CO1

CO 1 Understand object-oriented concepts and design classes and objects to solve problems
Syllabus: Classes and Objects, Constructors, Method Overloading, Access Modifiers

1. 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.
2. Read 2 matrices from the console and perform matrix addition.
3. Add complex numbers
4. Read a matrix from the console and check whether it is symmetric or not.
5. 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.

1. **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.**

```
//MainProduct1.java
class Product
{
    int pcode;
    String pname;
    int pprice;
    Product(int code, String name, int price)
    {
        pcode = code;
        pname = name;
        pprice = price;
    }
    void display()
    {
        System.out.println("Product Code : "+pcode);
        System.out.println("Product Name : "+pname);
        System.out.println("Product Price : "+pprice);
    }
}
class MainProduct1
{
    public static void main(String args[])
    {
        Product ob1=new Product(123,"Pen",10);
        Product ob2=new Product(423,"Pencil",5);
        Product ob3=new Product(233,"Book",30);
         System.out.println("\n Product with lowest price");
         if (ob1.pprice < ob2.pprice)
             if (ob1.pprice < ob3.pprice)
                 ob1.display();
             else
```

```
                ob3.display();
        else if (ob3.pprice < ob2.pprice)
            ob3.display();
        else
            ob2.display();
    }
}
```

```
 /*E:\S2 JAVA LAB 2024\JavaRec>javac MainProduct1.javad
E:\S2 JAVA LAB 2024\JavaRec>java MainProduct1
Product with lowest price
Product Code : 423
Product Name : Pencil
Product Price : 5  */
```

2. Read 2 matrices from the console and perform matrix addition.

//Add2Matrix.java

```
import java.util.Scanner;
class Matrix
 /* The Matrix class is defined to represent a matrix. It has private member variables matrix, rows, and
cols to store the matrix elements, number of rows, and number of columns respectively.*/
{
    private int matrix[ ][ ];
    private int rows;
    private int cols;
    public Matrix (int rows, int cols)
    /*The constructor initializes the matrix with the specified number of rows and columns.   */
    {
        this.rows = rows;
        this.cols = cols;
        this.matrix = new int [rows][cols];
    }
    public void readMatrix(Scanner sc)
    /* reads the elements of the matrix from the user using a Scanner object.  */
    {
        System.out.println ("Enter elements of the matrix:");
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
                matrix [i][j] = sc.nextInt();
        }
    }
    public void displayMatrix( ) // prints the matrix to the console
    {
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
                System.out.print (matrix[i][j] + " ");
            System.out.println ( );
        }
    }
    public static Matrix addMatrices(Matrix a, Matrix b)
    /* static belongs to the class itself, not any specific instance of the class.
       It takes two Matrix objects as parameters (a and b) and returns a new Matrix object which
       is the result of adding the two input matrices. It creates a new Matrix object (result) with
       the same dimensions as the input matrices. It adds corresponding elements from both
       input matrices and stores the result in the result matrix.        */
    {
        Matrix result = new Matrix (a.rows, a.cols);
        for (int i = 0; i < a.rows; i++)
        {
            for (int j = 0; j < a.cols; j++)
                result.matrix[i][j] = a.matrix[i][j] + b.matrix[i][j];
        }
        return result;
    }
}
public class Add2Matrix
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of rows in first matrix:");
        int row1 = sc.nextInt();
        System.out.print("Enter number of columns in first matrix:");
        int col1 = sc.nextInt();
        System.out.print("Enter number of rows in second matrix:");
        int row2 = sc.nextInt();
        System.out.print("Enter number of columns in second matrix:");
        int col2 = sc.nextInt();
        if (row1 != row2 || col1 != col2)
        {
            System.out.println("Addition not possible");
            return;
        }
        Matrix Matrix1 = new Matrix(row1, col1);
        Matrix1.readMatrix (sc);

        Matrix Matrix2 = new Matrix(row2, col2);
        Matrix2.readMatrix (sc);

        System. out.println("First Matrix:");
        Matrix1.displayMatrix( );
        System.out.println("Second Matrix:");
        Matrix2.displayMatrix ( );
        Matrix resultMatrix = Matrix.addMatrices (Matrix1, Matrix2);
        System. out.println ("Matrix after addition:");
        resultMatrix.displayMatrix ( );
    }
}
/*E:\S2 JAVA LAB 2024\JavaRec>javac Add2Matrix.java
E:\S2 JAVA LAB 2024\JavaRec>java Add2Matrix
Enter number of rows in first matrix:2
Enter number of columns in first matrix:2
Enter number of rows in second matrix:2
Enter number of columns in second matrix:2
Enter elements of the matrix:
1
1
1
1
Enter elements of the matrix:
2
2
2
2
First Matrix:
1 1
1 1
Second Matrix:
2 2
2 2
Matrix after addition:
3 3
3 3                    */
```

3. Add complex numbers

```
import java.util.Scanner;
class Complex
{
   int real, imaginary;
   Complex (int real, int imaginary) //Parameterized constructor
   {
      this.real = real;
      this.imaginary = imaginary;
   }
   Complex addComplex (Complex ip1, Complex ip2)
           /* method addComplex, which takes two Complex objects (ip1 and ip2) as parameters and returns a
                    Complex object.   */
   {
      Complex temp = new Complex (0, 0);    // This line creates a new Complex object named temp with initial real and
                          imaginary parts set to 0. This object will store the result of adding the two input complex numbers. */
      temp.real = ip1.real + ip2.real;
      temp.imaginary = ip1.imaginary + ip2.imaginary;
      return temp;
   }
}
```

```
public class Complex3
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner (System.in);

        System.out.println ("Enter real & imaginary part of the 1st complex number:");
         int real1 = sc.nextInt();
         int imaginary1 = sc.nextInt();
        Complex ip1 = new Complex (real1, imaginary1);
        System.out.println ("1st complex number: " + ip1.real + "+i" + ip1.imaginary);

        System.out.println ("Enter real & imaginary part of the 2nd complex number:");
         int real2 = sc.nextInt( );
         int imaginary2 = sc.nextInt( );
        Complex ip2 = new Complex (real2, imaginary2);
        System.out.println ("2nd complex number: " + ip2.real + "+i" + ip2.imaginary);

       // creates a new Complex object named result with initial real & imaginary parts set to 0.
        Complex result = new Complex (0, 0);

        result = result.addComplex (ip1, ip2);
        System.out.println ("Sum: " + result.real + "+i" + result.imaginary);
    }
}
/*E:\S2 JAVA LAB 2024\JavaRec>javac Complex3.java
E:\S2 JAVA LAB 2024\JavaRec>java Complex3
Enter real & imaginary part of the 1st complex number:
1
2
1st complex number: 1+i2
Enter real & imaginary part of the 2nd complex number:
3
4
2nd complex number: 3+i4
The sum of complex numbers is: 4+i6 */
```

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

```
//Symmetric.java
import java.util.Scanner;
class Matrix
{
    private int rows;
    private int columns;
    private int mat [ ][ ];
    public Matrix (int rows, int columns)
    {
        this.rows = rows;
        this.columns = columns;
        this.mat = new int [rows][columns];
    }
    public void read (Scanner sc)
    {
        System.out.println("Enter the elements of the matrix:");
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < columns; j++)
                mat[i][j] = sc.nextInt();
        }
    }
    public void printMatrix()
    {
        System.out.println("Matrix:");
        for (int i = 0; i < rows; i++)
        {
             for (int j = 0; j < columns; j++)
                 System.out.print(mat[i][j] + " ");
             System.out.println();
        }
    }
    public boolean isSymmetric()
    {
        if (rows != columns)
            return false;       // Not square, so not symmetric
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < i; j++)
              if (mat[i][j] != mat[j][i])
                     return false;     // Not symmetric
        }
        return true; // Symmetric
    }

    public Matrix transpose()
    {
        Matrix transMatrix = new Matrix(columns, rows);
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < columns; j++)
                transMatrix.mat[j][i] = mat[i][j];
        }
        return transMatrix;
    }
}
```

```
public class Symmetric
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner (System.in);
        System.out.println("Enter the number of rows in the matrix:");
        int rows = sc.nextInt();
        System.out.println("Enter the number of columns in the matrix:");
        int columns = sc.nextInt();
        Matrix mat = new Matrix (rows, columns);
        mat.read(sc); /* This line calls the read method on the mat object, passing a Scanner object sc as
                         a parameter. Likely, the read method reads matrix elements from the user via the provided
                          Scanner object.            */
        mat.printMatrix ();
        Matrix transMatrix = mat.transpose();
        transMatrix.printMatrix();

        if (mat.isSymmetric())
            System.out.println("The matrix is symmetric.");
        else
            System.out.println("The matrix is not symmetric.");
    }
}
/*E:\S2 JAVA LAB 2024\JavaRec>javac Symmetric.java
E:\S2 JAVA LAB 2024\JavaRec>java Symmetric
Enter the number of rows in the matrix:
2
Enter the number of columns in the matrix:
2
Enter the elements of the matrix:
1
2
2
1
Matrix:
1 2
2 1
Matrix:
1 2
2 1
The matrix is symmetric.*/
```

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

```
public class CPU // outer class
{
   int price;
   CPU (int price)
   {
     this.price=price;
   }
   public void display()
   {
        System.out.println("\nCPU info");
        System.out.println("CPU price is "+price);
   }
   class Processor /* nested class named "Processor" inside the CPU class. This nested class
                     will contain properties and behaviors related to the processor of the CPU. */
   {
      int cores;
      String producer;
      Processor(int cores, String producer)
      {
         this.cores=cores;
         this.producer=producer;
      }
      void display()
       {
       System.out.println("\nProcessor info");
       System.out.println("No. of Cores = "+cores);
       System.out.println("Manufacturer = "+producer);
       }
   }
   static class RAM /*nested static class named "RAM" inside the CPU class. This nested class will
                      contain properties & behaviors related to the RAM of the CPU.*/
   {
      int memory;
      String producer;
      RAM (int memory, String producer )
      {
         this.memory=memory;
         this.producer=producer;
      }
      void display()
       {
       System.out.println("\nRAM info");
       System.out.println("Memory = "+memory+" GB");
       System.out.println("Manufacturer = "+producer);
       }
   }
```

```
    public static void main(String[] args)
    {
        CPU cpuobj = new CPU(30000);
        CPU.Processor probj = cpuobj.new Processor (8,"Samsung");   /*new Processor
              object named "probj" using the "new" keyword with the "Processor" class. Processor is a
                non-static inner class, it's instantiated using an instance of the outer class */
        CPU.RAM ramobj= new CPU.RAM(8,"Intel");     /* creates a new RAM object named
                "ramobj" . The syntax CPU.RAM is used because the RAM class is a nested static class.*/
        cpuobj.display();
        probj.display();
        ramobj.display();
    }
}
/*
E:\S2 JAVA LAB 2024\JavaRec>javac CPU.java
E:\S2 JAVA LAB 2024\JavaRec>java CPU
Processor info
CPU price is 30000
Processor info
No. of Cores = 8
Manufacturer = Samsung
RAM info
Memory = 8 GB
Manufacturer = Intel */
```