric.");
        } else {
            System.out.println("The matrix is not symmetric.");
        }
        scanner.close();
    }
}
```

OUTPUT: -



```
C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>javac SymmetricMatrixCheck.java

C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>java SymmetricMatrixCheck
Enter the number of rows and columns of the matrix:
2
2
Enter the elements of the matrix:
1 2 2 1
The matrix is symmetric.
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 5

Date: 26/02/2024

AIM: -

Create CPU with attribute price. Create inner class Processor (no. of cores, manufacturer) and static nested class RAM (memory, manufacturer). Create an object of CPU and print information of Processor and RAM.

ALGORITHM: -

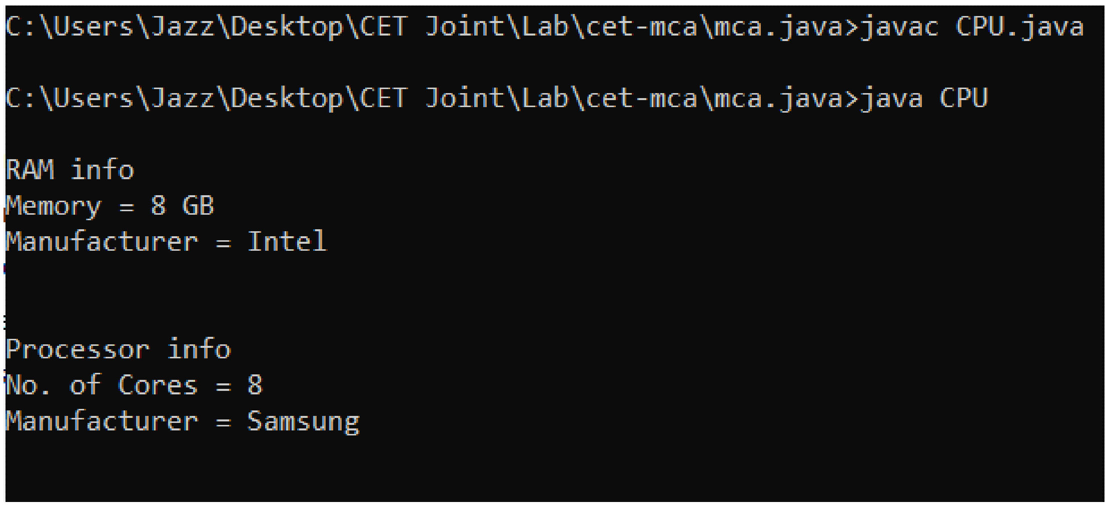
1. Define a class named "CPU" with an instance variable "price" of type int.
2. Inside "CPU", define an inner class "Processor" with instance variables "cores" (int) and "producer" (String).
3. Define a constructor for "Processor" initializing its fields.
4. Implement a method "display" in "Processor" to print core count and manufacturer.
5. Define a static nested class "RAM" in "CPU" with "mem" (int) and "manuf" (String) instance variables.
6. Create a constructor for "RAM" to set its properties.
7. Implement a method "display" in "RAM" to print memory size and manufacturer.
8. In the "main" method:
    a. Instantiate "obj1" of "RAM" with 8GB memory and "Intel" manufacturer.
    b. Create "obj2" of "CPU".
    c. Instantiate "obj3" of "Processor" with 8 cores and "Samsung" manufacturer using "obj2.new Processor()".
    d. Call "display" for both "obj1" and "obj3" to show their information.

PROGRAM: -

```
public class CPU {
  int price;
  class Processor {
    int cores;
    String producer;
    Processor(int noC, String manu) {
      cores = noC;
      producer = manu;
    }
```

```
        void display() {
            System.out.println("\nProcessor info");
            System.out.println("No. of Cores = " + cores);
            System.out.println("Manufacturer = " + producer + "\n");
        }
    }
    static class RAM {
        int mem;
        String manuf;
        RAM(int memory, String producer) {
            mem = memory;
            manuf = producer;
        }
        void display() {
            System.out.println("\nRAM info");
            System.out.println("Memory = " + mem + " GB");
            System.out.println("Manufacturer = " + manuf + "\n");
        }
    }
    public static void main(String[] args) {
        CPU.RAM obj1 = new CPU.RAM(8, "Intel");
        CPU obj2 = new CPU();
        CPU.Processor obj3 = obj2.new Processor(8, "Samsung");
        obj1.display();
        obj3.display();
    }
}
```

OUTPUT: -



```
C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>javac CPU.java

C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>java CPU

RAM info
Memory = 8 GB
Manufacturer = Intel


Processor info
No. of Cores = 8
Manufacturer = Samsung
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 6

Date: 20/03/2024

AIM: -

Search an element in an array.

ALGORITHM: -

```
1. Import the Scanner class from java.util.
2. Define a class named "array" with instance variables "sc" (Scanner), "arr" (array of
   integers), and "size" (integer).
3. Implement a constructor "array" that takes an integer "n" as a parameter to initialize the
   size of the array.
      a. Initialize the "size" variable with the given parameter.
      b. Initialize the "arr" array with size "n".
      c. Prompt the user to input elements for the array using Scanner.
      d. Use a for loop to iterate through the array and store the input elements.
4. Implement a method "search" in the "array" class that takes an integer "n" as a
   parameter to search for it in the array.
      a. Initialize a variable "c" to track if the number is found.
      b. Use a for loop to iterate through the array and check if each element is equal to
         "n".
      c. If "n" is found, set "c" to 1, print a message indicating that "n" is found, and
         break out of the loop.
      d. If "n" is not found, print a message indicating that "n" is not an element of the
         array.
5. Define a class named "search_array" with the main method.
6. Inside the main method:
      a. Create a Scanner object named "sc" to read user input.
      b. Prompt the user to input the number of elements in the array.
      c. Create an object "a" of the "array" class with the specified number of elements.
      d. Prompt the user to input the number to be searched.
      e. Call the "search" method of the "a" object with the user-input number.
```

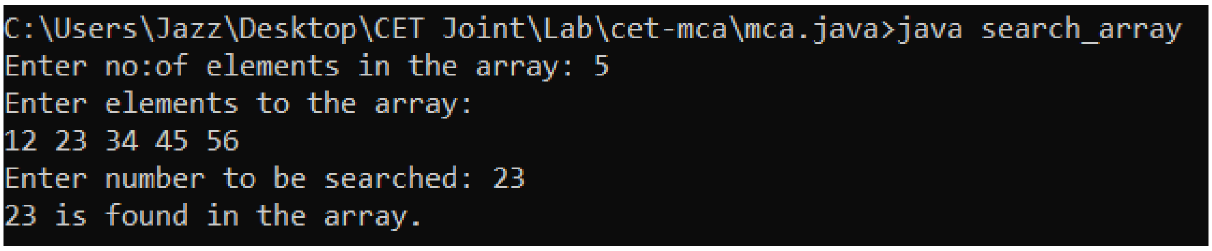
PROGRAM: -

```
import java.util.*;

class array

{

   Scanner sc=new Scanner(System.in);

   int arr[],size;
```

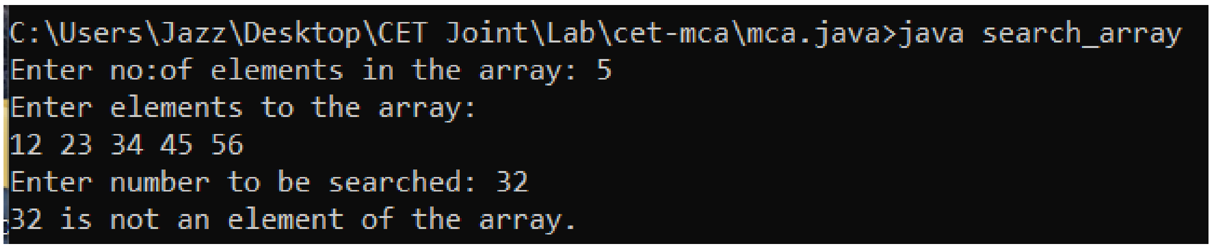
```
  array(int n)
  {
    size=n;
    arr=new int[n];
    System.out.println("Enter elements to the array: ");
    for(int i=0;i<n;i++)
      arr[i]=sc.nextInt();
  }
  void search(int n)
  {
    int c=0;
    for(int i=0;i<size;i++)
      if(arr[i]==n)
      {
        c=1;
        System.out.println(n+" is found in the array.");
        break;
      }
    if(c==0)
      System.out.println(n+" is not an element of the array.");
  }
}
class search_array
{
  public static void main(String[] args)
  {
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter no:of elements in the array: ");
    int n=sc.nextInt();
```

```
        array a=new array(n);

        System.out.print("Enter number to be searched: ");

        int num=sc.nextInt();

        a.search(num);

    }

}
```

OUTPUT: -



```
C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>java search_array
Enter no:of elements in the array: 5
Enter elements to the array:
12 23 34 45 56
Enter number to be searched: 23
23 is found in the array.
```



```
C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>java search_array
Enter no:of elements in the array: 5
Enter elements to the array:
12 23 34 45 56
Enter number to be searched: 32
32 is not an element of the array.
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 7

Date: 27/03/2024

AIM: -

Perform string manipulations.

ALGORITHM: -

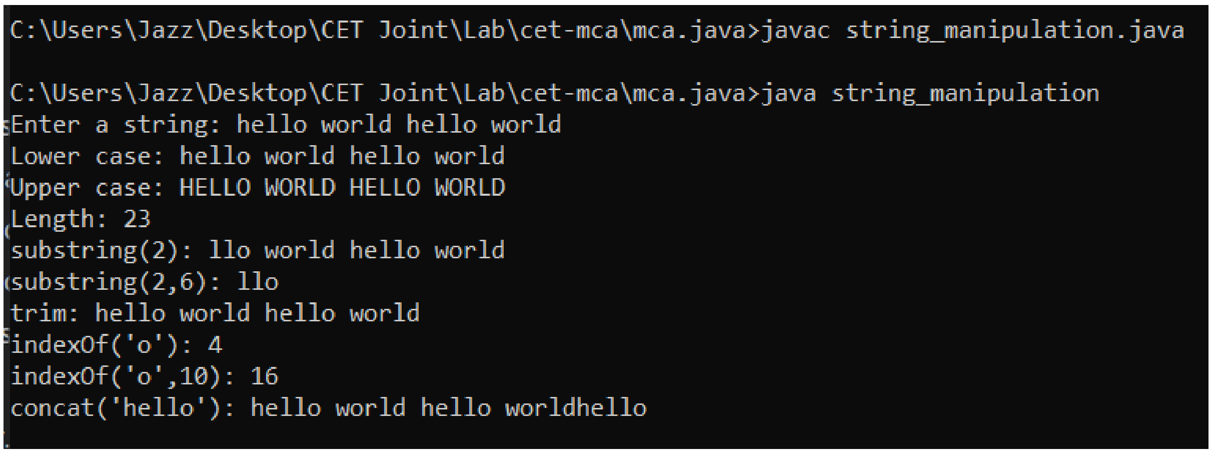
```
1. Import the Scanner class from java.util.
2. Define a class named "word" with instance variables "sc" (Scanner) and "s" (String).
3. Implement a constructor "word" without parameters:
    a. Initialize the Scanner object "sc".
    b. Prompt for and store a string in "s".
4. Implement a method "str_functions" in the "word" class without parameters:
    a. Print lowercase and uppercase versions of "s".
    b. Print the length of "s".
    c. Print substrings of "s" starting from index 2 and from index 2 to 5.
    d. Print "s" after trimming leading and trailing whitespaces.
    e. Print the index of the first occurrence of 'o' in "s" and from index 10.
    f. Concatenate "s" with "CR7" and print the result.
5. Define a class named "string_manipulation" with the main method:
    a. Create an object "w" of the "word" class.
    b. Call the "str_functions" method of "w".
```

PROGRAM: -

```
import java.util.*;

class word{

  Scanner sc=new Scanner(System.in);

  String s;

  word(){

    System.out.print("Enter a string: ");

    s=sc.nextLine();
  }
  void str_functions() {
    System.out.println("Lower case: "+s.toLowerCase());
    System.out.println("Upper case: "+s.toUpperCase());
    System.out.println("Length: "+s.length());
    System.out.println("substring(2): "+s.substring(2));
    System.out.println("substring(2,6): "+s.substring(2,6));
    System.out.println("trim: "+s.trim());
    System.out.println("indexOf('o'): "+s.indexOf('o'));
    System.out.println("indexOf('o',10): "+s.indexOf('o',10));
    System.out.println("concat('hello'): "+s.concat("hello"));
  }
}
class string_manipulation {
  public static void main(String[] args) {
    word w=new word();
    w.str_functions();
  }
}
```

```
class string_manipulation {
    public static void main(String[] args) {
        word w=new word();
        w.str_functions();
    }
}
```

OUTPUT: -



```
C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>javac string_manipulation.java

C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>java string_manipulation
Enter a string: hello world hello world
Lower case: hello world hello world
Upper case: HELLO WORLD HELLO WORLD
Length: 23
substring(2): llo world hello world
substring(2,6): llo
trim: hello world hello world
indexOf('o'): 4
indexOf('o',10): 16
concat('hello'): hello world hello worldhello
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 8

Date: 27/03/2024

AIM: -

Program to create a class for Employee having attributes eNo, eName, eSalary. Read n employee information and Search for an employee given eNo, using the concept of Array of Objects.

ALGORITHM: -

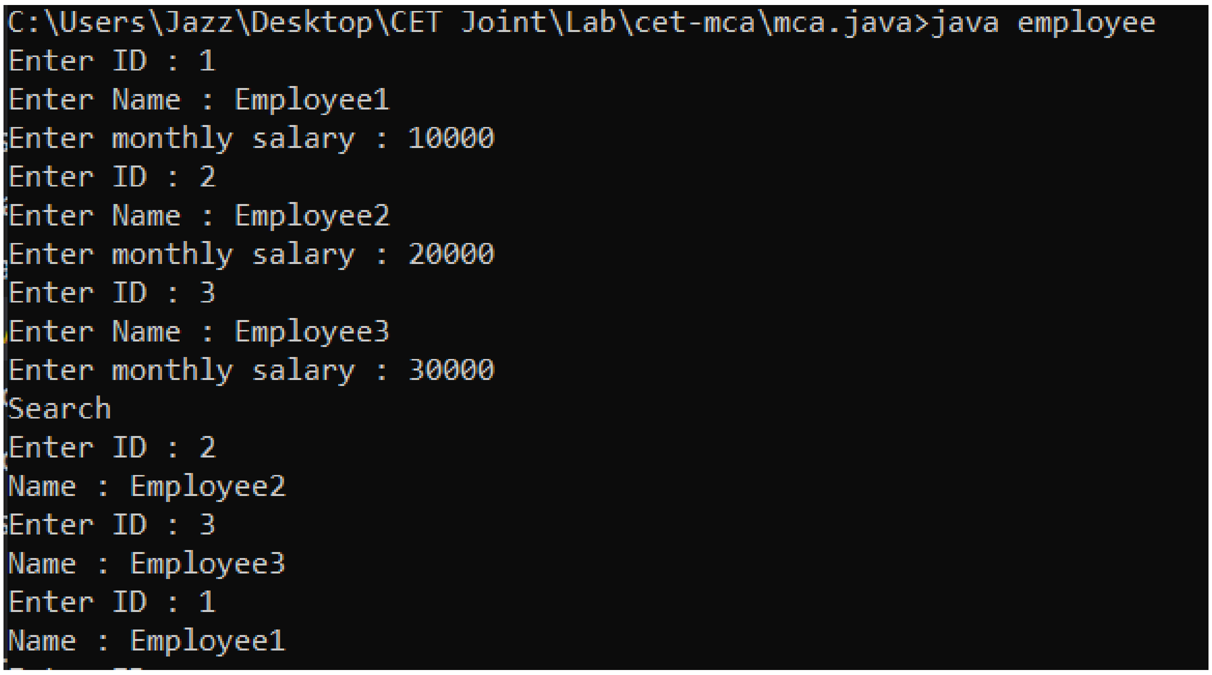
1. Import Scanner from java.util.
2. Define a class "employee" with integer variables "eNo" and "eSalary", and a String variable "eName".
3. Implement a method "read" to input employee details:
    a. Prompt and read ID, name, and monthly salary.
4. Implement a method "display" to print the employee name.
5. Define the main method:
    a. Initialize integer variables "i" and "n".
    b. Create an array "emp" of size "n".
    c. Input details for each employee.
    d. Continuously prompt for an employee ID to search.
    e. If found, display the employee name.

PROGRAM : -

```
import java.util.*;

class employee {

  int eNo;

  String eName;

  int eSalary;


  public void read(){

    Scanner sc= new Scanner(System.in);

    System.out.print("Enter ID : ");

    eNo = Integer.parseInt(sc.nextLine());

    System.out.print("Enter Name : ");
```

```
        eName = sc.nextLine();
        System.out.print("Enter monthly salary : ");
        eSalary = Integer.parseInt(sc.nextLine());
    }
    public void display(){
        System.out.println("Name : "+ eName );
    }
    public static void main(String []args){
        int i,n=3;
        int No;
        employee emp[] = new employee[n];
        for(i=0;i<n;i++){
            emp[i] = new employee();
            emp[i].read();
        }
        System.out.println("Search");
        while(true){
            Scanner sc= new Scanner(System.in);
            System.out.print("Enter ID : ");
            No = Integer.parseInt(sc.nextLine());
            for(i=0;i<n;i++){
                if(emp[i].eNo == No){
                    emp[i].display();
                    break;
                }
            }
        }
    }
}
```

OUTPUT: -



```
C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>java employee
Enter ID : 1
Enter Name : Employee1
Enter monthly salary : 10000
Enter ID : 2
Enter Name : Employee2
Enter monthly salary : 20000
Enter ID : 3
Enter Name : Employee3
Enter monthly salary : 30000
Search
Enter ID : 2
Name : Employee2
Enter ID : 3
Name : Employee3
Enter ID : 1
Name : Employee1
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 9

Date: 02/04/2024

AIM: -

Write a Java program to sort strings.

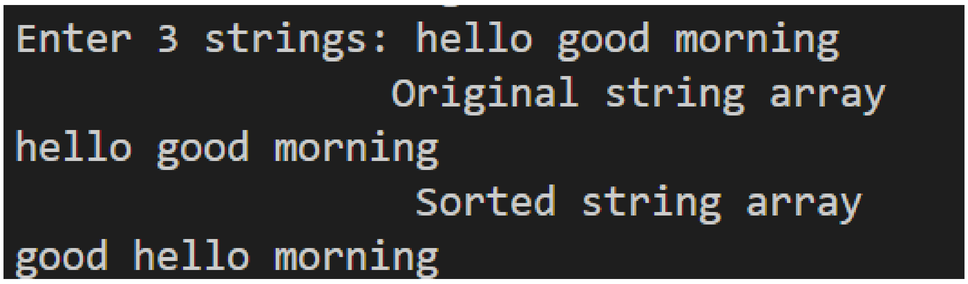
ALGORITHM: -

1. Create a class named **sort_string.** Declare instance variables **s[ ]** (array of stringd) and **size.**
2. Define a constructor to read and **size** and **strings.**
3. Prompt the user to enter the number of strings (**size**), and store the stings into the array.
4. Define a method **sort():**
    - Outer loop iterates over each elements of the array.
    - Inner loop iterates over the unsorted portion of the array.
    - Compare adjacent elements of the array and swap then if they are in the wrong order.
    - Repeat the process until the entire array is sorted.
5. Define method **display()** to display the array elements.
6. In the main method, create an object of the **sort_string class.**
7. Display original array.
8. Sort the array using **sort().**
9. Display the sorted string array.

PROGRAM : -

```
import java.util.*;
class sort_string{
  Scanner sc=new Scanner(System.in);
  String s[];
  int size;
  sort_string() {
    System.out.print("Enter no: of strings: ");
    size=sc.nextInt();
    s=new String[size];
    System.out.print("Enter "+size+" strings: ");
    for(int i=0;i<size;i++)
      s[i]=sc.next();
  }
  void sort(){
        for(int i=0;i<size;i++)
            for(int j=0;j<size-i-1;j++)
                if (s[j].compareTo(s[j+1])>0){
                    String temp=s[j];
                    s[j]=s[j+1];
                    s[j+1]=temp;
                }
    }
    void display(){
        for(int i=0;i<size;i++)
            System.out.print(s[i]+" ");
    }
}
public class sort_string_main {
    public static void main(String[] args) {
        sort_string s1=new sort_string();
        System.out.println("\t\tOriginal string array");
        s1.display();
        s1.sort();
        System.out.println("\n\t\t Sorted string array");
        s1.display();
    }
}
```

OUTPUT: -



```
Enter 3 strings: hello good morning
                Original string array
hello good morning
                 Sorted string array
good hello morning
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 10

Date: 02/04/2024

AIM: -

Program to calculate area of different shapes using overloaded functions.

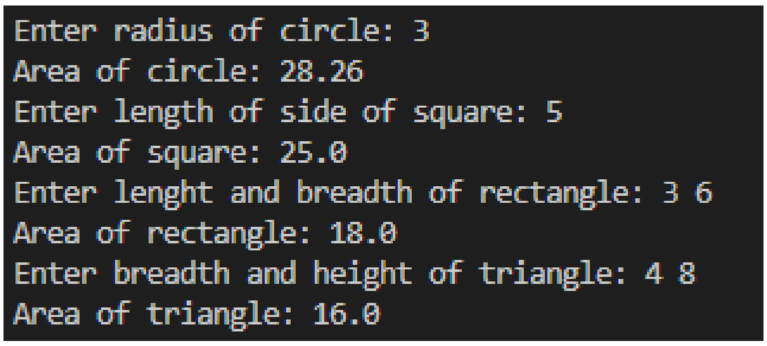
ALGORITHM: -

1. Create a class named **shape_area**, with the following methods.
   - **void area(double r)**: Calculates and prints the area of a circle using the formula: π * r * r.
   - **void area(float r)**: Calculates and prints the area of a square using the formula: side * side.
   - **void area(double l, double b)**: Calculates and prints the area of a rectangle using the formula: length * breadth.
   - **void area(float b, float h)**: Calculates and prints the area of a triangle using the formula: 0.5 * base * height.
2. In the **main** method:
   - Create an object of class **shape_area**.
   - Prompt the user to enter the radius of the circle and call the **area(double r)** method with the provided radius.
   - Prompt the user to enter the length of the side of the square and call the **area(float r)** method with the provided side length.
   - Prompt the user to enter the length and breadth of the rectangle and call the **area(double l, double b)** method with these values.
   - Prompt the user to enter the base and height of the triangle and call the **area(float b, float h)** method with these values.

PROGRAM : -

```
import java.util.*;
class shape_area{
  void area(double r){
    System.out.println("Area of circle: "+String.format("%.2f", 3.14*r*r));
  }
  void area(float r){
    System.out.println("Area of square: "+(r*r));
  }
  void area(double l,double b){
    System.out.println("Area of rectangle: "+(l*b));
  }
  void area(float b,float h){
    System.out.println("Area of triangle: "+(0.5*b*h));
  }
}
class area_main{
  public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    shape_area s =new shape_area();
    System.out.print("Enter radius of circle: ");
    s.area(sc.nextDouble());

    System.out.print("Enter length of side of square: ");
    s.area(sc.nextFloat());

    System.out.print("Enter lenght and breadth of rectangle: ");
    double a=sc.nextDouble();
    double b=sc.nextDouble();
    s.area(a,b);

    System.out.print("Enter breadth and height of triangle: ");
    float a1=sc.nextFloat();
    float b1=sc.nextFloat();
    s.area(a1,b1);
  }
}
```

OUTPUT: -



```
Enter radius of circle: 3
Area of circle: 28.26
Enter length of side of square: 5
Area of square: 25.0
Enter lenght and breadth of rectangle: 3 6
Area of rectangle: 18.0
Enter breadth and height of triangle: 4 8
Area of triangle: 16.0
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 11

Date: 02/04/2024

AIM: -

Create a class ‘Employee’ with data members Empid, Name, Salary, Address and constructors to initialize the data members. Create another class ‘Teacher’ that inherit the properties of class employee and contain its own data members department, Subjects taught and constructors to initialize these data members and also include display function to display all the data members. Use array of objects to display details of N teachers.

ALGORITHM: -

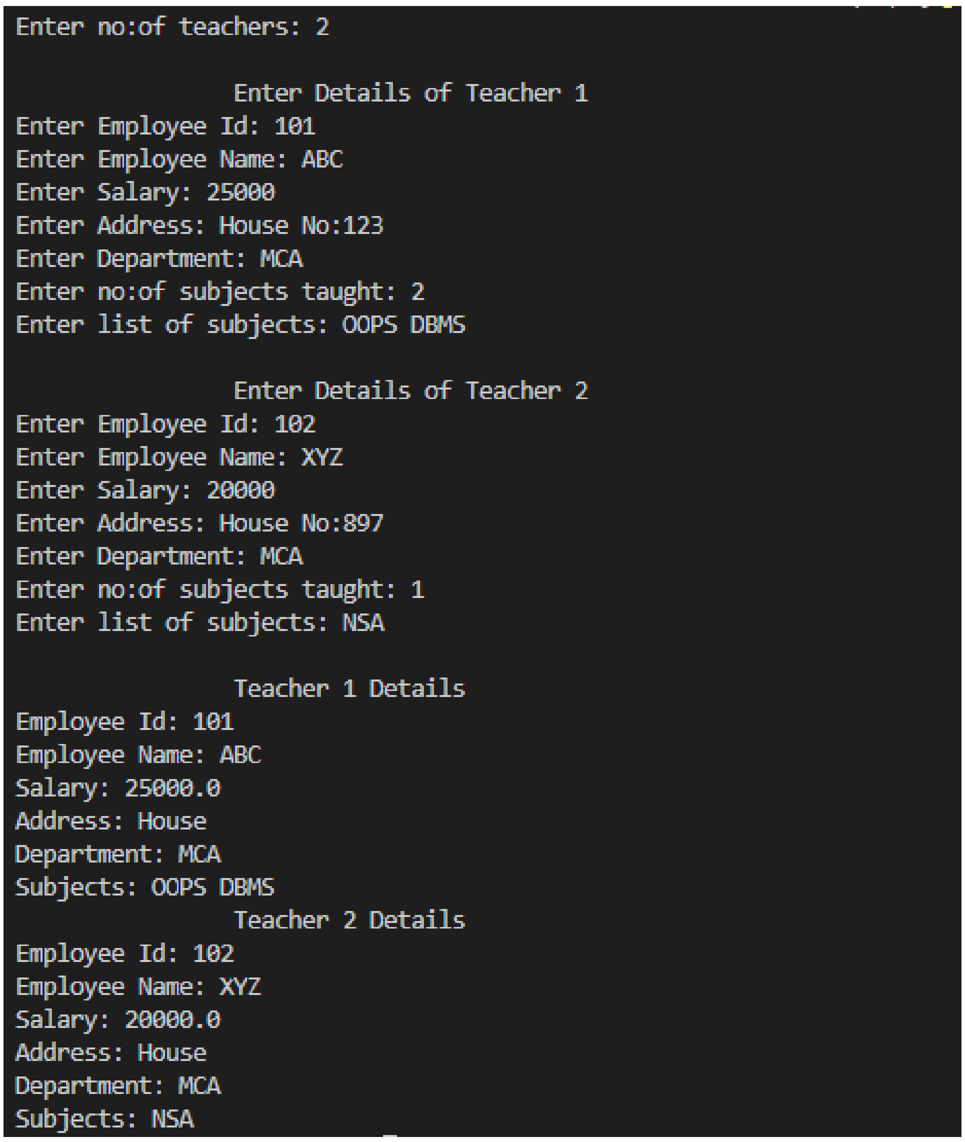
1. Create a class named **Employee.**
    - Declare instance variables **Empid**, **name**, **address**, and **salary.**
    - Define a constructor that initializes the employee details.
2. Define a class **Teacher** inherited from **Employee.**
    - Declare additional instance variable **dept**, **subject[]**, and **no(**number of subjects taught.
    - Define a constructor that initializes teacher details by calling superclass constructors and then initialize the instance variable of class **Teacher.**
    - Define a method **display()** to display the details of the teacher.
3. In the **main** method:
    - Prompt the user to enter number of teachers **(n).**
    - Create an array **t[]** of type **Teacher** with size **n.**
    - Using a loop instantiate each **Teacher** object, prompting the user to enter details for each teacher.
    - Use another loop to display the details of each teacher by calling the **display()** method.

PROGRAM : -

```
import java.util.*;
class Employee
{
  Scanner sc=new Scanner(System.in);
  int Empid;
  String name,address;
  double salary;
  Employee(int x)
{
    System.out.println("\n\t\tEnter Details of Teacher "+x);
```

```
        System.out.print("Enter Employee Id: ");
        Empid=sc.nextInt();
        System.out.print("Enter Employee Name: ");
        name=sc.next();
        System.out.print("Enter Salary: ");
        salary=sc.nextDouble();
        System.out.print("Enter Address: ");
        address=sc.next();
    }
}
class Teacher extends Employee
{
    Scanner sc=new Scanner(System.in);
    String dept,subject[];
    int no;
    Teacher(int x)
 {
        super(x);
        System.out.print("Enter Department: ");
        dept=sc.next();
        System.out.print("Enter no:of subjects taught: ");
        no=sc.nextInt();
        subject=new String[no];
        System.err.print("Enter list of subjects: ");
        for(int i=0;i<no;i++)
            subject[i]=sc.next();
    }
    void display(int j)
{
        System.out.println("\n\t\tTeacher "+j+" Details");
        System.out.println("Employee Id: "+Empid);
        System.out.println("Employee Name: "+name);
        System.out.println("Salary: "+salary);
        System.out.println("Address: "+address);
        System.out.println("Department: "+dept);
        System.out.print("Subjects: ");
        for(int i=0;i<no;i++)
            System.out.print(subject[i]+" ");
    }
}
class teacher_main
{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter no:of teachers: ");
        int n=sc.nextInt();
        Teacher t[]= new Teacher[n];
        for(int i=0;i<n;i++)
            t[i]=new Teacher(i+1);
        for(int i=0;i<n;i++)
            t[i].display(i+1);
    }
}
```

OUTPUT: -



```
Enter no:of teachers: 2

                Enter Details of Teacher 1
Enter Employee Id: 101
Enter Employee Name: ABC
Enter Salary: 25000
Enter Address: House No:123
Enter Department: MCA
Enter no:of subjects taught: 2
Enter list of subjects: OOPS DBMS

                Enter Details of Teacher 2
Enter Employee Id: 102
Enter Employee Name: XYZ
Enter Salary: 20000
Enter Address: House No:897
Enter Department: MCA
Enter no:of subjects taught: 1
Enter list of subjects: NSA

                Teacher 1 Details
Employee Id: 101
Employee Name: ABC
Salary: 25000.0
Address: House
Department: MCA
Subjects: OOPS DBMS
                Teacher 2 Details
Employee Id: 102
Employee Name: XYZ
Salary: 20000.0
Address: House
Department: MCA
Subjects: NSA
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 12

Date: 09/04/2024

AIM: -

Create a class ‘Person’ with data members Name, Gender, Address, Age and a constructor to initialize the data members and another class ‘Employee’ that inherits the properties of class Person and also contains its own data members like Empid, Company_name, Qualification, Salary and its own constructor. Create another class ‘Teacher’ that inherits the properties of class Employee and contains its own data members like Subject, Department, Teacherid and also contain constructors and methods to display the data members. Use array of objects to display details of N teachers.

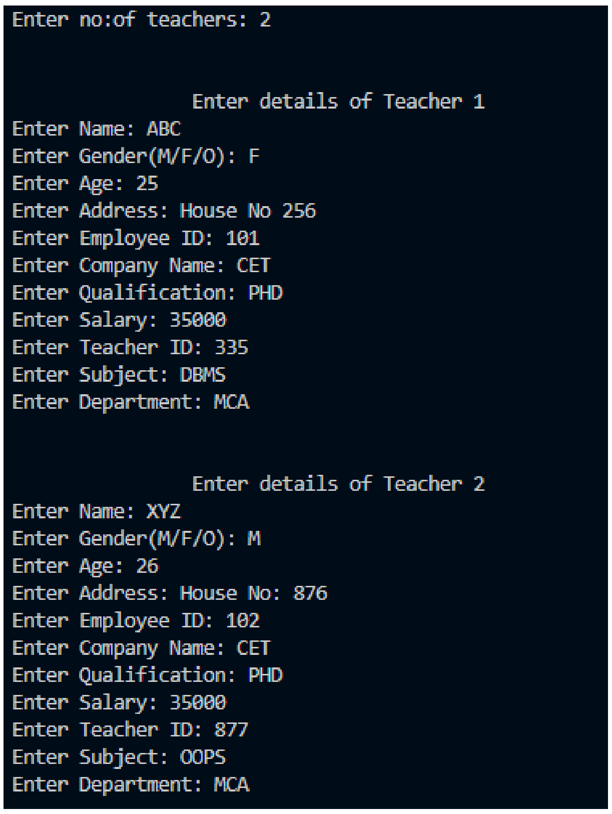
ALGORITHM: -

1. Create a class named **Person.**
   - Declare instance variables **name, address, gender,** and **age.**
   - Define a constructor to initialize the instance variables.
2. Declare a class **Employee** inherited from **Person.**
   - Declare instance variables **EmpId, CompanyName, qualification,** and **salary.**
   - Define a constructor to call the superclass constructor and initialize employee details.
3. Declare a class **Teacher** inherited from **Employee.**
   - Declare instance variables **subject, department,** and **TeacherID.**
   - Define a constructor to call the superclass constructor and initialize teacher details.
   - Define a **display()** method to display the teacher details.
4. In the **main** method:
   - Prompt the user to enter number of teachers **(n).**
   - Create an array **t[]** of type **Teacher** with size **n.**
   - Using a loop instantiate each **Teacher** object, prompting the user to enter details for each teacher.
   - Use another loop to display the details of each teacher by calling the **display()** method.

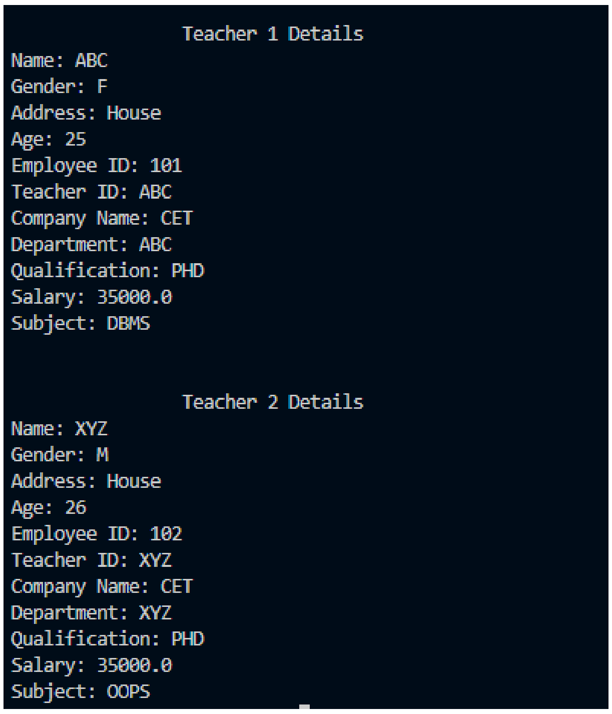
PROGRAM : -

```
import java.util.*;
class Person{
  Scanner sc=new Scanner(System.in);
  String name,address;
  char gender;
  int age;
  Person(int x){
        System.out.println("\n\n\t\tEnter details of Teacher "+x);
        System.out.print("Enter Name: ");
        name=sc.next();
        System.out.print("Enter Gender(M/F/O): ");
        gender=sc.next().charAt(0);
        System.out.print("Enter Age: ");
        age=sc.nextInt();
        System.out.print("Enter Address: ");
        address=sc.next();
    }
}
class Employee extends Person{
    Scanner sc=new Scanner(System.in);
    int EmpId;
    String CompanyName,qualification;
    double salary;
    Employee(int x){
        super(x);
        System.out.print("Enter Employee ID: ");
        EmpId=sc.nextInt();
        System.out.print("Enter Company Name: ");
        CompanyName=sc.next();
        System.out.print("Enter Qualification: ");
        qualification=sc.next();
        System.out.print("Enter Salary: ");
        salary=sc.nextDouble();
    }
}
class Teacher extends Employee{
    Scanner sc=new Scanner(System.in);
    String subject,department;
    int TeacherId;
    Teacher(int x){
        super(x);
        System.out.print("Enter Teacher ID: ");
        TeacherId=sc.nextInt();
        System.out.print("Enter Subject: ");
        subject=sc.next();
        System.out.print("Enter Department: ");
        department=sc.next();
    }
    void display(int n){
        System.out.println("\n\n\t\tTeacher "+n+" Details");
        System.out.println("Name: "+name);
        System.out.println("Gender: "+gender);
        System.out.println("Address: "+address);
        System.out.println("Age: "+age);
        System.out.println("Employee ID: "+EmpId);
        System.out.println("Teacher ID: "+name);
        System.out.println("Company Name: "+CompanyName);
        System.out.println("Department: "+name);
        System.out.println("Qualification: "+qualification);
        System.out.println("Salary: "+salary);
        System.out.println("Subject: "+subject);
    }
}
class teacher_main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter no:of teachers: ");
        int n =sc.nextInt();
        Teacher t[]=new Teacher[n];
        for(int i=0;i<n;i++)
            t[i]=new Teacher(i+1);
        for(int i=0;i<n;i++)
            t[i].display(i+1);
    }
}
```

OUTPUT: -



```
Enter no:of teachers: 2


                Enter details of Teacher 1
Enter Name: ABC
Enter Gender(M/F/O): F
Enter Age: 25
Enter Address: House No 256
Enter Employee ID: 101
Enter Company Name: CET
Enter Qualification: PHD
Enter Salary: 35000
Enter Teacher ID: 335
Enter Subject: DBMS
Enter Department: MCA


                Enter details of Teacher 2
Enter Name: XYZ
Enter Gender(M/F/O): M
Enter Age: 26
Enter Address: House No: 876
Enter Employee ID: 102
Enter Company Name: CET
Enter Qualification: PHD
Enter Salary: 35000
Enter Teacher ID: 877
Enter Subject: OOPS
Enter Department: MCA
```



```
               Teacher 1 Details
Name: ABC
Gender: F
Address: House
Age: 25
Employee ID: 101
Teacher ID: ABC
Company Name: CET
Department: ABC
Qualification: PHD
Salary: 35000.0
Subject: DBMS


               Teacher 2 Details
Name: XYZ
Gender: M
Address: House
Age: 26
Employee ID: 102
Teacher ID: XYZ
Company Name: CET
Department: XYZ
Qualification: PHD
Salary: 35000.0
Subject: OOPS
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 13

Date: 09/04/2024

AIM: -

Create an interface having prototypes of functions area() and perimeter(). Create two classes Circle and Rectangle which implements the above interface. Create a menu driven program to find area and perimeter of objects.

ALGORITHM: -

1. Define the **Shape** interface:
   - Declare two abstract methods **area**() and **perimeter**().
2. Define the **Circle** class:
   - Declare a double variable **radius**.
   - Implement a constructor that takes the **radius** as a parameter and assigns it to the **radius** member variable.
   - Implement the **area**() method to calculate and print the area of the circle.
   - Implement the **perimeter**() method to calculate and print the perimeter of the circle.
3. Define the **Rectangle** class:
   - Declare double variables **length** and **breadth**.
   - Implement a constructor that takes the **length** and **breadth** as parameters and assigns them to the respective member variables.
   - Implement the **area**() method to calculate and print the area of the rectangle.
   - Implement the **perimeter**() method to calculate and print the perimeter of the rectangle.
4. In the **main** method:
   - Declare an integer variable **ch** to store the user's choice.
   - Start a **do-while loop** to display a menu and execute the corresponding choice until the user chooses to exit.
   - Inside the loop, display a menu for the user to choose between Circle, Rectangle, or Exit.
   - If the user chooses **Circle**:
     - Prompt the user to enter the **radius** of the circle.
     - Create a Circle object with the given radius.
     - Call the **area**() and **perimeter**() methods of the Circle object.
   - If the user chooses Rectangle:
     - Prompt the user to enter the **length** and **breadth** of the rectangle.
     - Create a Rectangle object with the given length and breadth.
     - Call the **area**() and **perimeter**() methods of the Rectangle object.
   - If the user chooses Exit:
     - Print "Exiting..." and exit the loop.
   - If the user enters an **invalid choice**, display "Invalid choice!".
   - Repeat the loop until the user chooses to exit (option 3).

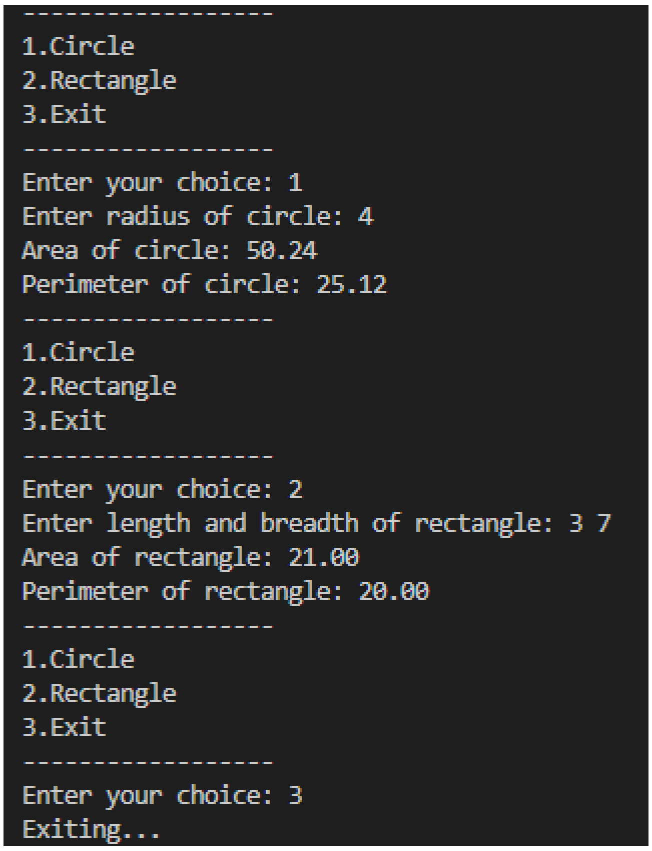
PROGRAM : -

```
import java.util.*;
interface Shape{
   public void area();
   public void perimeter();
}
class Circle implements Shape{
   double radius;
   Circle(double r){
      radius = r;
   }
   public void area(){
      System.out.println("Area of circle: " + String.format("%.2f", (3.14*radius*radius)));
   }
   public void perimeter(){
      System.out.println("Perimeter of circle: " + String.format( "%.2f",(2*3.14*radius)));
   }
}
class Rectangle implements Shape{
   double length,breadth;
   Rectangle(double l,double b){
      length = l;
      breadth = b;
   }
   public void area(){
      System.out.println("Area of rectangle: " + String.format("%.2f" ,(length*breadth)));
   }
   public void perimeter(){
      System.out.println("Perimeter of rectangle: " + String.format(
"%.2f",(2*(length+breadth))));
   }
```

```
}
class circle_rect {
   public static void main(String[] args){
      Scanner sc=new Scanner(System.in);
      int ch;
      do{
         System.out.print("------------------\n1.Circle\n2.Rectangle \n3.Exit\n------------------
\nEnter your choice: ");
         ch=sc.nextInt();
         switch(ch){
            case 1:
                  System.out.print("Enter radius of circle: ");
                  double r=sc.nextDouble();
                  Circle c =new Circle(r);
                  c.area();
                  c.perimeter();
                  break;
            case 2:
                  System.out.print("Enter length and breadth of rectangle: ");
                  double l=sc.nextDouble();
                  double b=sc.nextDouble();
                  Rectangle r1 =new Rectangle(l,b);
                  r1.area();
                  r1.perimeter();
                  break;
            case 3:
                  System.out.println("Exiting...");
                  break;
            default:
                  System.out.println("Invalid choice!");
         }
```

```
        }while(ch!=3);
    }
}
```

OUTPUT: -



```
------------------
1.Circle
2.Rectangle
3.Exit
------------------
Enter your choice: 1
Enter radius of circle: 4
Area of circle: 50.24
Perimeter of circle: 25.12
------------------
1.Circle
2.Rectangle
3.Exit
------------------
Enter your choice: 2
Enter length and breadth of rectangle: 3 7
Area of rectangle: 21.00
Perimeter of rectangle: 20.00
------------------
1.Circle
2.Rectangle
3.Exit
------------------
Enter your choice: 3
Exiting...
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 14

Date: 09/04/2024

AIM: -

Create a Graphics package that has classes and interfaces for figures Rectangle, Triangle,Square and Circle. Test the package by finding the area of these figures.

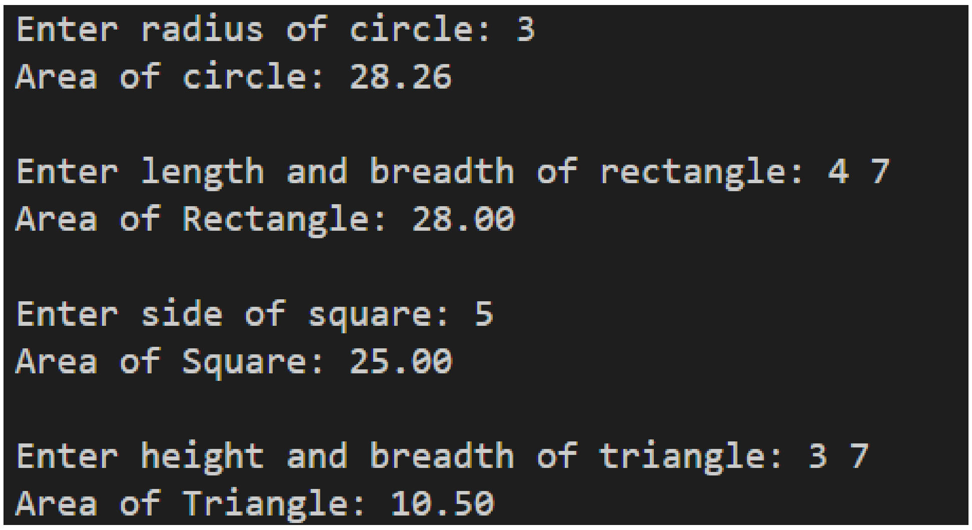
ALGORITHM: -

1. Define the **Figure** interface, and declare an abstract method **area**().
2. Define the **Circle** class:
   - Declare a double variable **radius**.
   - Implement a **constructor** that takes the **radius** as a parameter and assigns it to the **radius** member variable.
   - Implement the **area**() method to calculate and print the area of the circle.
3. Define the **Rectangle** class:
   - Declare double variables **length** and **breadth**.
   - Implement a **constructor** that takes the **length** and **breadth** as parameters and assigns them to the respective member variables.
   - Implement the **area**() method to calculate and print the area of the rectangle.
4. Define the **Square** class:
   - Declare a double variable **length**.
   - Implement a constructor that takes the **length** as a parameter and assigns it to the length member variable.
   - Implement the **area**() method to calculate and print the area of the square.
5. Define the **Triangle** class:
   - Declare double variables **breadth** and **height**.
   - Implement a constructor that takes the **breadth** and **height** as parameters and assigns them to the respective member variables.
   - Implement the **area**() method to calculate and print the area of the triangle.
6. In the main method:
   - Prompt the user to enter the **radius** of a **circle** and read the input.
   - Create a **Circle object** with the given radius.
   - Call the **area**() method of the Circle object to calculate and print the area of the circle.
   - Prompt the user to enter the **length** and **breadth** of a rectangle and read the inputs.
   - Create a **Rectangle object** with the given length and breadth.
   - Call the **area**() method of the Rectangle object to calculate and print the area of the rectangle.
   - Prompt the user to enter the **side of a square** and read the input.
   - Create a Square object with the given side length.
   - Call the **area**() method of the Square object to calculate and print the area of the square.
   - Prompt the user to enter the **height** and **breadth** of a triangle and read the inputs.
   - Create a Triangle object with the given height and breadth.
   - Call the **area**() method of the Triangle object to calculate and print the area of the triangle.

PROGRAM : -

```
package Graphics;
interface Figure{
   public void area();
}
public class Circle implements Figure{
   double radius;
   public Circle(double r){
      radius = r;
   }
   public void area(){
      System.out.println("Area of circle: " + String.format("%.2f" ,(3.14*radius*radius)));
   }
}
public class Rectangle implements Figure{
   double length;
   double breadth;
   public Rectangle(double l,double b){
      length=l;
      breadth=b;
   }
   public void area(){
      System.out.println("Area of Rectangle: " + String.format("%.2f" ,(length*breadth)));
   }
}
public class Square implements Figure{
   double length;
   public Square(double l){
      length=l;
   }
   public void area(){
      System.out.println("Area of Square: " + String.format("%.2f" ,(length*length)));
   }
}
public class Triangle implements Figure{
   double breadth;
   double height;
   public Triangle(double b,double h){
      height=h;
      breadth=b;
   }
   public void area(){
      System.out.println("Area of Triangle: " + String.format("%.2f" ,(0.5*height*breadth)));
   }
}
```

<u>OUTPUT: -</u>



```
Enter radius of circle: 3
Area of circle: 28.26

Enter length and breadth of rectangle: 4 7
Area of Rectangle: 28.00

Enter side of square: 5
Area of Square: 25.00

Enter height and breadth of triangle: 3 7
Area of Triangle: 10.50
```

<u>RESULT: -</u>

The program was executed successfully, and output is obtained.

EXERCISE 15

Date: 09/04/2024

AIM: -

Write a user defined exception class to authenticate the user name and password.

ALGORITHM: -

```
1. Define two custom exceptions: nameexception and passexception, both extending
   RuntimeException, and displaying the message passed with the exception.
2. Define a class user with instance variables name and password.
3. Declare a constructor to initialize the variables.
4. Declare a method login within user class to check if the provided username and password
   match the stored values. If they match, print Login Successful, otherwise throw a
   exception indicating invalid username or password.
5. In the main class:
     • Prompt the user to enter a name and validate it character by character.
         o If the character is not an alphabet, throw a nameexception indicating
           invalid name.
     • Prompt the user to enter a password and validate it.
         o If the password length is less than 8 characters, throw a passexception,
           indicating minimum length of password.
         o If the password does not contain at least one digit, throw a passexception,
           indicating minimum requirement of 1 digit in the password.
6. Create a user object with the provided name and password.
7. Prompt for login username and password.
8. Call the login method of the user object.
```

PROGRAM : -

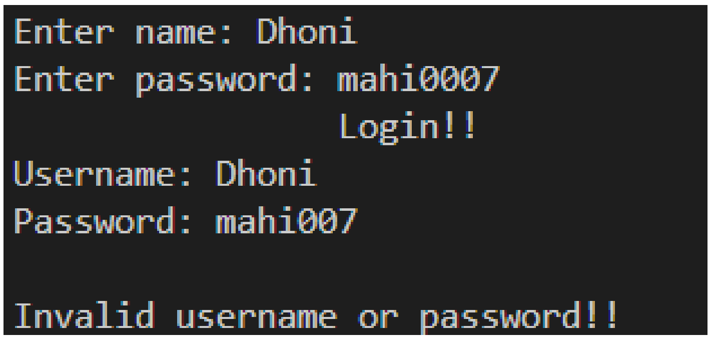
```
import java.util.*;

class nameexception extends RuntimeException{

   nameexception(String s){

      super(s);

   }

}

class passexception extends RuntimeException{

   passexception(String s){

      super(s);

   }

}
```

```
class user{
   String name,password;
   user(String n, String p){
      name=n;
      password=p;
   }
   void login(String n, String p){
      try{
         if(name.equals(n)&&password.equals(p))
            System.out.println("\t\tLogin successful");
         else
            throw new passexception("Invalid username or password!! ");
      }
      catch (passexception e){
         System.out.println("\n"+e.getMessage());
         System.exit(0);
      }
   }
}
class validation{
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter name: ");
      String s=sc.next();
      try{
         for(int i=0;i<s.length();i++){
            char ch=s.charAt(i);
            if((ch>=65&&ch<=90)||(ch>=97&&ch<=122))
               continue;
            else
               throw new nameexception("Invalid Name");
```

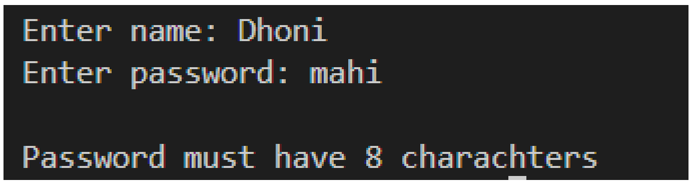
```
        }
    }
    catch (nameexception e){
        System.out.println("\n"+e.getMessage());
        System.exit(0);
    }
    System.out.print("Enter password: ");
    String pass=sc.next();
    try{
        int p=0;
        if(pass.length()<8)
            throw new nameexception("Password must have 8 characters ");
        for(int i=0;i<pass.length();i++){
            char ch=pass.charAt(i);
            if((ch>=48&&ch<=57)){
                p=1;
            }
        }
        if(p==0)
            throw new passexception("Password must contain atleast 1 number");
    }
    catch (nameexception e){
        System.out.println("\n"+e.getMessage());
        System.exit(0);
    }
    catch (passexception e){
        System.out.println("\n"+e.getMessage());
        System.exit(0);
    }
    user u1=new user(s,pass);
    System.out.println("\t\tLogin!!");
```

```
        System.out.print("Username: ");
        String n1=sc.next();
        System.out.print("Password: ");
        String p1=sc.next();
        u1.login(n1,p1);
    }
}
```

OUTPUT: -



```
Enter name: Dhoni
Enter password: mahi0007
                Login!!
Username: Dhoni
Password: mahi007

Invalid username or password!!
```



```
Enter name: Dhoni
Enter password: mahi

Password must have 8 charachters
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 16

Date: 09/04/2024

AIM: -

Find the average of N positive integers, raising a user defined exception for each negative input.

ALGORITHM: -

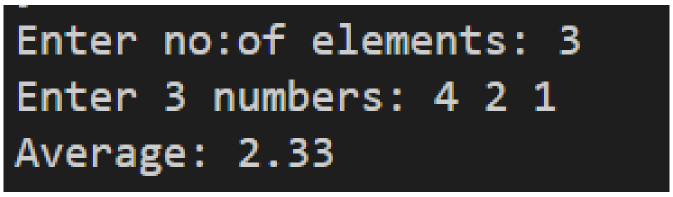
1. Define a custom exception class **negative** which extends **RuntimeException**. This exception is thrown when a negative number is encountered.
2. Define the main class **Average**.
3. Prompt the user to enter the number of elements **n**.
4. Create an integer array **nos** of size **n**.
5. Initialize a variable **sum** to store the sum of the entered numbers.
6. Prompt the user to enter **n** numbers and store them in the array **nos**.
    - While taking input, if any number entered is negative, throw a **negative** exception indicating that a negative number is not allowed.
    - If the entered number is positive, add it to the **sum**.
7. Calculate and display the average by dividing the **sum** by the number of elements (**n**).

PROGRAM : -

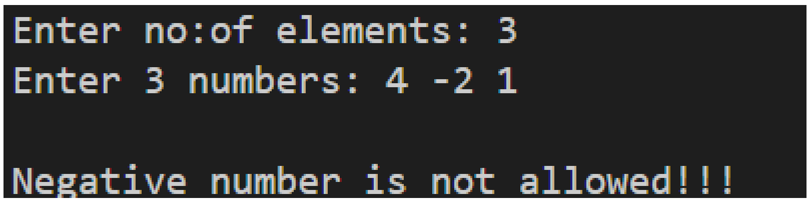
```
import java.util.*;
class negative extends RuntimeException{
  negative(String s){
    super(s);
  }
}
class Average{
  public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter no:of elements: ");
    int n=sc.nextInt();
    int nos[]=new int[n];
    double sum=0;
    System.out.print("Enter "+n+" numbers: ");
    for(int i=0;i<n;i++){
      nos[i]=sc.nextInt();
      try{
        if(nos[i]<0)
```

```
                throw new negative("Negative number is not allowed !!!");
              else
              sum+=nos[i];
           }
           catch(negative e){
              System.out.println("\n"+e.getMessage());
              System.exit(0);
           }
        }
        System.out.println("Average: " + String.format("%.2f", sum/n));
    }
}
```

OUTPUT: -



```
Enter no:of elements: 3
Enter 3 numbers: 4 2 1
Average: 2.33
```



```
Enter no:of elements: 3
Enter 3 numbers: 4 -2 1

Negative number is not allowed!!!
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 17

Date: 16/04/2024

AIM: -

Define 2 classes; one for generating multiplication table of 5 and other for displaying first N prime numbers. Implement using thread class.

ALGORITHM: -

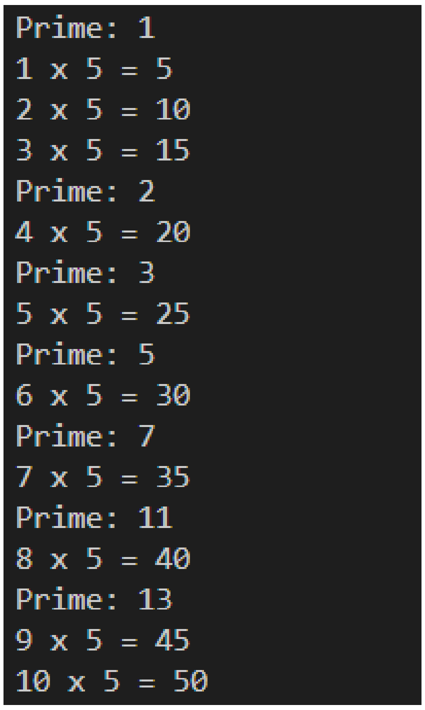
1. Define a class named **multiple5** that **extends** the **Thread** class.
2. Override the **run**() method in the multiple5 class.
   - Iterate from 1 to 10.
   - For each iteration, print the value of **i multiplied by 5.**
3. Define another class named **Prime** that **extends** the **Thread** class.
4. Declare an integer variable **n** in the Prime class.
5. Define a **parameterized constructor** in the **Prime** class that takes an integer parameter limit and assigns it to the **n** variable.
6. Define a method named **isprime(int n)** in the **Prime** class that checks whether a given number n is prime or not.
   - Iterate from 2 to n/2.
   - If n is divisible by any number in this range, **return 0** (indicating not prime).
   - Otherwise, **return 1** (indicating prime).
7. Override the **run**() method in the Prime class.
   - Inside the **run**() method, iterate from 1 to n.
   - For each iteration, check if the number is prime using the **isprime**() method.
   - If it's prime, print the prime number.
8. In the **main**() method:
   - Create an instance **m** of the **multiple5** class.
   - Prompt the user to enter a **limit** for generating prime numbers.
   - Create an instance **m1** of the **Prime** class with the input limit n.
   - **Start** both threads **m** and **m1**.

PROGRAM : -

```
import java.util.*;
class multiple5 extends Thread{
   public void run(){
      for(int i=1;i<=10;i++){
         System.out.println(i+" x 5 = "+(i*5));
      }
   }
}
class Prime extends Thread{
   int n;
```

```
    Prime(int limit){
        n=limit;
    }
    int isprime(int n){
        for(int i=2;i<=n/2;i++){
            if(n%i==0)
            return 0;
        }
        return 1;
    }
    public void run(){
        for(int i=1;i<=n;i++){
            if(isprime(i)==1)
                System.out.println("Prime: "+i);
        }
    }
}
public class Threadclass{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        multiple5 m =new multiple5();
        System.out.print("Enter limit for generating prime: ");
        int n=sc.nextInt();
        Prime m1 =new Prime(n);
        m.start();
        m1.start();
    }
}
```

OUTPUT: -



```
Prime: 1
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
Prime: 2
4 x 5 = 20
Prime: 3
5 x 5 = 25
Prime: 5
6 x 5 = 30
Prime: 7
7 x 5 = 35
Prime: 11
8 x 5 = 40
Prime: 13
9 x 5 = 45
10 x 5 = 50
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 18

Date: 16/04/2024

AIM: -

Define 2 classes; one for generating Fibonacci numbers and other for displaying even numbers in a given range. Implement using threads. (Runnable Interface).

ALGORITHM: -

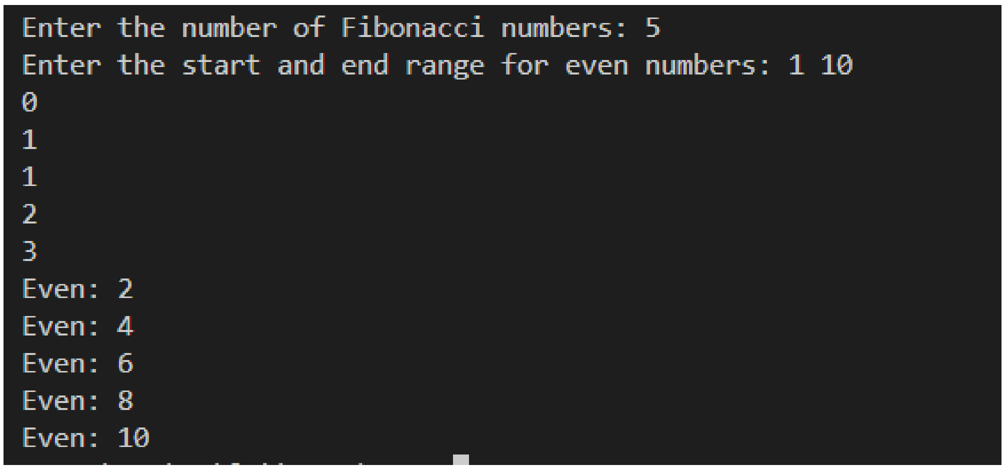
1. Define a class named **fibonacci** that **implements** the **Runnable interface**.
2. Declare an integer variable **n** to store the limit.
3. Define a **parameterized constructor** in the **fibonacci** class that takes an integer parameter limit and assigns it to the **n** variable.
4. Implement the **run**() method in the fibonacci class.
   - Initialize variables **a** and **b** to **0** and **1** respectively.
   - Iterate from 1 to n.
   - **Print** the value of **a**.
   - Calculate the next Fibonacci number c by adding a and b.
   - Update a and b for the next iteration.
5. Define another class named **Even** that **implements** the **Runnable interface**.
6. Declare integer variables **start** and **end** in the Even class.
7. Define a **parameterized constructor** in the Even class that takes two integers **a** and **b** and assigns them to **start** and **end** respectively.
8. Implement the **run**() method in the Even class.
   - Iterate from start to end.
   - Check if the current number is even (i % 2 == 0).
   - If it's even, print the even number.
9. In the main():
10. Prompt the user to enter the number of Fibonacci numbers (n).
11. Prompt the user to enter the start and end range for even numbers.
12. Create two **threads f** and **e** with **instances** of **fibonacci** and **Even** classes respectively, passing necessary parameters.
13. **Start** both threads **f** and **e**.

PROGRAM : -

```
import java.util.*;
class fibonacci implements Runnable{
  int n;
  fibonacci(int limit){
    n=limit;
  }
  public void run(){
        int a = 0, b = 1, c;
        for (int i = 1; i <= n; i++){
            System.out.println(a);
            c = a + b;
            a = b;
            b = c;
        }
    }
}
class Even implements Runnable{
    int start,end;
    Even(int a,int b){
        start=a;
        end=b;
    }
    public void run(){
        for (int i = start; i <= end; i++){
            if (i % 2 == 0){
                System.out.println("Even: "+i);
            }
        }
    }
}
public class Runnable_interface
{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the number of Fibonacci numbers: ");
        int n = sc.nextInt();
        System.out.print("Enter the start and end range for even numbers: ");
        int start = sc.nextInt();
        int end = sc.nextInt();
        Thread f=new Thread(new fibonacci(n));
        Thread e=new Thread(new Even(start,end));
        f.start();
```

```
      e.start();
   }
}
```

OUTPUT: -



```
Enter the number of Fibonacci numbers: 5
Enter the start and end range for even numbers: 1 10
0
1
1
2
3
Even: 2
Even: 4
Even: 6
Even: 8
Even: 10
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 19

Date: 16/04/2024

AIM: -

Program to create a generic stack and do the Push and Pop operations.

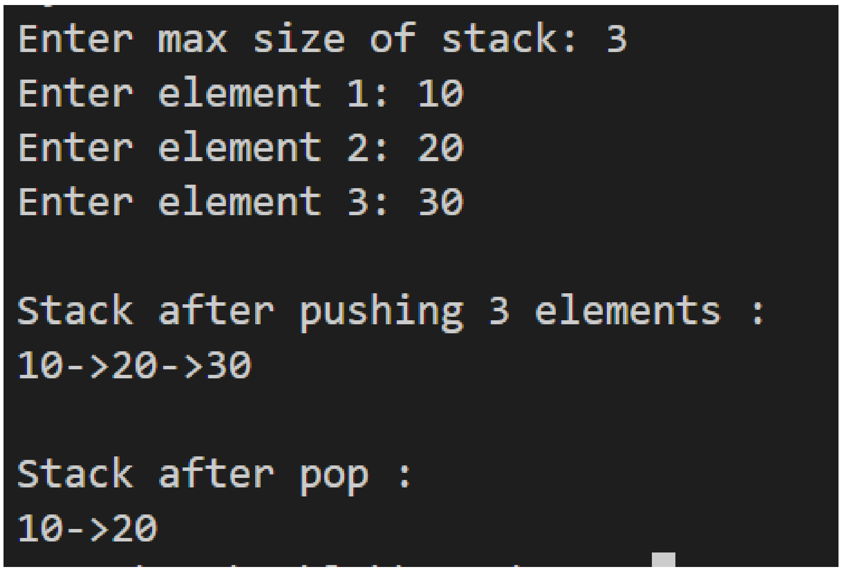
ALGORITHM: -

1. Define a generic class called stack with the following attributes:
   - **A**: An **ArrayList** to store elements of generic type T.
   - **top**: An integer representing the **index** of the **top** element in the stack. Initialized to -1.
   - **size**: An integer representing the **maximum size** of the stack.
2. Define a constructor that initializes the size of the stack and creates an ArrayList of the specified size.
3. Implement the **push** method:
   - Check if the stack is **full** (top + 1 == size).
   - If not, **increment** top **by** 1 and add the element to the ArrayList at the top index.
4. Implement the **top** method:
   - Check if the stack is **empty** (top == -1).
   - If not, **return** the **element at the top** index of the ArrayList.
5. Implement the **pop** method:
   - Check if the stack is empty (top == -1).
   - If not, **decrement top by 1.**
6. Implement the empty method:
   - Return true if the stack is empty (top == -1), otherwise return false.
7. Implement the **toString** method to represent the stack as a string:
   - **Initialize** an **empty string.**
   - Iterate over the elements in the ArrayList from index 0 to top - 1.
   - Append each element to the string followed by '->'.
   - Append the last element (at index top) to the string.
   - Return the constructed string.
8. Define the **main** method:
   - Prompt the user to enter the **maximum size** of the stack and read the input.
   - Create an **instance** of the **stack** class with the specified maximum size.
   - Use a loop to push elements into the stack:
   - Print the stack after pushing all elements.
   - Pop an element from the stack.
   - Print the stack after popping.

PROGRAM : -

```
import java.util.*;
class stack<T>{
    ArrayList<T> A;
    int top = -1;
    int size;
    stack(int size){
        this.size = size;
        this.A = new ArrayList<T>(size);
    }
    void push(T X){
        if (top + 1 == size) {
            System.out.println("Stack Overflow");
        }
        else {
            top = top + 1;
            if (A.size() > top)
                A.set(top, X);
            else
                A.add(X);
        }
    }
    T top(){
        if (top == -1) {
            System.out.println("Stack Underflow");
            return null;
        }
        else
            return A.get(top);
    }
    void pop(){
        if (top == -1){
            System.out.println("Stack Underflow");
        }
        else
            top--;
    }
    boolean empty() { return top == -1; }
    public String toString(){
        String Ans = "";
        for (int i = 0; i < top; i++){
            Ans += String.valueOf(A.get(i)) + "->";
        }
        Ans += String.valueOf(A.get(top));
        return Ans;
    }
}
public class GenericStack {
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter max size of stack: ");
        int n=sc.nextInt();
        stack<Integer> s1 = new stack<>(n);
        int v;
        for(int i=0; i<n; i++){
            System.out.print("Enter element "+(i+1)+": ");
            v=sc.nextInt();
            s1.push(v);
        }
        System.out.println("\nStack after pushing "+n+" elements :\n" + s1);
        s1.pop();
        System.out.println("\nStack after pop :\n" + s1);
    }
}
```

OUTPUT: -



```
Enter max size of stack: 3
Enter element 1: 10
Enter element 2: 20
Enter element 3: 30

Stack after pushing 3 elements :
10->20->30

Stack after pop :
10->20
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 20

Date: 16/04/2024

AIM: -

Program to maintain a list of Strings using ArrayList from collection framework, perform built-in operations.

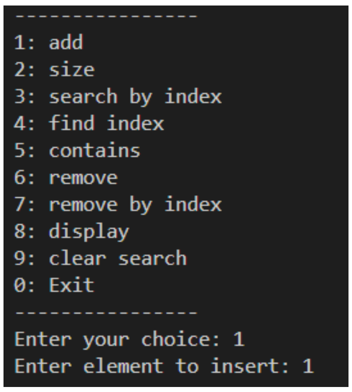
ALGORITHM: -

1. Create an empty **Arrayl.ist** of Strings called **list**.
2. Create variables **el** (to store user input for elements) and **ch** (to store user input for choices).
3. Start a do-while loop with the condition **ch** not equal to 0.
4. Inside the loop, display a menu of options and prompt the user to enter their choice.
5. On the basis user's choice:
    - Case 1: Prompt the user to enter an clement and add it to the **ArrayList** using **list_add(e1).**
    - Case 2: Print the size of the **Arraylist** using **list.size()**.
    - Case 3: Prompt the user to enter an **index** and print the element at that index using **list. get(index)**.
    - Case 4: Prompt the user to enter an **element** and **find its index** using **list.indexOf(el)**.
    - Case 5: Prompt the user to enter an **clement** and check **if it exists** in the ArrayList using **list.contains(el).**
    - Case 6. Prompt the user to enter an **element** and **remove** it from the ArrayList using **list.remove(el)**.
    - Case 7: Prompt the user to enter an **index** and **remove** the element at that index using **list.remove(index).**
    - Case 8: **Print** the **entire ArrayList** using **'list'**.
    - Case 9: **Clear** the **ArrayList** using **list. clear( )**
    - Case 0: Exit the loop.

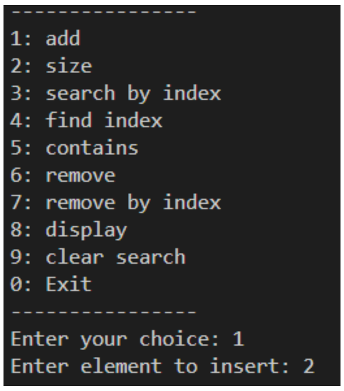
PROGRAM : -

```
import java.util.*;
public class Arraylist {
  public static void main(String[] args) {
    ArrayList <String> list = new ArrayList<String>();
    Scanner sc=new Scanner(System.in);
    String el;
    int ch;
    do{
        System.out.print("\n----------------\n1: add\n2: size\n3: search by index\n4: find
index\n5: contains\n6: remove\n7: remove by index\n8: display\n9: clear search\n0: Exit\n-
---------------\nEnter your choice: ");
        ch=sc.nextInt();
        switch(ch){
          case 1:
            System.out.print("Enter element to insert: ");
            el=sc.next();
            list.add(el);
            break;
          case 2:
            System.out.println("Size of the arraylist: "+list.size());
            break;
          case 3:
            System.out.print("Enter index of element to search: ");
            int index=sc.nextInt();
            System.out.println("Element at index "+index+" is "+list.get(index));
            break;
          case 4:
            System.out.print("Enter an element to find index: ");
            el=sc.next();
            index=list.indexOf(el);
            System.out.println("Index of "+el+" is "+index);
            break;
          case 5:
            System.out.print("Enter a element");
            el=sc.next();
            boolean contains=list.contains(el);
            System.out.println(el+" is in the list :"+contains);
            break;
          case 6:
                System.out.print("Enter the element to be removed: ");
                el=sc.next();
                boolean removed=list.remove(el);
                System.out.println("After removing "+el+ " ArrayList : "+list);
                break;
            case 7:
                System.out.print("Enter the index to remove the element: ");
                index=sc.nextInt();
                list.remove(index);
                System.out.println("After removing the element, arraylist:"+list);
                break;
            case 8:
                System.out.println("Arraylist: "+list);
                break;
            case 9:
                list.clear();
                break;
            case 0:
                break;
          }
      }while(ch!=0);
   }
}
```

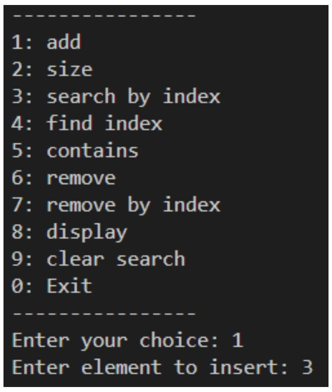
OUTPUT: -



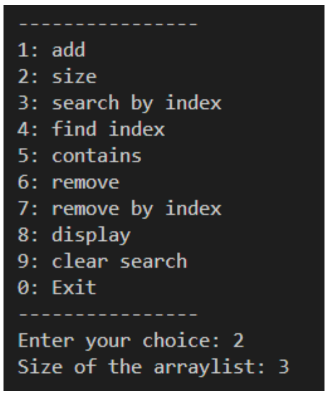
```
----------------
1: add
2: size
3: search by index
4: find index
5: contains
6: remove
7: remove by index
8: display
9: clear search
0: Exit
----------------
Enter your choice: 1
Enter element to insert: 1
```



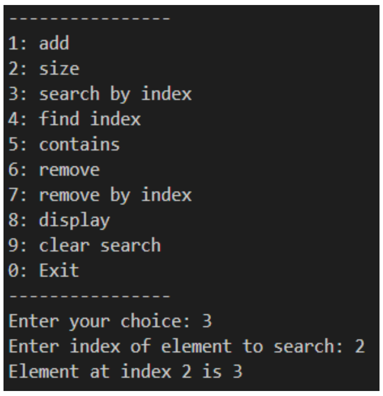
```
----------------
1: add
2: size
3: search by index
4: find index
5: contains
6: remove
7: remove by index
8: display
9: clear search
0: Exit
----------------
Enter your choice: 1
Enter element to insert: 2
```



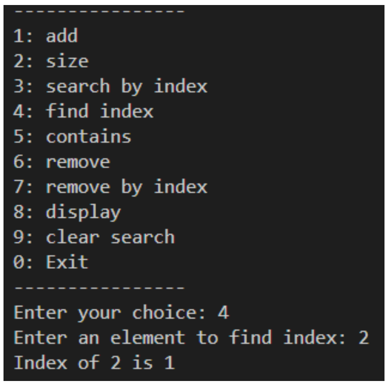
```
----------------
1: add
2: size
3: search by index
4: find index
5: contains
6: remove
7: remove by index
8: display
9: clear search
0: Exit
----------------
Enter your choice: 1
Enter element to insert: 3
```



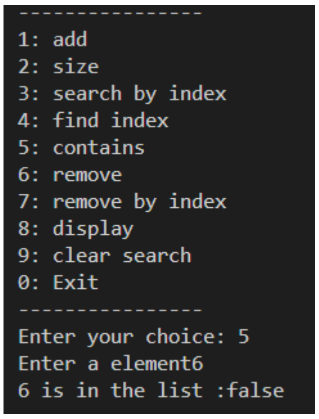
```
----------------
1: add
2: size
3: search by index
4: find index
5: contains
6: remove
7: remove by index
8: display
9: clear search
0: Exit
----------------
Enter your choice: 2
Size of the arraylist: 3
```



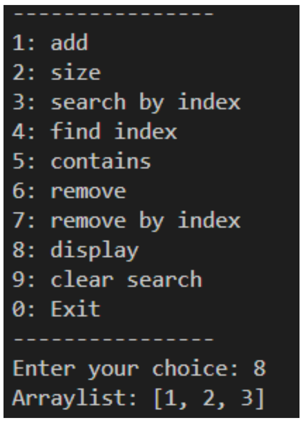
```
----------------
1: add
2: size
3: search by index
4: find index
5: contains
6: remove
7: remove by index
8: display
9: clear search
0: Exit
----------------
Enter your choice: 3
Enter index of element to search: 2
Element at index 2 is 3
```



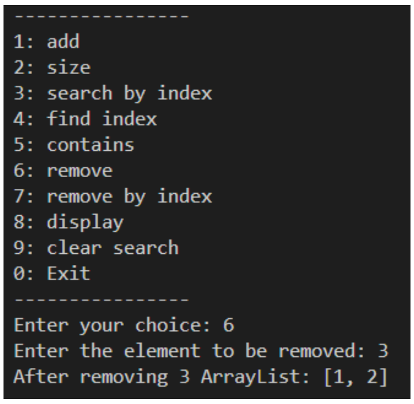
```
----------------
1: add
2: size
3: search by index
4: find index
5: contains
6: remove
7: remove by index
8: display
9: clear search
0: Exit
----------------
Enter your choice: 4
Enter an element to find index: 2
Index of 2 is 1
```



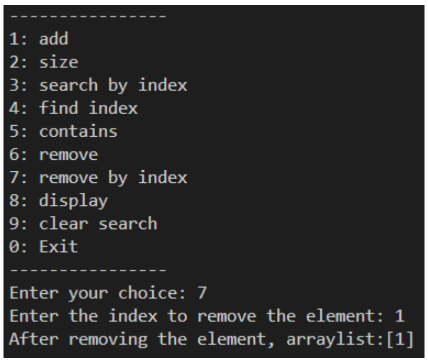
```
----------------
1: add
2: size
3: search by index
4: find index
5: contains
6: remove
7: remove by index
8: display
9: clear search
0: Exit
----------------
Enter your choice: 5
Enter a element6
6 is in the list :false
```



```
----------------
1: add
2: size
3: search by index
4: find index
5: contains
6: remove
7: remove by index
8: display
9: clear search
0: Exit
----------------
Enter your choice: 8
Arraylist: [1, 2, 3]
```



```
----------------
1: add
2: size
3: search by index
4: find index
5: contains
6: remove
7: remove by index
8: display
9: clear search
0: Exit
----------------
Enter your choice: 6
Enter the element to be removed: 3
After removing 3 ArrayList: [1, 2]
```



```
----------------
1: add
2: size
3: search by index
4: find index
5: contains
6: remove
7: remove by index
8: display
9: clear search
0: Exit
----------------
Enter your choice: 7
Enter the index to remove the element: 1
After removing the element, arraylist:[1]
```

RESULT: -

The program was executed successfully, and output is obtained

EXERCISE 21

Date: 16/04/2024

AIM: -

Program to demonstrate the creation of queue object using the PriorityQueue class.

ALGORITHM: -

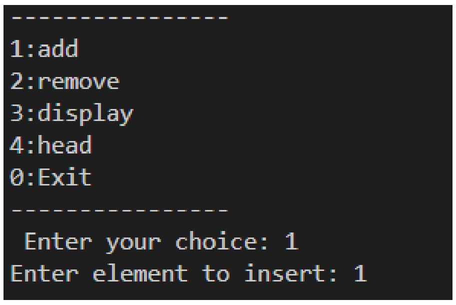
1. Create a new **PriorityQueue** object called q to store strings.
2. Declare variables el and ch of type String and int respectively.
3. Based on the users choice
   - Case I: **Add** :
     - Prompt the user to enter the clement to insert and store it in the variable cl.
     - Call the add method on the q object, passing el as the argument.
   - Case 2: **Remove**
     - Prompt the user to enter the clement to remove and store it in the variable el.
     - Call the remove method on the q object, passing el as the argument .
   - Case 3: **Display**
     - Print the contents of the q priority queue.
   - Case 4: **Head**
     - Print the head of the q priority queue using the peck method.
   - Case 5: **Poll**
     - Remove and return the head of the q priority queue using the poll method.
   - Case 0: **Wrong choice**

PROGRAM : -

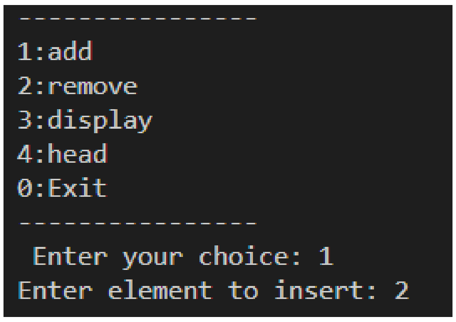
```
import java.util.*;
public class Queue{
  public static void main(String[] args){
    PriorityQueue<String>q=new PriorityQueue<String>();
    Scanner sc=new Scanner(System.in);
    String el;
    int ch;
    do{
      System.out.print("\n----------------\n1:add\n2:remove \n3:display\n4:head\0:wrong
choice\n----------------\n Enter your choice: ");
      ch=sc.nextInt();
      switch(ch){
        case 1:
          System.out.print("Enter element to insert:\n");
```

```
            el=sc.next();
            q.add(el);
            break;
          case 2:
            q. remove();
            break;
          case 3:
            System.out.println("Priority queue:"+q);
            break;
          case 4:
            System.out.println("Head of the queue:"+q.peek());
            break;
          case 9:break;
        }
      }while(ch!=0);
    }
}
```

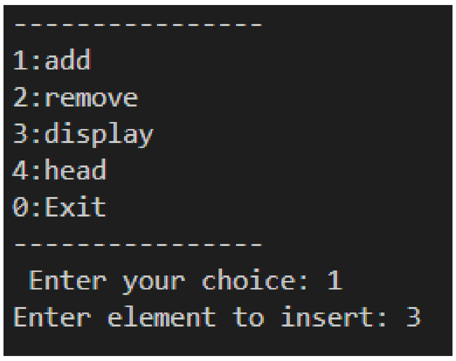
OUTPUT: -



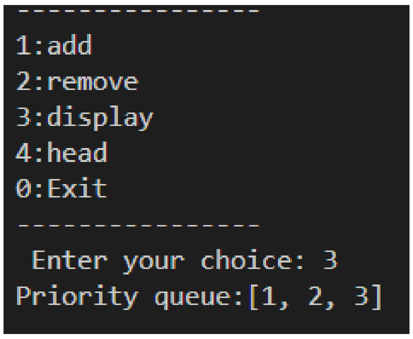
```
----------------
1:add
2:remove
3:display
4:head
0:Exit
----------------
 Enter your choice: 1
Enter element to insert: 1
```



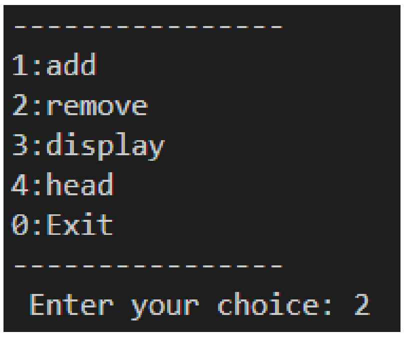
```
----------------
1:add
2:remove
3:display
4:head
0:Exit
----------------
 Enter your choice: 1
Enter element to insert: 2
```



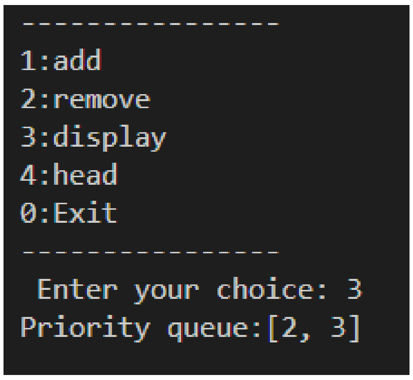
```
----------------
1:add
2:remove
3:display
4:head
0:Exit
----------------
 Enter your choice: 1
Enter element to insert: 3
```



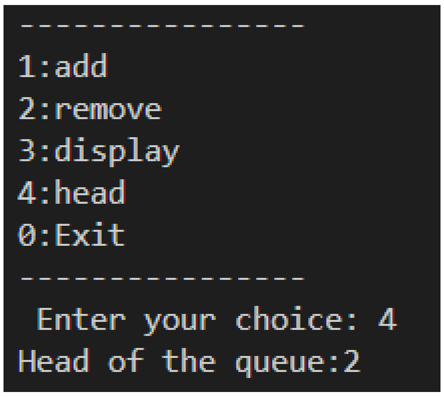
```
----------------
1:add
2:remove
3:display
4:head
0:Exit
----------------
 Enter your choice: 3
Priority queue:[1, 2, 3]
```



```
----------------
1:add
2:remove
3:display
4:head
0:Exit
----------------
 Enter your choice: 2
```



```
----------------
1:add
2:remove
3:display
4:head
0:Exit
----------------
 Enter your choice: 3
Priority queue:[2, 3]
```



```
----------------
1:add
2:remove
3:display
4:head
0:Exit
----------------
 Enter your choice: 4
Head of the queue:2
```

<u>RESULT: -</u>

The program was executed successfully, and output is obtained.

EXERCISE 22

Date: 16/04/2024

AIM: -

Program to demonstrate the creation of Set object using the LinkedHashset class.

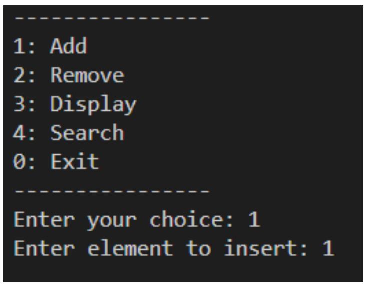
ALGORITHM: -

1. Create a new **LinkedHashSet** object called set to store strings.
2. Declare variables **el** and **ch** of type String and int respectively.
3. Based on the users choice:
   - Case I: **Add**
     - Prompt the user to enter the element to insert and store it in the variable el.
     - Call the add method on the set object, passing el as the argument.
   - Case 2: **Remove**
     - Prompt the user to enter the element to remove and store it in the variable el.
     - Call the remove method on the set object, passing el as the argument.
   - Case 3: **Display**
     - Print the contents of the set linked hash set.
   - Case 4: **Search**
     - Prompt the user to enter the element to search and store it in the variable el.
     - Call the contains method on the set object, passing el as the argument, and store the result in a boolean variable contains.
     - Print whether the set contains the element or not.
   - Case 0: **Exit**
     - Break out of the switch statement.

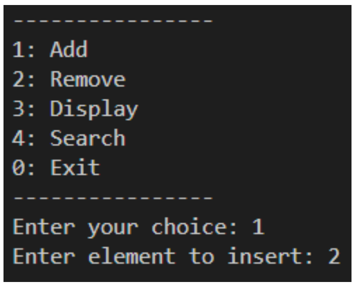
PROGRAM : -

```
import java.util.*;
public class Hashset {
  public static void main(String[] args) {
    Set <String> set =new LinkedHashSet<String>();
    Scanner sc=new Scanner(System.in);
    String el;
    int ch;
    do{
      System.out.print("\n----------------\n1: Add\n2: Remove\n3: Display\n4: Search\n0:
Exit\n----------------\nEnter your choice: ");
      ch=sc.nextInt();
      switch(ch){
        case 1:
                System.out.print("Enter element to insert: ");
                el=sc.next();
                set.add(el);
                break;
            case 2:
                System.out.print("Enter element to remove: ");
                el=sc.next();
                set.remove(el);
                break;
            case 3:
                System.out.println("Linked Hashset: "+set);
                break;
            case 4:
                System.out.print("Enter element to search: ");
                el=sc.next();
                boolean contains=set.contains(el);
                System.out.println("Set contains "+el+" : "+contains);
                break;
            case 0:
                System.out.println("Exiting");
                break;
          }
       }while(ch!=0);
    }
}
```

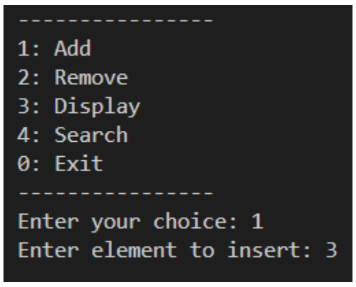
OUTPUT: -



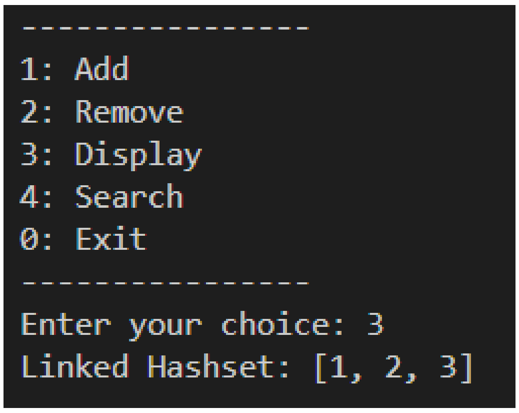
```
----------------
1: Add
2: Remove
3: Display
4: Search
0: Exit
----------------
Enter your choice: 1
Enter element to insert: 1
```



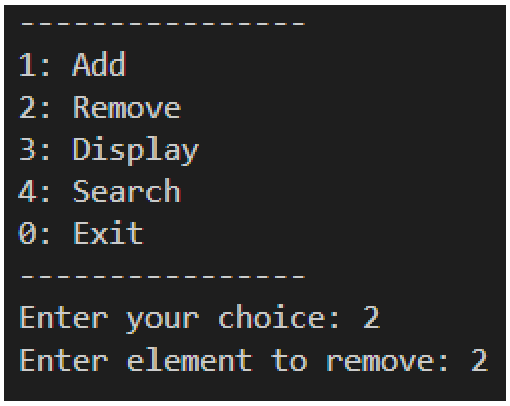
```
----------------
1: Add
2: Remove
3: Display
4: Search
0: Exit
----------------
Enter your choice: 1
Enter element to insert: 2
```



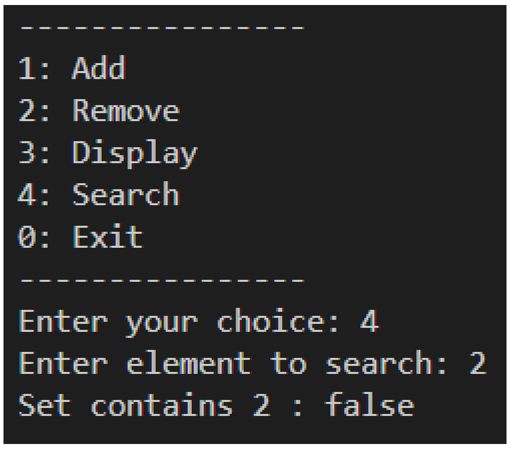
```
----------------
1: Add
2: Remove
3: Display
4: Search
0: Exit
----------------
Enter your choice: 1
Enter element to insert: 3
```



```
----------------
1: Add
2: Remove
3: Display
4: Search
0: Exit
----------------
Enter your choice: 3
Linked Hashset: [1, 2, 3]
```



```
----------------
1: Add
2: Remove
3: Display
4: Search
0: Exit
----------------
Enter your choice: 2
Enter element to remove: 2
```



```
----------------
1: Add
2: Remove
3: Display
4: Search
0: Exit
----------------
Enter your choice: 4
Enter element to search: 2
Set contains 2 : false
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 23

Date: 16/04/2024

AIM: -

Implement a simple calculator using AWT components

ALGORITHM: -

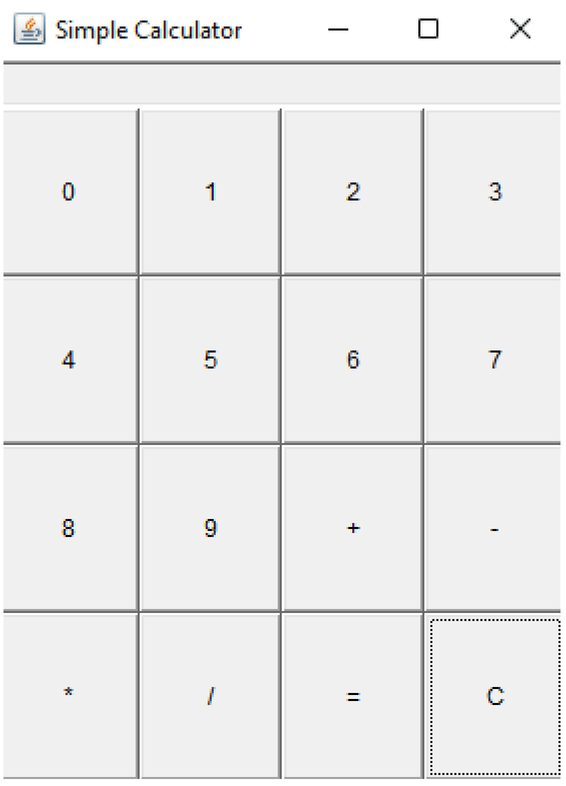
1. Import necessary packages: **java.awt.*** and **java.awt.event.***.
2. Declare the **SimpleCalculator** class, which **extends Frame** and **implements ActionListener**.
3. Define class variables:
   - **textField**: TextField for displaying input and result.
   - **numberButtons**: Array of Buttons for numbers 0-9.
   - **operationButtons**: Array of Buttons for operations (+, -, *, /).
   - **equalsButton**: Button for performing the calculation.
   - **clearButton**: Button for clearing the text field.
   - **num1**, **num2**: Doubles to store operands.
   - **result**: Double to store the result of the operation.
   - **operation**: Character to store the current operation.
4. Implement the **constructor** SimpleCalculator():
   - Set the **title** and **size** of the **frame**.
   - Set layout to BorderLayout.
   - Create and configure the textField, add it to the NORTH of the frame.
   - Create **buttonPanel** with **GridLayout(4, 4).**
   - Create **numberButtons** for digits 0-9, add ActionListener, and add them to **buttonPanel**.
   - Create **operationButtons** for operations (+, -, *, /), add ActionListener, and add them to **buttonPanel**.
   - Create **equalsButton** with ActionListener and add it to **buttonPanel**.
   - Create **clearButton** with ActionListener and add it to **buttonPanel**.
   - Add **buttonPanel** to the CENTER of the frame.
   - Add window closing listener to exit the application.
5. Implement **actionPerformed(ActionEvent ae)** method:
6. Get the action command from the event.
7. If the command is a digit (0-9), append it to the text field.
8. If the command is 'C', clear the text field.
9. If the command is '=', perform the calculation based on the stored operation and display the result in the text field.
10. If the command is an operation (+, -, *, /), store the current value in num1, store the operation, and clear the text field.
11. In the main method:
    - Create an instance of SimpleCalculator.
    - Make the calculator visible.

PROGRAM : -

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

public class SimpleCalculator extends Frame implements ActionListener{
   private TextField textField;
   private Button[] numberButtons;
   private Button[] operationButtons;
   private Button equalsButton;
   private Button clearButton;
   private double num1, num2, result;
   private char operation;
   public SimpleCalculator(){
      setTitle("Simple Calculator");
      setSize(300, 400);
      setLayout(new BorderLayout());

      textField = new TextField();
      textField.setEditable(false);
      add(textField, BorderLayout.NORTH);

      Panel buttonPanel = new Panel();
      buttonPanel.setLayout(new GridLayout(4, 4));

      numberButtons = new Button[10];
      for (int i = 0; i < 10; i++){
         numberButtons[i] = new Button(String.valueOf(i));
         numberButtons[i].addActionListener(this);
         buttonPanel.add(numberButtons[i]);
      }
```

```
        operationButtons = new Button[4];
        operationButtons[0] = new Button("+");
        operationButtons[1] = new Button("-");
        operationButtons[2] = new Button("*");
        operationButtons[3] = new Button("/");
        for (int i = 0; i < 4; i++){
            operationButtons[i].addActionListener(this);
            buttonPanel.add(operationButtons[i]);
        }
        equalsButton = new Button("=");
        equalsButton.addActionListener(this);
        buttonPanel.add(equalsButton);

        clearButton = new Button("C");
        clearButton.addActionListener(this);
        buttonPanel.add(clearButton);

        add(buttonPanel, BorderLayout.CENTER);

        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent windowEvent){
                System.exit(0);
            }
        });
    }
    public void actionPerformed(ActionEvent ae){
        String command = ae.getActionCommand();
        if (command.charAt(0) >= '0' && command.charAt(0) <= '9')
            textField.setText(textField.getText() + command);
        else if (command.charAt(0) == 'C')
            textField.setText("");
        else if (command.charAt(0) == '='){
          num2 = Double.parseDouble(textField.getText());
          switch (operation){
            case '+':
              result = num1 + num2;
              break;
            case '-':
              result = num1 - num2;
              break;
            case '*':
              result = num1 * num2;
              break;
            case '/':
              if (num2 != 0)
                result = num1 / num2;
              else
                textField.setText("Cannot divide by zero");
              break;
          }
          textField.setText(String.valueOf(result));
        } else {
          num1 = Double.parseDouble(textField.getText());
          operation = command.charAt(0);
          textField.setText("");
        }
    }
    public static void main(String[] args){
        SimpleCalculator calculator = new SimpleCalculator();
        calculator.setVisible(true);
    }
}
```

<u>OUTPUT: -</u>



<u>RESULT: -</u>

The program was executed successfully, and output is obtained.

EXERCISE 24

Date: 16/04/2024

AIM: -

Develop a program to handle all mouse events and window events.

ALGORITHM: -

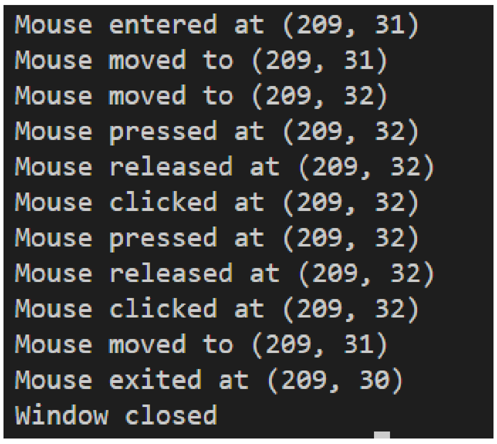
1. Import the required libraries**: javax.swing.*** and **java.awt.event.*.**
2. Define a class named **MouseEventDemo** that extends **JFrame**.
3. Define the constructor:
   - Set the title of the window to "Mouse Events Demo".
   - Set the size of the window to 400x300 pixels.
   - Set the default close operation to exit on close.
4. Add a MouseListener to the frame using **addMouseListener(new MouseAdapter() { ... })**.
5. Override methods for the following mouse events:
   - **mouseClicked(MouseEvent e):** Print the coordinates of the mouse when clicked.
   - **mousePressed(MouseEvent e)**: Print the coordinates of the mouse when pressed.
   - **mouseReleased(MouseEvent e)**: Print the coordinates of the mouse when released.
   - **mouseEntered(MouseEvent e)**: Print the coordinates of the mouse when entered into the frame.
   - **mouseExited(MouseEvent e)**: Print the coordinates of the mouse when exited from the frame.
6. Add a **WindowListener** to the frame using **addWindowListener(new WindowAdapter() { ... })**.
7. Override **the windowClosing(WindowEvent e)** method to print a message when the window is closed.
8. In the **main** method:
   - Use SwingUtilities.invokeLater() to ensure that Swing components are created and accessed in the Event Dispatch Thread for thread safety.
   - Create an instance of MouseEventDemo.
   - Set the visibility of the frame to true.

PROGRAM : -

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

public class MouseEventDemo extends JFrame {
  public MouseEventDemo(){
    setTitle("Mouse Events Demo");
    setSize(400, 300);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        addMouseListener(new MouseAdapter(){
            public void mouseClicked(MouseEvent e) {
                System.out.println("Mouse clicked at (" + e.getX() + ", " + e.getY() + ")");
            }
            public void mousePressed(MouseEvent e){
                System.out.println("Mouse pressed at (" + e.getX() + ", " + e.getY() + ")");
            }
            public void mouseReleased(MouseEvent e) {
                System.out.println("Mouse released at (" + e.getX() + ", " + e.getY() + ")");
            }
            public void mouseEntered(MouseEvent e) {
                System.out.println("Mouse entered at (" + e.getX() + ", " + e.getY() + ")");
            }
            public void mouseExited(MouseEvent e) {
                System.out.println("Mouse exited at (" + e.getX() + ", " + e.getY() + ")");
            }
        });
        addMouseMotionListener(new MouseMotionAdapter()
        {
            public void mouseDragged(MouseEvent e) {
                System.out.println("Mouse dragged to (" + e.getX() + ", " + e.getY() + ")");
            }
            public void mouseMoved(MouseEvent e) {
                System.out.println("Mouse moved to (" + e.getX() + ", " + e.getY() + ")");
            }
        });

        addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e) {
                System.out.println("Window closed");
            }
        });
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() ->{
            MouseEventDemo demo = new MouseEventDemo();
            demo.setVisible(true);
        });
    }
}
```

OUTPUT: -



```
Mouse entered at (209, 31)
Mouse moved to (209, 31)
Mouse moved to (209, 32)
Mouse pressed at (209, 32)
Mouse released at (209, 32)
Mouse clicked at (209, 32)
Mouse pressed at (209, 32)
Mouse released at (209, 32)
Mouse clicked at (209, 32)
Mouse moved to (209, 31)
Mouse exited at (209, 30)
Window closed
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 25

Date: 19/04/2024

AIM: -

Program to list the sub directory and files in a given directory and also search for a file name.

ALGORITHM: -

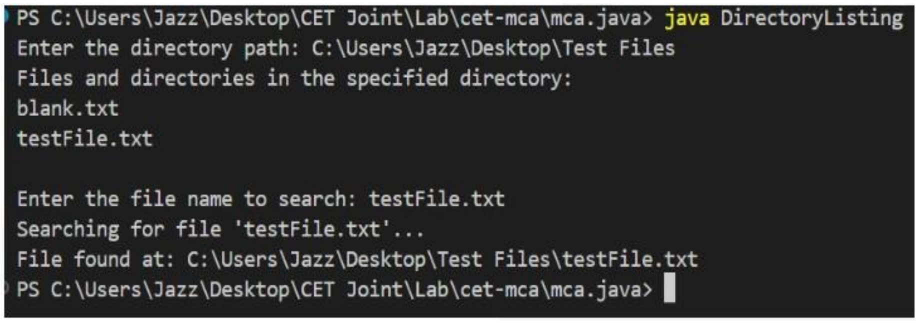
```
1. Define class DirectoryListing.
2. Define static method listFilesAndDirectories(directory: File):
    • For each file in directory:
        o Print file name.
    • If file is a directory, call listFilesAndDirectories recursively.
3. Define static method searchFile(directory: File, fileName: String):
    • For each file in directory:
        o If file matches fileName, print its absolute path.
        o If file is a directory, call searchFile recursively.
    • If file is not found, print a message indicating so.
4. Define main method:
    • Read directory path from user input.
    • Create File object directory with directory path.
    • If directory exists and is a directory:
        o Print files and directories in specified directory.
        o Prompt user to enter file name to search.
        o Search for file in directory.
    • If directory does not exist or is not a directory, print message indicating invalid
      directory path.
```

PROGRAM : -

```java
import java.io.File;
import java.util.Scanner;


public class DirectoryListing {
  static void listFilesAndDirectories(File directory) {
    File[] fileList = directory.listFiles();
    if (fileList != null) {
      for (File file : fileList) {
```

```
                System.out.println(file.getName());
                if (file.isDirectory()) {
                    listFilesAndDirectories(file);
                }
            }
        }
    }
    static void searchFile(File directory, String fileName) {
        File[] fileList = directory.listFiles();
        if (fileList != null) {
            for (File file : fileList) {
                if (file.isFile() && file.getName().equals(fileName)) {
                    System.out.println("File found at: " + file.getAbsolutePath());
                    return;
                } else if (file.isDirectory()) {
                    searchFile(file, fileName);
                }
            }
        }
        System.out.println("File '" + fileName + "' not found.");
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the directory path: ");
        String directoryPath = scanner.nextLine();
        File directory = new File(directoryPath);
        if (directory.exists() && directory.isDirectory()) {
            System.out.println("Files and directories in the specified directory:");
            listFilesAndDirectories(directory);

            System.out.print("\nEnter the file name to search: ");
            String fileName = scanner.nextLine();

            System.out.println("Searching for file '" + fileName + "'...");
            searchFile(directory, fileName);
        } else {
            System.out.println("Invalid directory path.");
        }
        scanner.close();
    }
}
```

<u>OUTPUT: -</u>



```
PS C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java> java DirectoryListing
Enter the directory path: C:\Users\Jazz\Desktop\Test Files
Files and directories in the specified directory:
blank.txt
testFile.txt

Enter the file name to search: testFile.txt
Searching for file 'testFile.txt'...
File found at: C:\Users\Jazz\Desktop\Test Files\testFile.txt
PS C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java>
```

<u>RESULT: -</u>

The program was executed successfully, and output is obtained.

EXERCISE 26

Date: 19/04/2024

AIM: -

Write a program to write to a file, then read from the file and display the contents on the console.

ALGORITHM: -

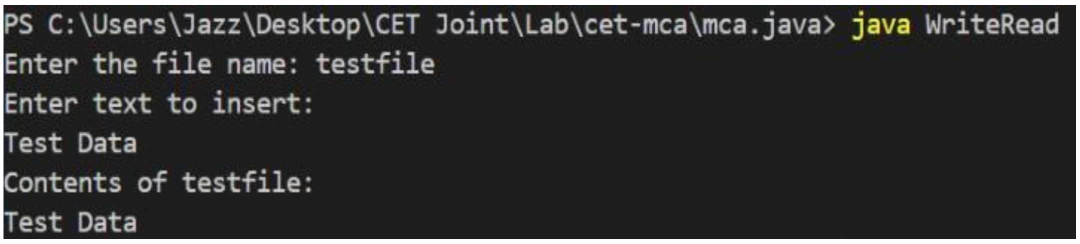
1. Create a Scanner object (sc) to read user input.
2. Prompt the user to enter the file name and store it in fileName.
3. Create a FileOutputStream (fos) with fileName.
4. Prompt the user to enter text to insert and store it in text.
5. Write text to fos.
6. Close fos.
7. Create a FileInputStream (fis) with fileName.
8. Read bytes from fis into a byte array (b).
9. Close fis.
10. Convert byte array (b) to a string (fileContent).
11. Print "Contents of " + fileName + ":".
12. Print fileContent.
13. Close the Scanner (sc).

PROGRAM : -

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

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

      // Prompt user to enter the file name
      System.out.print("Enter the file name: ");
      String fileName = sc.nextLine();

      // Write to the file
      FileOutputStream fos = new FileOutputStream(fileName);
      System.out.println("Enter text to insert: ");
      String text = sc.nextLine();
      fos.write(text.getBytes());
```

```
        fos.close();

        // Read from the file
        FileInputStream fis = new FileInputStream(fileName);
        byte[] b = new byte[fis.available()];
        fis.read(b);
        fis.close();

        // Display contents on the console
        String fileContent = new String(b);
        System.out.println("Contents of " + fileName + ":");
        System.out.println(fileContent);
        sc.close();
    }
}
```

OUTPUT: -



```
PS C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java> java WriteRead
Enter the file name: testfile
Enter text to insert:
Test Data
Contents of testfile:
Test Data
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 27

Date: 19/04/2024

AIM: -

Write a program to copy one file to another.

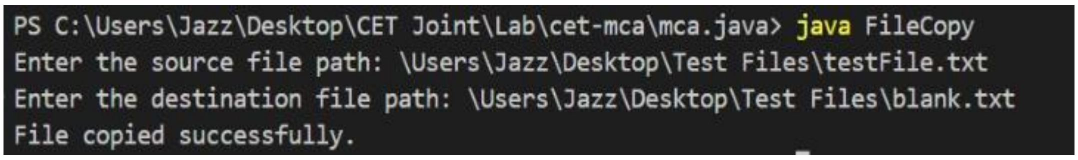
ALGORITHM: -

1. Create BufferedReader (reader) for user input.
2. Prompt user for source file path, store in sourceFile.
3. Prompt user for destination file path, store in destinationFile.
4. Open input stream (in) for source file, output stream (out) for destination file.
5. Read from input stream into buffer, write to output stream until end of file.
6. Close input and output streams.
7. Close BufferedReader.

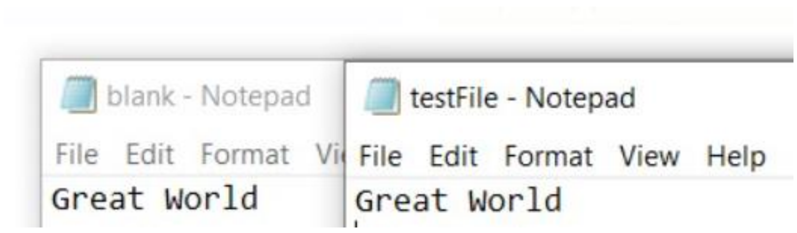
PROGRAM : -

```
import java.io.*;

public class FileCopy {
    public static void main(String[] args) throws IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

        System.out.print("Enter the source file path: ");
        String sourceFile = reader.readLine();

        System.out.print("Enter the destination file path: ");
        String destinationFile = reader.readLine();

        try (InputStream in = new FileInputStream(sourceFile);
             OutputStream out = new FileOutputStream(destinationFile)) {

            byte[] buffer = new byte[1024];
            int bytesRead;

            while ((bytesRead = in.read(buffer)) != -1) {
                out.write(buffer, 0, bytesRead);
            }
            System.out.println("File copied successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred during file copy: " + e.getMessage());
        } finally {
            reader.close();
        }
    }
}
```

OUTPUT: -



```
PS C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java> java FileCopy
Enter the source file path: \Users\Jazz\Desktop\Test Files\testFile.txt
Enter the destination file path: \Users\Jazz\Desktop\Test Files\blank.txt
File copied successfully.
```



```
blank - Notepad
File  Edit  Format  Vi
Great World
```

```
testFile - Notepad
File  Edit  Format  View  Help
Great World
```

RESULT: -

The program was executed successfully, and output is obtained.

EXERCISE 28

Date: 26/04/2024

AIM: -

Write a program that reads from a file having integers. Copy even numbers and odd numbers to separate files.

ALGORITHM: -

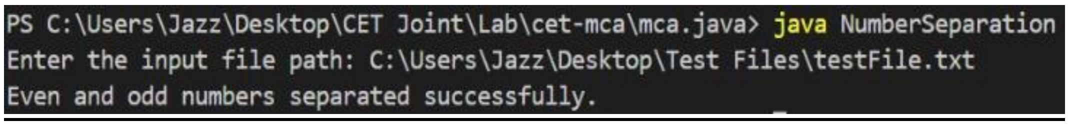
1. Create a BufferedReader (reader) to read user input.
2. Prompt the user to enter the input file path and store it in inputFile.
3. Define file paths for evenFile and oddFile.
4. Create a BufferedReader (fileReader) to read from the input file.
5. Create a BufferedWriter (evenWriter) to write even numbers to the even file.
6. Create a BufferedWriter (oddWriter) to write odd numbers to the odd file.
7. Read each line from the input file, parse it to an integer, and determine if it's even or odd.
8. Write even numbers to the even file and odd numbers to the odd file.
9. Print a success message indicating the separation is complete.
10. Close the BufferedReader, BufferedWriter for even numbers, and BufferedWriter for odd numbers.
11. Close the BufferedReader for user input.

PROGRAM : -

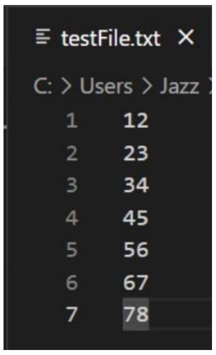
```
import java.io.*;


public class NumberSeparation {
  public static void main(String[] args) throws IOException {
    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("Enter the input file path: ");
    String inputFile = reader.readLine();
    String evenFile = "even.txt";
    String oddFile = "odd.txt";
    BufferedReader fileReader = new BufferedReader(new FileReader(inputFile));
    BufferedWriter evenWriter = new BufferedWriter(new FileWriter(evenFile));
    BufferedWriter oddWriter = new BufferedWriter(new FileWriter(oddFile));
    String line;
```

```
        while ((line = fileReader.readLine()) != null) {
            int number = Integer.parseInt(line);
            if (number % 2 == 0) {
                evenWriter.write(line);
                evenWriter.newLine();
            } else {
                oddWriter.write(line);
                oddWriter.newLine();
            }
        }
        System.out.println("Even and odd numbers separated successfully.");
        reader.close();
        fileReader.close();
        evenWriter.close();
        oddWriter.close();
    }
}
```

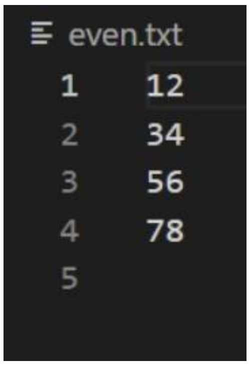
OUTPUT: -



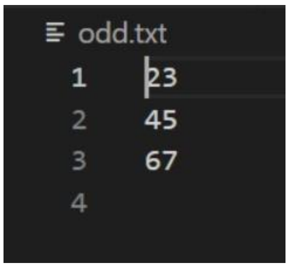
```
PS C:\Users\Jazz\Desktop\CET Joint\Lab\cet-mca\mca.java> java NumberSeparation
Enter the input file path: C:\Users\Jazz\Desktop\Test Files\testFile.txt
Even and odd numbers separated successfully.
```



```
testFile.txt
C: > Users > Jazz
12
23
34
45
56
67
78
```



```
even.txt
12
34
56
78
```



```
odd.txt
23
45
67
```

RESULT: -

The program was executed successfully, and output is obtained.