COLLEGE OF ENGINEERING CHERTHALA

<u>LAB RECORD</u>

20MCA132

OBJECT ORIENTED PROGRAMMING LAB

<u>CERTIFICATE</u>

This certifies the bona fide work of Mr./Ms. ……………………………………....., Registration Number ……………………., Second Semester MCA, College of Engineering Cherthala, for the academic year 2023-24.

Teacher In Charge External Examiner

Internal Examiner

INDEX

Experiment No.1 **Date: 1-2-2024**

Class and Object

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 the Product Class:
    a) Create the variables: p_code, p_name, price.
    b) Create a constructor to initialize these variables.
    c) Define a method display to print the product details.
2) Inside the main method:
    a) Create three Product objects with specified details.
    b) Call the display method for each product to print its details.
    c) Initialize a variable lowest to the first product.
    d) Compare the price of the second product with lowest, and update lowest
       if the second product has a lower price.
    e) Compare the price of the third product with lowest, and update lowest if
       the third product has a lower price.
    f) Print the price of the lowest product as the lowest price.
```

Source Code:

```
class Product {
   String p_code;
   String p_name;
   int price;

   Product(String c, String n, int p) {
      p_code = c;
      p_name = n;
      price = p;
   }

   void display() {
      System.out.println(p_code + " " + p_name + " " + price);
   }
}
```

```
class Products {
   public static void main(String args[]) {
      Product p1 = new Product("p09", "refrigerator", 45000);
      Product p2 = new Product("p13", "air conditioner", 34000);
      Product p3 = new Product("p01", "television", 12000);

      p1.display();
      p2.display();
      p3.display();

      Product lowest = p1;
      if (p2.price < lowest.price) {
         lowest = p2;
      }
      if (p3.price < lowest.price) {
         lowest = p3;
      }

      System.out.println(lowest.p_name + " has the lowest price");
   }
}
```

Output:

```
p09 refrigerator 45000
p13 air conditioner 34000
p01 television 12000
television has the lowest price
```

Result:

Program is executed successfully and output is obtained

Experiment No.2 **Date: 6-2-2024**

Matrix Addition

Aim:

Read 2 matrices from the console and perform matrix addition.

Algorithm:

1) Import the `Scanner` class.
2) Define the `MatrixAddition` class.
3) Define the `main` method:
    a) Create a `Scanner` object.
    b) Prompt the user to enter the number of rows for the matrices and read the input.
    c) Prompt the user to enter the number of columns for the matrices and read the input.
    d) Create three 2D arrays `matrixA`, `matrixB`, and `sumMatrix` of size `rows x columns`.
    e) Prompt the user to enter the elements of the first matrix and read them into `matrixA` using the `readMatrix` method.
    f) Prompt the user to enter the elements of the second matrix and read them into `matrixB` using the `readMatrix` method.
    g) Add the two matrices `matrixA` and `matrixB` element-wise and store the result in `sumMatrix` using the `addMatrices` method.
    h) Print the sum matrix `sumMatrix` using the `printMatrix` method.
    i) Close the `Scanner` object.
4) Define the `readMatrix` method:
    a) Accept a `Scanner` object and a 2D array `matrix` as parameters.
    b) Use nested loops to read elements from the user and store them in the `matrix`.
5) Define the `addMatrices` method:
    a) Accept two 2D arrays `matrixA` and `matrixB`, and a 2D array `sumMatrix` as parameters.
    b) Use nested loops to add the corresponding elements of `matrixA` and `matrixB` and store the result in `sumMatrix`.
6) Define the `printMatrix` method:
    a) Accept a 2D array `matrix` as a parameter.
    b) Use nested loops to print the elements of the `matrix`.

Source Code:

MatrixAddition.java:

```
import java.util.Scanner;

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

        System.out.println("Enter the number of columns:");
        int columns = scanner.nextInt();

        int[][] matrixA = new int[rows][columns];
        int[][] matrixB = new int[rows][columns];
        int[][] sumMatrix = new int[rows][columns];

        System.out.println("Enter the elements of the first matrix:");
        readMatrix(scanner, matrixA);
        System.out.println("Enter the elements of the second matrix:");
        readMatrix(scanner, matrixB);

        addMatrices(matrixA, matrixB, sumMatrix);

        System.out.println("The sum matrix is:");
        printMatrix(sumMatrix);
        scanner.close();
    }
    static void readMatrix(Scanner scanner, int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }
    }

    static void addMatrices(int[][] matrixA, int[][] matrixB, int[][] sumMatrix) {
        for (int i = 0; i < matrixA.length; i++) {
            for (int j = 0; j < matrixA[0].length; j++) {
```

```
            sumMatrix[i][j] = matrixA[i][j] + matrixB[i][j];
         }
      }
   }
   static void printMatrix(int[][] matrix) {
      for (int i = 0; i < matrix.length; i++) {
         for (int j = 0; j < matrix[0].length; j++) {
            System.out.print(matrix[i][j] + "\t");
         }
         System.out.println();
      }
   }
}
```

Output:

```
Enter the number of rows:
3
Enter the number of columns:
2
Enter the elements of the first matrix:
1 5
2 3
4 7
Enter the elements of the second matrix:
5 6
2 1
6 3
The sum matrix is:
6	11
4	4
10	10
```

Result:

Program is executed successfully and output is obtained.

Experiment No.3 **Date: 13-2-2024**

Add Complex Numbers

Aim:

Add complex numbers.

Algorithm:

```
1) Define the `Complex` class with attributes `real` and `imaginary`.
2) Implement a constructor in the `Complex` class to initialize `real` and
   `imaginary` with given values.
3) Define a static method `sum` in the `Complex` class:
   a) Create and return a new `Complex` object with the sum of the real parts
      and the sum of the imaginary parts of the input complex numbers.
4) In the `main` method:
   a) Create two `Complex` objects `first` and `second` with given real and
      imaginary parts.
   b) Call the `sum` method with `first` and `second` as arguments and store the
      result in `result`.
   c) Print the real and imaginary parts of `result`.
```

Source Code:

Complex.java:

```java
class Complex {
  int real;
  int imaginary;

  public Complex(int real, int imaginary) {
    this.real = real;
    this.imaginary = imaginary;
  }

  public static Complex sum(Complex c1, Complex c2) {
    return new Complex(c1.real + c2.real, c1.imaginary + c2.imaginary);
  }

  public static void main(String[] args) {
    Complex first = new Complex(1, 2);
    Complex second = new Complex(32, 22);
    Complex result = sum(first, second);
    System.out.println("Sum is: " + result.real + "+" + result.imaginary + "i");
  }
}
```

Output:

```
Sum is: 33+24i
```

Result:

Program is executed successfully and output is obtained.

Experiment No.4 **Date: 16-2-2024**

Symmetric Matrix

Aim:

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

Algorithm:

1) Define the `Matrix` class with attributes `size` (size of the matrix), `arr` (2D array to store the matrix elements), and a `Scanner` object `in`.
2) Implement a constructor in the `Matrix` class to initialize `size` and create the `arr` array.
3) Define the `createMatrix` method in the `Matrix` class:
    a) Prompt the user to enter the values of the matrix.
    b) Read the values into the `arr` array.
4) Define the `displayMatrix` method in the `Matrix` class:
    a) Display the elements of the matrix.
5) Define the `isSymmetric` method in the `Matrix` class:
    a) Check if the matrix is symmetric by comparing `arr[i][j]` with `arr[j][i]` for each pair of indices `(i, j)`.
    b) If any pair of elements is not equal, return `false`.
    c) If all pairs are equal, return `true`.
6) Define the `Symmetric` class with the `main` method:
    a) Create a `Scanner` object `scanner`.
    b) Prompt the user to enter the size of the matrix.
    c) Read the size of the matrix.
    d) Create a `Matrix` object `matrix` with the given size.
    e) Call the `createMatrix` method to create the matrix.
    f) Display the matrix using the `displayMatrix` method.
    g) Check if the matrix is symmetric using the `isSymmetric` method.
    h) Print "It's symmetric." if the matrix is symmetric, otherwise print "It's not symmetric."
    i) Close the `Scanner` object `scanner`.

Source Code:

Symmetric.java:

```
import java.util.Scanner;

class Matrix {
    int size;
    int[][] arr;
    Scanner in = new Scanner(System.in);

    Matrix(int size) {
        this.size = size;
        arr = new int[size][size];
    }

    void createMatrix() {
        System.out.println("Enter the values of the matrix: ");
        for (int i = 0; i < size; ++i)
            for (int j = 0; j < size; ++j)
                arr[i][j] = in.nextInt();
    }

    void displayMatrix() {
        for (int i = 0; i < size; ++i) {
            for (int j = 0; j < size; ++j)
                System.out.print(arr[i][j] + " ");
            System.out.println();
        }
    }

    boolean isSymmetric() {
        for (int i = 0; i < size; ++i)
            for (int j = 0; j < size; ++j)
                if (arr[i][j] != arr[j][i])
                    return false;
        return true;
    }
}

class Symmetric {
    public static void main(String args[]) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the size of the matrix: ");
        int size = scanner.nextInt();

        Matrix matrix = new Matrix(size);
        matrix.createMatrix();
        System.out.println();
        matrix.displayMatrix();

        if (matrix.isSymmetric())
            System.out.println("It's symmetric.");
        else
            System.out.println("It's not symmetric.");

        scanner.close();
    }
}
```

Output:

```
Enter the size of the matrix:
3
Enter the values of the matrix:
1 2 3 3 4 5 3 5 7

1 2 3
3 4 5
3 5 7
It's not symmetric.
```

Result:

Program is executed successfully and output is obtained.

Experiment No.5 **Date: 20-2-2024**

Inner Class

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 the `Cpu` class:
    a) Declare `price` as a private final variable.
    b) Create a constructor to initialize `price`.

2) Define the non-static inner class `Processor` within `Cpu`:
    a) Declare `core` and `name` as variables.
    b) Create a constructor to initialize `core` and `name`.
    c) Define the `display` method to print `price`, `core`, and `name`.

3) Define the static inner class `Ram` within `Cpu`:
    a) Declare `size` and `name` as variables.
    b) Create a constructor to initialize `size` and `name`.
    c) Define the `display` method to print `size` and `name`.

4) Define the `Innerclass` class with the `main` method:
    a) Create an instane of `Cpu` named `c` with the price set to 70000.
    b) Create an instance of the inner class `Processor` using `c` with `core` set to
       16 and `name` set to "Intel", and name it `p`.
    c) Call the `display` method on `p`.
    d) Create an instance of the static inner class `Ram` with `size` set to 16 and
       `name` set to "Adata", and name it `r`.
    e) Call the `display` method on `r`.
```

Source Code:

Innerclass.java:

```
class Cpu {
   int price;

   public Cpu(int price) {
      this.price = price;
   }
```

```
    class Processor {
        int core;
        String name;

        Processor(int core, String name) {
            this.core = core;
            this.name = name;
        }

        void display() {
            System.out.println("Price : " + price);
            System.out.println("Core : " + core);
            System.out.println("Brand : " + name);
        }
    }
    static class Ram {
        int size;
        String name;

        Ram(int size, String name) {
            this.size = size;
            this.name = name;
        }
        void display() {
            System.out.println("Size : " + size);
            System.out.println("Brand : " + name);
        }
    }
}

class Innerclass {
    public static void main(String args[]) {
        Cpu c = new Cpu(70000);
        Cpu.Processor p = c.new Processor(16, "Intel");
        p.display();
        Cpu.Ram r = new Cpu.Ram(16, "Adata");
        r.display();
    }
}
```

Output:

```
Price : 70000
Core : 16
Brand : Intel
Size : 16
Brand : Adata
```

Result:

Program is executed successfully and output is obtained.

Experiment No.6 **Date: 27-2-2024**

Sort Array

Aim:

Program to Sort strings.

Algorithm:

1) Import necessary packages: java.util.Scanner and java.util.Arrays.
2) Define the `SortStrings` class with the `main` method:
   a) Create a Scanner object `sc` for user input.
   b) Prompt the user to enter the size of the array.
   c) Read the integer input `n` for the size of the array and consume the newline character.
3) Initialize an array of Strings `arr` with size `n`.
4) Prompt the user to enter the elements of the array using a loop.
5) Print the original array elements using a loop.
6) Sort the array using `Arrays.sort(arr)`.
7) Print the sorted array elements using `Arrays.toString(arr)`.
8) Close the Scanner object `sc`.

Source Code:

SortStrings.java:

```
import java.util.*;
class SortStrings {
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the size of the array:");
      int n = sc.nextInt();
      sc.nextLine();  // Consume the newline
      String[] arr = new String[n];
      System.out.println("Enter the elements:");
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextLine();
      }
      System.out.println("Array elements are:");
      for (String str : arr) {
         System.out.println(str);
      }
      Arrays.sort(arr);
```

```
        System.out.println("Array elements after sorting:");
        System.out.println(Arrays.toString(arr));
        sc.close();
    }
}
```

Output:

```
Enter the size of the array:
3
Enter the elements:
Apple
Orange
Grapes
Array elements are:
Apple
Orange
Grapes
Array elements after sorting:
[Apple, Grapes, Orange]
```

Result:

Program is executed successfully and output is obtained.

Experiment No.7 **Date: 1-3-2024**

Search Array

Aim:

Search an element in an array.

Algorithm:

```
1)  Import necessary packages: java.util.Scanner.
2)  Define the `SearchElement` class with the `main` method:
    a)  Create a Scanner object `sc` for user input.
    b)  Prompt the user to enter the size of the array and read the integer input
        `n` for the size of the array.
    c)  Initialize an integer array `arr` with size `n`.
    d)  Prompt the user to enter the elements of the array using a loop.
    e)  Prompt the user to enter the element to search and read the integer input
        `s`.
    f)  Initialize a boolean variable `found` to false and an integer variable
        `position` to -1.
    g)  Use a loop to iterate through each element of the array:
    h)  If the current element is equal to the element to search, set `found` to
        true, set `position` to the current index plus 1, and break out of the loop.
    i)  If `found` is true, print "Element found at position: " followed by the
        `position`; otherwise, print "Element not found".
    j)  Close the Scanner object `sc`.
```

Source Code:

SearchElement.java:

```java
import java.util.*;

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

        System.out.println("Enter the size of the array:");
        int n = sc.nextInt();
        int[] arr = new int[n];

        System.out.println("Enter the array elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.println("Enter the element to search:");
        int s = sc.nextInt();

        boolean found = false;
        int position = -1;
        for (int i = 0; i < n; i++) {
            if (arr[i] == s) {
                found = true;
                position = i + 1;
                break;
            }
        }

        if (found) {
            System.out.println("Element found at position: " + position);
        } else {
            System.out.println("Element not found");
        }

        sc.close();
    }
}
```

Output:

```
Enter the size of the array:
5
Enter the array elements:
12 35 68 63 24
Enter the element to search:
35
Element found at position: 2
```

Result:

Program is executed successfully and output is obtained.

Experiment No.8 **Date: 2-3-2024**

String Manipulations

Aim:

Perform string manipulations.

Algorithm:

```
1)  Import the Scanner class from the java.util package.
2)  Create a Scanner object to read user input.
3)  Prompt the user to enter a string.
4)  Read the input string using the Scanner object.
5)  Convert the input string to lowercase.
6)  Display the lowercase string.
7)  Convert the input string to uppercase.
8)  Display the uppercase string.
9)  Extract and display the substring from index 1 to 5 of the input string.
10) Display the length of the input string.
```

Source Code:

StringManipulation.java:

```
import java.util.*;

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

      System.out.println("Enter a string:");
      String s1 = sc.nextLine();

      String s1Lower = s1.toLowerCase();
      System.out.println("Lower case: " + s1Lower);

      String s1Upper = s1.toUpperCase();
      System.out.println("Upper case: " + s1Upper);

      System.out.println("Substring (1 to 5): " + s1.substring(1, 5));

      System.out.println("Length of the string is: " + s1.length());
   }
}
```

Output:

```
Enter a string:
Good Afternoon
Lower case: good afternoon
Upper case: GOOD AFTERNOON
Substring (1 to 5): ood
Length of the string is: 14
```

Result:

Program is executed successfully and output is obtained.

Experiment No.9 **Date: 4-3-2024**

Array of Objects

Aim:

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

Algorithm:

1) Import the Scanner class.
2) Define a class Employee with id, name, and salary attributes.
3) Create a constructor in Employee to initialize the attributes.
4) Define a display method in Employee to print employee details.
5) Define a class Main.
6) Inside Main's main method:
   a) Create a Scanner object.
   b) Prompt for and read the number of employees.
   c) Create an array of Employee objects based on the number of employees.
   d) Use a loop to input details for each employee and store them in the array.
   e) Prompt for and read the employee ID to be searched.
   f) Use a loop to search for the employee in the array.
   g) If found, set a flag and break the loop.
   h) Display whether the employee exists or not.
   i) If the employee exists, display the employee details.

Source Code:

Main.java:

```
import java.util.Scanner;

class Employee {
   int id;
   String name;
   int salary;

   Employee(int id, String name, int salary) {
      this.id = id;
      this.name = name;
      this.salary = salary;
   }

   void display() {
        System.out.printf("%-10d%-20s%-10d%n", id, name, salary);
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number of employees:");
        int n = in.nextInt();
        in.nextLine(); // Consume newline
        Employee[] employees = new Employee[n];

        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for employee " + (i + 1) + ":");
            System.out.print("ID: ");
            int id = in.nextInt();
            in.nextLine(); // Consume newline
            System.out.print("Name: ");
            String name = in.nextLine();
            System.out.print("Salary: ");
            int salary = in.nextInt();
            employees[i] = new Employee(id, name, salary);
        }

        System.out.println("Enter the employee ID to search:");
        int searchId = in.nextInt();
        boolean found = false;
        for (Employee emp : employees) {
            if (emp.id == searchId) {
                found = true;
                System.out.println("Employee found:");
                emp.display();
                break;
            }
        }

        if (!found) {
            System.out.println("Employee not found.");
        }
    }
}
```

Output:

```
Enter the number of employees:
3
Enter details for employee 1:
ID: 2500
Name: Kevin
Salary: 25000
Enter details for employee 2:
ID: 2501
Name: James
Salary: 20000
Enter details for employee 3:
ID: 2502
Name: Rob
Salary: 30000
Enter the employee ID to search:
2501
Employee found:
2501    James         20000
```

Result:

Program is executed successfully and output is obtained.

Experiment No.10 **Date: 12-3-2024**

Method Overloading

Aim:
Area of different shapes using overloaded functions.

Algorithm:

```
1) Import the Scanner class.
2) Define a class Shapes.
3) Inside the Shapes class:
   a) Define a method area that takes two integers for the rectangle's length
      and breadth and prints its area.
   b) Define a method area that takes a float for the circle's radius and prints its
      area.
   c) Define a method area that takes a double for the square's side length and
      prints its area.
4) Define a class Overload.
5) Inside the Overload class:
   a) Define the main method.
   b) Create a Scanner object for user input.
   c) Create an instance of the Shapes class.
   d) Prompt and read the length and breadth for the rectangle.
   e) Call the area method of Shapes with the length and breadth.
   f) Prompt and read the radius for the circle.
   g) Call the area method of Shapes with the radius.
   h) Prompt and read the side length for the square.
   i) Call the area method of Shapes with the side length.
```

Source Code:

Overload.java:
```
import java.util.Scanner;

class Shapes {
  void area(int l, int b) {
    System.out.println("Area of rectangle is " + l * b);
  }

  void area(float r) {
    System.out.println("Area of circle is " + 3.14 * r * r);
  }
```

```
    void area(double s) {
        System.out.println("Area of square is " + s * s);
    }
}

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

        Shapes shapes = new Shapes();

        System.out.println("Enter the length and breadth ");
        int l = in.nextInt();
        int b = in.nextInt();
        shapes.area(l, b);

        System.out.println("Enter the radius ");
        float r = in.nextFloat();
        shapes.area(r);

        System.out.println("Enter the side ");
        double s = in.nextDouble();
        shapes.area(s);
    }
}
```

Output:

```
Enter the length and breadth
4
5
Area of rectangle is 20
Enter the radius
3
Area of circle is 28.259999999999998
Enter the side
4
Area of square is 16.0
```

Result:

Program is executed successfully and output is obtained.

Experiment No.11 **Date: 22-3-2024**

Simple Inheritance

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) Define the Employee class:
   a) Declare emp_id, emp_name, emp_salary, and address variables.
   b) Create a constructor to initialize these variables.
2) Define the Teacher class extending Employee:
   a) Declare department and subject variables.
   b) Create a constructor to initialize all variables.
   c) Define a display method to print teacher details.
3) Define the Inheritance class with the main method:
   a) Create Scanner objects for input.
   b) Prompt and read the number of teachers.
   c) Initialize an array of Teacher objects.
   d) For each teacher:
      i) Prompt and read emp_id, emp_name, emp_salary, address,
         department, and subject.
      ii) Create a new Teacher object and store it in the array.
   e) Prompt and read the emp_id to search.
   f) Initialize a boolean flag for search result.
   g) Loop through the Teacher array using an enhanced for loop to find the
      teacher with the given emp_id:
   h) If found, set the flag to true, call the display method, and break the loop.
   i) If the flag is false, print "Employee not found".
   j) Close the Scanner objects.
```

Source Code:

Inheritance.java:

```
import java.util.Scanner;

class Employee {
   int emp_id;
```

```
    String emp_name;
    int emp_salary;
    String address;

    Employee(int emp_id, String emp_name, int emp_salary, String address) {
        this.emp_id = emp_id;
        this.emp_name = emp_name;
        this.emp_salary = emp_salary;
        this.address = address;
    }
}
class Teacher extends Employee {
    String department;
    String subject;

    Teacher(int emp_id, String emp_name, int emp_salary, String address, String
department, String subject) {
        super(emp_id, emp_name, emp_salary, address);
        this.department = department;
        this.subject = subject;
    }

    void display() {
        System.out.println("Employee id: " + this.emp_id);
        System.out.println("Employee name: " + this.emp_name);
        System.out.println("Employee salary: " + this.emp_salary);
        System.out.println("Employee address: " + this.address);
        System.out.println("Subject: " + this.subject);
        System.out.println("Department: " + this.department);
    }
}

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

        System.out.println("Enter the number of teachers:");
        int n = sc.nextInt();
        sc.nextLine();  // consume the newline
    Teacher[] teachers = new Teacher[n];

    for (int i = 0; i < n; i++) {
      System.out.println("Enter the details of teacher " + (i + 1) + ":");

      System.out.print("Subject: ");
      String subject = s.nextLine();

      System.out.print("Department: ");
      String department = s.nextLine();

      System.out.print("Employee id: ");
      int emp_id = sc.nextInt();
      sc.nextLine();  // consume the newline

      System.out.print("Employee name: ");
      String emp_name = s.nextLine();

      System.out.print("Salary: ");
      int emp_salary = sc.nextInt();
      sc.nextLine();  // consume the newline

      System.out.print("Address: ");
      String address = s.nextLine();

      teachers[i] = new Teacher(emp_id, emp_name, emp_salary, address,
department, subject);
    }
    System.out.print("Enter the employee number to be searched: ");
    int no = s.nextInt();

    boolean found = false;
    for (Teacher teacher : teachers) {
      if (teacher.emp_id == no) {
        found = true;
        teacher.display();
        break;
      }
    }
    if (!found) {
      System.out.println("Employee not found");
```

```
        }

        sc.close();
        s.close();
    }
}
```

Output:

```
Enter the number of teachers:
3
Enter the details of teacher 1:
Subject: Physics
Department: Science
Employee id: 101
Employee name: Alex
Salary: 25000
Address: Kollam
Enter the details of teacher 2:
Subject: Maths
Department: Mathematics
Employee id: 104
Employee name: Vinod
Salary: 30000
Address: Thrissur
Enter the details of teacher 3:
Subject: AI
Department: Computer Science
Employee id: 110
Employee name: Dennis
Salary: 26000
Address: Kottyam
Enter the employee number to be searched: 110
Employee id: 110
Employee name: Dennis
Employee salary: 26000
Employee address: Kottyam
Subject: AI
Department: Computer Science
```

Result:

Program is executed successfully and output is obtained.

Experiment No.12 **Date: 26-4-2024**

Multilevel Inheritance

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)  Define class Person
    a)  Declare variables: name, gender, address, age
    b)  Define constructor to initialize variables using Scanner input
2)  Define class Employee extending Person
    a)  Declare variables: empid, compname, quali, salary
    b)  Define constructor to initialize variables using Scanner input
3)  Define class Teacher1 extending Employee
    a)  Declare variables: subject, department, teacherid
    b)  Define constructor to initialize variables using Scanner input
    c)  Define method display() to print all variables
4)  In main method of Teacher1
    a)  Declare variable n
    b)  Initialize Scanner
    c)  Prompt user for number of teachers
    d)  Create array of Teacher1 objects of size n
    e)  Initialize each Teacher1 object using constructor
    f)  Call display() method for each Teacher1 object to print details
```

Source Code:

Teacher1.java:

```
import java.util.Scanner;

class Person {
   String name;
   String gender;
   String address;
   int age;
```

```
    Person() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Name");
        name = sc.next();
        System.out.println("Enter Gender");
        gender = sc.next();
        System.out.println("Enter Address");
        address = sc.next();
        System.out.println("Enter Age");
        age = sc.nextInt();
    }
}

class Employee extends Person {
    int empid;
    String compname, quali;
    float salary;

    Employee() {
        Scanner x = new Scanner(System.in);
        System.out.println("Enter Employee ID");
        empid = x.nextInt();
        System.out.println("Enter Company Name");
        compname = x.next();
        System.out.println("Enter Qualification");
        quali = x.next();
        System.out.println("Enter Salary");
        salary = x.nextFloat();
    }
}

class Teacher1 extends Employee {
    String subject, department;
    int teacherid;

    Teacher1() {
        Scanner x = new Scanner(System.in);
        System.out.println("Enter Subject");
        subject = x.next();
        System.out.println("Enter Department");
        department = x.next();
        System.out.println("Enter Teacher ID");
        teacherid = x.nextInt();
    }

    void display() {
        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("COMPANY NAME: " + compname);
        System.out.println("QUALIFICATION: " + quali);
        System.out.println("SALARY: " + salary);
        System.out.println("SUBJECT: " + subject);
        System.out.println("DEPARTMENT: " + department);
        System.out.println("TEACHER ID: " + teacherid + "\n");
    }

    public static void main(String[] args) {
        int n;
        Scanner y = new Scanner(System.in);
        System.out.println("Enter number of teachers:");
        n = y.nextInt();
        Teacher1[] teachers = new Teacher1[n];
        for (int i = 0; i < n; i++) {
            teachers[i] = new Teacher1();
        }
        System.out.println("DETAILS:\n");
        for (int i = 0; i < n; i++) {
            teachers[i].display();
        }
    }
}
```

Output:

```
Enter number of teachers:
2
Enter Name
```

```
John
Enter Gender
Male
Enter Address
123 Main St
Enter Age
35
Enter Employee ID
101
Enter Company Name
ABC Corp
Enter Qualification
MSc
Enter Salary
50000
Enter Subject
Math
Enter Department
Science
Enter Teacher ID
201

Enter Name
Jane
Enter Gender
Female
Enter Address
456 Elm St
Enter Age
30
Enter Employee ID
102
Enter Company Name
XYZ Inc
Enter Qualification
PhD
Enter Salary
55000
Enter Subject
Physics
Enter Department
```

```
Science
Enter Teacher ID
202

DETAILS:

NAME: John
GENDER: Male
ADDRESS: 123 Main St
AGE: 35
EMPLOYEE ID: 101
COMPANY NAME: ABC Corp
QUALIFICATION: MSc
SALARY: 50000.0
SUBJECT: Math
DEPARTMENT: Science
TEACHER ID: 201

NAME: Jane
GENDER: Female
ADDRESS: 456 Elm St
AGE: 30
EMPLOYEE ID: 102
COMPANY NAME: XYZ Inc
QUALIFICATION: PhD
SALARY: 55000.0
SUBJECT: Physics
DEPARTMENT: Science
TEACHER ID: 202
```

Result:

Program is executed successfully and output is obtained.

Experiment No.13 **Date: 1-4-2024**

Hierarchical Inheritance

Aim:

Write a program has class Publisher, Book, Literature and Fiction. Read the information and print the details of books from either the category, using inheritance.

Algorithm:

1) Define the Publisher class:
   a) Declare publisherName and agencyName variables.
   b) Create a constructor to initialize these variables.

2) Define the Book class extending Publisher:
   a) Declare bookName, author, price, and type variables.
   b) Create a constructor to initialize all variables.

3) Define the Literature class extending Book:
   a) Create a constructor to initialize all variables.
   b) Define a display method to print literature book details.

4) Define the Fiction class extending Book:
   a) Create a constructor to initialize all variables.
   b) Define a display method to print fiction book details.

5) Define the Category class with the main method:
   a) Create Scanner objects for input.
   b) Prompt and read details for a literature book.
   c) Create a Literature object with the entered details.
   d) Prompt and read details for a fiction book.
   e) Create a Fiction object with the entered details.
   f) Display the details of both books.

Source Code:

Category.java:

```
import java.util.Scanner;

class Publisher {
   String publisherName;
```

```
    String agencyName;

    Publisher(String publisherName, String agencyName) {
        this.publisherName = publisherName;
        this.agencyName = agencyName;
    }
}

class Book extends Publisher {
    String bookName;
    String author;
    int price;
    String type;

    Book(String publisherName, String agencyName, String bookName, String
author, int price, String type) {
        super(publisherName, agencyName);
        this.bookName = bookName;
        this.author = author;
        this.price = price;
        this.type = type;
    }
}

class Literature extends Book {
    Literature(String publisherName, String agencyName, String bookName,
String author, int price, String type) {
        super(publisherName, agencyName, bookName, author, price, type);
    }

    void display() {
        System.out.println("Publisher Name: " + this.publisherName);
        System.out.println("Agency Name: " + this.agencyName);
        System.out.println("Book Name: " + this.bookName);
        System.out.println("Author: " + this.author);
        System.out.println("Price: " + this.price);
        System.out.println("Type: " + this.type);
    }
}

class Fiction extends Book {
  Fiction(String publisherName, String agencyName, String bookName, String
author, int price, String type) {
    super(publisherName, agencyName, bookName, author, price, type);
  }

  void display() {
    System.out.println("Publisher Name: " + this.publisherName);
    System.out.println("Agency Name: " + this.agencyName);
    System.out.println("Book Name: " + this.bookName);
    System.out.println("Author: " + this.author);
    System.out.println("Price: " + this.price);
    System.out.println("Type: " + this.type);
  }
}

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

    System.out.println("Enter the details of the literature book:");
    System.out.print("Publisher Name: ");
    String publisherName1 = s.nextLine();
    System.out.print("Agency Name: ");
    String agencyName1 = s.nextLine();
    System.out.print("Book Name: ");
    String bookName1 = s.nextLine();
    System.out.print("Author: ");
    String author1 = s.nextLine();
    System.out.print("Price: ");
    int price1 = sc.nextInt();
    sc.nextLine(); // Consume the newline
    System.out.print("Type: ");
    String type1 = s.nextLine();

    Literature literatureBook = new Literature(publisherName1, agencyName1,
bookName1, author1, price1, type1);

    System.out.println("\nEnter the details of the fiction book:");
    System.out.print("Publisher Name: ");
    String publisherName2 = s.nextLine();
```

```
        System.out.print("Agency Name: ");
        String agencyName2 = s.nextLine();
        System.out.print("Book Name: ");
        String bookName2 = s.nextLine();
        System.out.print("Author: ");
        String author2 = s.nextLine();
        System.out.print("Price: ");
        int price2 = sc.nextInt();
        sc.nextLine(); // Consume the newline
        System.out.print("Type: ");
        String type2 = s.nextLine();

        Fiction fictionBook = new Fiction(publisherName2, agencyName2,
bookName2, author2, price2, type2);

        System.out.println("\nDetails of the literature book:");
        literatureBook.display();

        System.out.println("\nDetails of the fiction book:");
        fictionBook.display();
    }
}
```

Output:

```
Enter the details of the literature book:
Publisher Name: Penguin Random House
Agency Name: Penguin Group
Book Name: To Kill a Mockingbird
Author: Harper Lee
Price: 399
Type: Literature

Enter the details of the fiction book:
Publisher Name: HarperCollins
Agency Name: HarperCollins Publishers
Book Name: The Hobbit
Author: J.R.R. Tolkien
Price: 499
Type: Fiction
```

```
Details of the literature book:
Publisher Name: Penguin Random House
Agency Name: Penguin Group
Book Name: To Kill a Mockingbird
Author: Harper Lee
Price: 399
Type: Literature

Details of the fiction book:
Publisher Name: HarperCollins
Agency Name: HarperCollins Publishers
Book Name: The Hobbit
Author: J.R.R. Tolkien
Price: 499
Type: Fiction
```

Result:

Program is executed successfully and output is obtained.

Experiment No.14 **Date: 4-4-2024**

Multiple Inheritance Using Interface

Aim:

Create classes Student and Sports. Create another class Result inherited from Student and Sports. Display the academic and sports score of a student.

Algorithm:

```
1) Define the Sports interface with sportsGetData and sportsDispData methods.
2) Define the Student class with name, roll_no, S1, and S2 variables and a
   constructor to initialize them.
3) Define the Result class extending Student and implementing Sports:
   a) Add rank, item, and in (Scanner) variables.
   b) Define a constructor to call the superclass constructor.
   c) Define an acadamic method to calculate and print the percentage.
   d) Implement the sportsGetData method to input item and rank.
   e) Implement the sportsDispData method to display item and rank.
4) Define the Studentresult class with the main method:
   a) Create a Scanner object.
   b) Input name, roll, S1, and S2.
   c) Create a Result object.
   d) Call sportsGetData, acadamic, and sportsDispData methods.
```

Source Code:

Studentresult.java:

```
import java.util.Scanner;

interface Sports
{
   public void sportsGetData();
   public void sportsDispData();
}

class Student
{
   String name;
   int roll_no;
   int S1,S2;

   Student(String name, int roll_no, int S1, int S2)
```

```
    {
        this.name = name;
        this.roll_no = roll_no;
        this.S1 = S1;
        this.S2 = S2;
    }
}

class Result extends Student implements Sports
{
    int rank;
    String item;
    Scanner in = new Scanner(System.in);
    Result(String name, int roll_no, int S1, int S2)
    {
        super(name, roll_no, S1, S2);
    }

    void acadamic()
    {
        int percentage = (S1+S2)*100/200;
        System.out.print("Percentage = " + percentage);
    }

    public void sportsGetData()
    {
        System.out.print("Enter the item name ");
        item = in.nextLine();
        System.out.print("Enter the rank ");
        rank = in.nextInt();
    }

    public void sportsDispData()
    {
        System.out.println("Sports item :" + item);
        System.out.print("Rank : "+ rank);
    }

}

class Studentresult
```

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

      System.out.print("Enter tha name ");
      String name = in.nextLine();
      System.out.print("Enter Roll no ");
      int roll = in.nextInt();
      System.out.print("Enter mark in first subject ");
      int S1 = in.nextInt();
      System.out.print("Enter mark in second subject ");
      int S2 = in.nextInt();

      Result r = new Result(name,roll,S1,S2);
      r.sportsGetData();

      r.acadamic();
      System.out.println();
      r.sportsDispData();

   }
}
```

Output:

```
Enter the name: John Doe
Enter Roll no: 101
Enter mark in first subject: 85
Enter mark in second subject: 90
Enter the item name: Basketball
Enter the rank: 1
Percentage = 87
Sports item: Basketball
Rank: 1
```

Result:

Program is executed successfully and output is obtained.

Experiment No.15 **Date: 6-4-2024**

Interface

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 interface Shape
    a)  Define method: area()
    b)  Define method: perimeter()
2)  Define class Circle that implements Shape
    a)  Declare variable r
    b)  Define constructor to initialize r
    c)  Implement method area() to calculate and display area of circle
    d)  Implement method perimeter() to calculate and display perimeter of circle
3)  Define class Rectangle that implements Shape
    a)  Declare variables l, b
    b)  Define constructor to initialize l and b
    c)  Implement method area() to calculate and display area of rectangle
    d)  Implement method perimeter() to calculate and display perimeter of
        rectangle
4)  Define main class Interface
    a)  Declare and initialize Scanner
    b)  Declare variable ch
    c)  Do-while loop
        i)   Display menu
        ii)  Get user choice
        iii) Switch-case for choice
    d)  End do-while loop
5)  Close Scanner
```

Source Code:

Interface.java:

```
import java.util.Scanner;

interface Shape {
   void area();
   void perimeter();
}
```

```
class Circle implements Shape {
  double r;

  Circle(double r) {
    this.r = r;
  }

  public void area() {
    System.out.println("Area of the circle is " + 3.14 * r * r);
  }

  public void perimeter() {
    System.out.println("Perimeter of the circle is " + 2 * 3.14 * r);
  }
}

class Rectangle implements Shape {
  int l, b;

  Rectangle(int l, int b) {
    this.l = l;
    this.b = b;
  }

  public void area() {
    System.out.println("Area of rectangle is " + l * b);
  }

  public void perimeter() {
    System.out.println("Perimeter of rectangle is " + 2 * (l + b));
  }
}

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

    do {
      System.out.println("\nEnter your choice:");
```

```
            System.out.println("1. To Find area and perimeter of a circle");
            System.out.println("2. To find area and perimeter of rectangle");
            System.out.println("0. Exit");
            ch = in.nextInt();
            switch (ch) {
              case 1:
                System.out.println("Enter the radius :");
                double r = in.nextDouble();
                Circle c = new Circle(r);
                c.area();
                c.perimeter();
                break;
              case 2:
                System.out.println("Enter the length and breadth :");
                int l = in.nextInt();
                int b = in.nextInt();
                Rectangle re = new Rectangle(l, b);
                re.area();
                re.perimeter();
                break;
              case 0:
                System.out.println("Exiting...");
                break;
              default:
                System.out.println("Invalid choice");
                break;
            }
        } while (ch != 0);
        in.close();
    }
}
```

Output:

```
Enter your choice:
1. To Find area and perimeter of a circle
2. To find area and perimeter of rectangle
0. Exit
1
Enter the radius :
5
Area of the circle is 78.5
```

```
Perimeter of the circle is 31.400000000000002

Enter your choice:
1. To Find area and perimeter of a circle
2. To find area and perimeter of rectangle
0. Exit
2
Enter the length and breadth :
4
5
Area of rectangle is 20
Perimeter of rectangle is 18

Enter your choice:
1. To Find area and perimeter of a circle
2. To find area and perimeter of rectangle
0. Exit
0
Exiting...
```

Result:

Program is executed successfully and output is obtained.

Experiment No.16 **Date: 9-4-2024**

Packages

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) Create a package named graphics.
2) Define an interface Figure with a method area().
3) Create classes Rectangle, Triangle, Square, and Circle in the graphics package implementing the Figure interface.
4) Implement the area() method for each class.
5) Create a test class in the default package to test the area calculation.

Source Code:

Figure.java

```
package graphics;

public interface Figure {
   double area();
}

class Rectangle implements Figure {
   private double length;
   private double breadth;

   public Rectangle(double length, double breadth) {
      this.length = length;
      this.breadth = breadth;
   }

   public double area() {
      return length * breadth;
   }
}

class Triangle implements Figure {
   private double base;
   private double height;
```

```
    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    public double area() {
        return 0.5 * base * height;
    }
}

class Square implements Figure {
    private double side;

    public Square(double side) {
        this.side = side;
    }

    public double area() {
        return side * side;
    }
}

class Circle implements Figure {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    public double area() {
        return Math.PI * radius * radius;
    }
}
```

<u>TestGraphics.java</u>

```
import graphics.*;
import java.util.Scanner;

public class TestGraphics {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter length and breadth of the rectangle: ");
        double length = scanner.nextDouble();
        double breadth = scanner.nextDouble();
        Rectangle rectangle = new Rectangle(length, breadth);
        System.out.println("Area of Rectangle: " + rectangle.area());

        System.out.print("Enter base and height of the triangle: ");
        double base = scanner.nextDouble();
        double height = scanner.nextDouble();
        Triangle triangle = new Triangle(base, height);
        System.out.println("Area of Triangle: " + triangle.area());

        System.out.print("Enter side of the square: ");
        double side = scanner.nextDouble();
        Square square = new Square(side);
        System.out.println("Area of Square: " + square.area());

        System.out.print("Enter radius of the circle: ");
        double radius = scanner.nextDouble();
        Circle circle = new Circle(radius);
        System.out.println("Area of Circle: " + circle.area());

        scanner.close();
    }
}
```

Output:

```
Enter length and breadth of the rectangle: 5 10
Area of Rectangle: 50.0
Enter base and height of the triangle: 6 8
Area of Triangle: 24.0
Enter side of the square: 4
Area of Square: 16.0
Enter radius of the circle: 3
Area of Circle: 28.274333882308138
```

Result:

Program is executed successfully and output is obtained.

Experiment No.17 **Date: 16-4-2024**

Exception Handling

Aim:

To write a java program to perform even or odd exception.

Algorithm:

```
1)  Define class Oddexception extending Exception
    a)  Define constructor to call superclass constructor with message

2)  Define class Evenexception extending Exception
    a)  Define constructor to call superclass constructor with message

3)  Define class Oddeven
    a)  Define method Oddevenx(int x) throws Oddexception, Evenexception
        i)   Check if x is even
    b)  If true, throw Evenexception with message "Even"
        i)   If false, throw Oddexception with message "Odd"

4)  Define main class Oddevenexception
        i)   Create instance of Oddeven
        ii)  Initialize Scanner
        iii) Prompt user to enter a number
        iv)  Read the number
        v)   Use try-catch block to call Oddevenx(int x)
        vi)  Catch and print Evenexception
        vii) Catch and print Oddexception
```

Source Code:

Oddevenexception.java:

```
import java.util.Scanner;

class Oddexception extends Exception {
   Oddexception(String s) {
      super(s);
   }
}
class Evenexception extends Exception {
   Evenexception(String s) {
      super(s);
   }
}


class Oddeven {
  void Oddevenx(int x) throws Oddexception, Evenexception {
    if (x % 2 == 0) {
      throw new Evenexception("Even");
    } else {
      throw new Oddexception("Odd");
    }
  }
}

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

    System.out.println("Enter the number");
    int n = in.nextInt();
    try {
      o.Oddevenx(n);
    } catch (Evenexception e) {
      System.out.println(e);
    } catch (Oddexception e) {
      System.out.println(e);
    }
  }
}
```

Output:

```
Enter the number
5
Oddexception: Odd

Enter the number
4
Evenexception: Even
```

Result:

Program is executed successfully and output is obtained.

Experiment No.18 **Date: 19-4-2024**

Threads

Aim:

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

Algorithm:

```
1) Create a class `Multiplication` that extends `Thread`.
   a) Override the `run` method to print the multiplication table of 5.
2) Create a class `PrimeNumbers` that extends `Thread`.
   a) Define a constructor that accepts an integer `n` and initializes the member
      variable `n`.
   b) Override the `run` method to call the static method `prime_N` with `n`.
   c) Define a static method `prime_N` that prints all prime numbers up to `N`.
3) In the `ThreadClass` class:
   a) Create a `Scanner` object to read user input.
   b) Prompt the user to enter the value of `n` for prime numbers.
   c) Create instances of `Multiplication` and `PrimeNumbers` threads.
   d) Start both threads.
   e) Close the `Scanner` object.
```

Source Code:

ThreadClass.java:

```java
import java.util.Scanner;

class Multiplication extends Thread {
   @Override
   public void run() {
      System.out.println("Multiplication table of 5:");
      for (int i = 1; i <= 10; i++) {
         System.out.println(i + " X 5 = " + i * 5);
      }
      System.out.println("Exiting from Thread Multiplication...");
   }
}

class PrimeNumbers extends Thread {
   private int n;

   public PrimeNumbers(int n) {
        this.n = n;
    }

    @Override
    public void run() {
        prime_N(n);
    }

    static void prime_N(int N) {
        int x, y, flag;
        System.out.println("All the Prime numbers within 1 and " + N + " are:");
        for (x = 1; x <= N; x++) {
            if (x == 1 || x == 0) {
                continue;
            }
            flag = 1;
            for (y = 2; y <= x / 2; ++y) {
                if (x % y == 0) {
                    flag = 0;
                    break;
                }
            }
            if (flag == 1) {
                System.out.print(x + "\t");
            }
        }
        System.out.println();
    }
}

public class ThreadClass {
    public static void main(String args[]) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the value of n for Prime numbers: ");
        int n = scanner.nextInt();

        Multiplication multiplicationThread = new Multiplication();
        PrimeNumbers primeNumbersThread = new PrimeNumbers(n);


        primeNumbersThread.start();
        multiplicationThread.start();

        scanner.close();
    }
}
```

Output:

```
Enter the value of n for Prime numbers: 5
Multiplication table of 5:
All the Prime numbers within 1 and 5 are:
2       3       5
1 X 5 = 5
2 X 5 = 10
3 X 5 = 15
4 X 5 = 20
5 X 5 = 25
6 X 5 = 30
7 X 5 = 35
8 X 5 = 40
9 X 5 = 45
10 X 5 = 50
Exiting from Thread Multiplication...
```

Result:

Program is executed successfully and output is obtained.

Experiment No.19 **Date: 23-4-2024**

Event Handling

Aim:

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

Algorithm:

1) Create a class `MOUSEWINDOWEX` that extends `JFrame` and implements `MouseListener` and `WindowListener`.
   a) Set the frame bounds, default close operation, and visibility in the constructor.
   b) Add `MouseListener` and `WindowListener` to the frame.
2) Implement the `MouseListener` methods in `MOUSEWINDOWEX`:
   a) `mousePressed`: Draw a blue oval and print "Mouse Pressed".
   b) `mouseReleased`: Draw a green oval and print "Mouse Released".
   c) `mouseClicked`: Draw a red oval and print "Mouse Clicked".
   d) `mouseEntered`: Draw a yellow oval and print "Mouse Entered".
   e) `mouseExited`: Draw a black oval and print "Mouse Exited".
3) Implement the `WindowListener` methods in `MOUSEWINDOWEX`:
   a) `windowActivated`: Print "WINDOW activated".
   b) `windowClosed`: Print "WINDOW closed".
   c) `windowClosing`: Print "WINDOW closing" and dispose the frame.
   d) `windowDeactivated`: Print "WINDOW deactivated".
   e) `windowDeiconified`: Print "WINDOW deiconified".
   f) `windowIconified`: Print "WINDOW iconified".
   g) `windowOpened`: Print "WINDOW opened".
4) Create a class `MOUSEWINDOW` with the `main` method to instantiate `MOUSEWINDOWEX`.

Source Code:

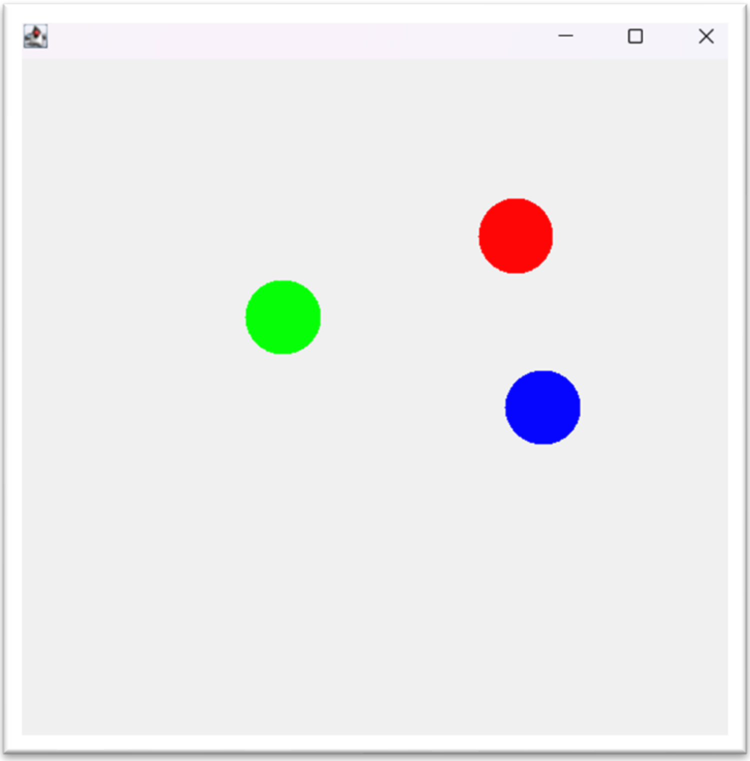
MOUSEWINDOW.java:

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


class MOUSEWINDOWEX extends JFrame implements MouseListener,
WindowListener {
  MOUSEWINDOWEX() {
    setBounds(100, 200, 500, 500);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        setVisible(true);
        addMouseListener(this);
        addWindowListener(this);
    }

    public void mousePressed(MouseEvent e) {
        Graphics g = getGraphics();
        g.setColor(Color.BLUE);
        g.fillOval(e.getX(), e.getY(), 50, 50);
        System.out.println("Mouse Pressed");
    }

    public void mouseReleased(MouseEvent e) {
        Graphics g = getGraphics();
        g.setColor(Color.GREEN);
        g.fillOval(e.getX(), e.getY(), 50, 50);
        System.out.println("Mouse Released");
    }

    public void mouseClicked(MouseEvent e) {
        Graphics g = getGraphics();
        g.setColor(Color.RED);
        g.fillOval(e.getX(), e.getY(), 50, 50);
        System.out.println("Mouse Clicked");
    }

    public void mouseEntered(MouseEvent e) {
        Graphics g = getGraphics();
        g.setColor(Color.YELLOW);
        g.fillOval(e.getX(), e.getY(), 50, 50);
        System.out.println("Mouse Entered");
    }

    public void mouseExited(MouseEvent e) {
        Graphics g = getGraphics();
        g.setColor(Color.BLACK);
        g.fillOval(e.getX(), e.getY(), 50, 50);
        System.out.println("Mouse Exited");
    }

    public void windowActivated(WindowEvent e) {
        System.out.println("WINDOW activated");
    }

    public void windowClosed(WindowEvent e) {
        System.out.println("WINDOW closed");
    }

    public void windowClosing(WindowEvent e) {
        System.out.println("WINDOW closing");
        dispose();
    }

    public void windowDeactivated(WindowEvent e) {
        System.out.println("WINDOW deactivated");
    }

    public void windowDeiconified(WindowEvent e) {
        System.out.println("WINDOW deiconified");
    }

    public void windowIconified(WindowEvent e) {
        System.out.println("WINDOW iconified");
    }

    public void windowOpened(WindowEvent e) {
        System.out.println("WINDOW opened");
    }
}

class MOUSEWINDOW {
    public static void main(String[] args) {
        MOUSEWINDOWEX frame = new MOUSEWINDOWEX();
    }
}
```

Output:

```
WINDOW opened
Mouse Entered
Mouse Pressed
Mouse Released
Mouse Pressed
Mouse Released
Mouse Clicked
Mouse Exited
Window closing
```



Result:
Program is executed successfully and output is obtained.

Experiment No.20 **Date: 23-4-2024**

Abstract Window Toolkit

Aim:

Program to find maximum of three numbers using AWT.

Algorithm:

```
1) Create a class `MaxOfThreeNumbers` that extends `Frame` and implements
   `ActionListener`.
   a) Initialize TextField components `num1`, `num2`, `num3`, and `result`.
   b) Set `result` TextField to be non-editable.
   c) Add Labels and TextFields to the Frame using `FlowLayout`.
   d) Add a Button `findMax` and set an `ActionListener` to it.
   e) Set Frame properties: title, size, and visibility.
2) Implement `actionPerformed` method to handle the button click event:
   a) Parse integers from `num1`, `num2`, and `num3` TextFields.
   b) Find the maximum of the three numbers.
   c) Set the maximum value to the `result` TextField.
3) In the `main` method, create an instance of `MaxOfThreeNumbers`.
```

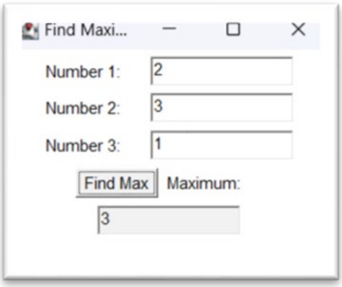
Source Code:

MaxOfThreeNumbers.java:

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

public class MaxOfThreeNumbers extends Frame implements ActionListener {
   TextField num1, num2, num3, result;
   Button findMax;

   public MaxOfThreeNumbers() {
      setLayout(new FlowLayout());
      num1 = new TextField(10);
      num2 = new TextField(10);
      num3 = new TextField(10);
      result = new TextField(10);
      result.setEditable(false);

      add(new Label("Number 1: "));
      add(num1);
      add(new Label("Number 2: "));
      add(num2);
```

```
        add(new Label("Number 3: "));
        add(num3);

        findMax = new Button("Find Max");
        add(findMax);
        add(new Label("Maximum: "));
        add(result);
        findMax.addActionListener(this);

        setTitle("Find Maximum of Three Numbers");
        setSize(250, 200);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent ae) {
        int number1 = Integer.parseInt(num1.getText());
        int number2 = Integer.parseInt(num2.getText());
        int number3 = Integer.parseInt(num3.getText());
        int max = Math.max(number1, Math.max(number2, number3));
        result.setText(String.valueOf(max));
    }
    public static void main(String[] args) {
        new MaxOfThreeNumbers();
    }
}
```

Output:



```
Number 1:  2
Number 2:  3
Number 3:  1
Maximum:
3
```

Result:
Program is executed successfully and output is obtained.

Experiment No.21 **Date:9-5-2024**

File Manipulation

Aim:

Write a program to copy one file to another.

Algorithm:

```
1) Open a FileInputStream for "Sample.txt".
2) Open a FileOutputStream for "cp.txt".
3) Initialize an integer variable `c`.
4) Read from the input file character by character until the end of the file:
   a) Write the character to the output file.
   b) Print the character to the console.
5) Close the FileInputStream and FileOutputStream.
```

Source Code:

Readfile.java:

```
import java.io.*;

class Readfile {
   public static void main(String arg[]) throws IOException {
      FileInputStream f = new FileInputStream("Sample.txt");
      FileOutputStream f2 = new FileOutputStream("cp.txt");
      int c;
      while ((c = f.read()) != -1) {
         f2.write(c);
         System.out.print((char) c);
      }
      f.close();
      f2.close();
   }
}
```

Sample.txt:

```
Hello World!
```

Output:

```
Hello World!
```

Result:

Program is executed successfully and output is obtained.

Experiment No.22 **Date:9-5-2024**

File Manipulation II

Aim:

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

Algorithm:

1) Open a FileInputStream for "numbers.txt".
2) Open a FileOutputStream for "odd.txt".
3) Open a FileOutputStream for "even.txt".
4) Initialize an integer variable `c`.
5) Read from the input file character by character until the end of the file:
    a) If the ASCII value of the character is even write the character to "even.txt".
    b) Else write the character to "odd.txt".
    c) Print the character to the console.
6) Close the FileInputStream and both FileOutputStreams.
7) Open a FileInputStream for "odd.txt".
8) Read and print the contents of "odd.txt".
9) Close the FileInputStream for "odd.txt".
10) Open a FileInputStream for "even.txt".
11) Read and print the contents of "even.txt".
12) Close the FileInputStream for "even.txt".

Source Code:

File.java:
```
import java.io.*;

class File {
  public static void main(String arg[]) throws IOException {
    FileInputStream f = new FileInputStream("numbers.txt");
    FileOutputStream f2 = new FileOutputStream("odd.txt");
    FileOutputStream f3 = new FileOutputStream("even.txt");
    int c;

    // Reading the input file and categorizing characters based on ASCII values
    while ((c = f.read()) != -1) {
      if (c % 2 == 0) {
        f3.write(c); // Write to even.txt if ASCII value is even
      } else {
            f2.write(c); // Write to odd.txt if ASCII value is odd
        }
        System.out.print((char) c); // Print the character to console
    }

    // Closing the file streams
    f.close();
    f2.close();
    f3.close();

    // Displaying contents of odd.txt
    System.out.println("\nContents of odd.txt:");
    try (FileInputStream oddFile = new FileInputStream("odd.txt")) {
        while ((c = oddFile.read()) != -1) {
            System.out.print((char) c);
        }
    }

    // Displaying contents of even.txt
    System.out.println("\nContents of even.txt:");
    try (FileInputStream evenFile = new FileInputStream("even.txt")) {
        while ((c = evenFile.read()) != -1) {
            System.out.print((char) c);
        }
    }
  }
}
```

Numbers.txt:

```
0123456789
```

Output:

```
0123456789
Contents of odd.txt:
13579
Contents of even.txt:
02468
```

Result:

Program is executed successfully and output is obtained.