Table of Contents

CYCLE 1: Basic programs using datatypes, operators and control statement in java

Program 1.1	**Date: 01/01/2024**
Write a Java program to check whether a string is palindrome or not.	

PROGRAM :

```
import java.util.*;

public class P7_1_palindrome {
	public static void main(String args[]) {
		Scanner s = new Scanner(System.in);
		int i;
		boolean flag = true;
		try {
			System.out.println("\nEnter a string to check for palindrome : ");
			String str = s.nextLine();
			str = str.toLowerCase(); // coverts entered string to lowercase

			// Comparing one character at a time till middle of the string is reached
			for (i = 0; i < (str.length() / 2); i++) {
				if (str.charAt(i) != str.charAt(str.length() - i - 1)) {
					flag = false;
					break;
				}
			}
			if (flag) {
				System.out.println("\nstring " + str + " is Palindrome.");
			} else {
				System.out.println("\nstring " + str + " is Not Palindrome.");
			}

		} catch (Exception e) {
			System.out.println("\nError : " + e);
		}
	}
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P7_1_palindrome

Enter a string to check for palindrome :
malayalam

string malayalam is Palindrome.

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 1.2	Date: 02/01/2024

Write a Java program to multiply two matrices.

PROGRAM :

```
import java.util.*;

public class P7_2_matrix_multiply {
	public static void main(String args[]) {
		Scanner s = new Scanner(System.in);
		int row1, col1, row2, col2;
		int i, j;
		try {
			// MATRIX 1
			System.out.println("\nEnter the number of rows of First    matrix :");
			row1 = s.nextInt();
			System.out.println("\nEnter the number of columns of First    matrix :");
			col1 = s.nextInt();

			System.out.println("\nEnter the Elements of First    matrix : ");
			int[][] matrix1 = new int[row1][col1];
			for (i = 0; i < row1; i++) {
				for (j = 0; j < col1; j++) {
					matrix1[i][j] = s.nextInt();
				}
			}

			// MATRIX 2
			System.out.println("\nEnter the number of rows of Second    matrix :");
			row2 = s.nextInt();
			System.out.println("\nEnter the number of columns of Second    matrix :");
			col2 = s.nextInt();

			System.out.println("\nEnter the Elements of Second    matrix : ");
			int[][] matrix2 = new int[row2][col2];
			for (i = 0; i < row2; i++) {
				for (j = 0; j < col2; j++) {
					matrix2[i][j] = s.nextInt();
				}
			}

			// no.of columns of First matrix and no.of rows of Second matrix should be
same
			// for matrix multiplication posssible.
			if (col1 != row2) {
				System.out.println("\nMatrix multiplication is not possible !");
			} else {
				// RESULT MATRIX
				int[][] result_matrix = multiply_matrix(matrix1, matrix2);
```

```
                System.out.println("\nResult Matrix is --->");

                for (i = 0; i < row1; i++) {
                    for (j = 0; j < col2; j++) {
                        System.out.print(result_matrix[i][j] + "\t");
                    }
                    System.out.println("\n");
                }
            }

        } catch (Exception e) {
            System.out.println("\nError : " + e);
        } finally {
            s.close();
        }
    }

    public static int[][] multiply_matrix(int[][] matrix1, int[][] matrix2) {
        int row1, col1, col2;
        int i, j, k;
        row1 = matrix1.length;
        col2 = matrix2[0].length;
        col1 = matrix1[0].length;

        int[][] result = new int[row1][col2];

        // multiplying
        for (i = 0; i < row1; i++) {
            for (j = 0; j < col2; j++) {
                for (k = 0; k < col1; k++) {
                    result[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }
        return result;

    }
}
```

OUTPUT :

```
E:\MUHAMMAD ANSHAD P A\JAVA\JAVA LAB>java P7_2_matrix_multiply

Enter the number of rows of First  matrix :
2

Enter the number of columns of First  matrix :
2

Enter the Elements of First  matrix :
1
2
3
4

Enter the number of rows of Second  matrix :
2

Enter the number of columns of Second  matrix :
2

Enter the Elements of Second  matrix :
4
5
6
7

Result Matrix is --->
16      19

36      43
```

Program 1.3	Date: 04/01/2024
Write a Java program to find the transpose of a matrix.	

PROGRAM :

```
import java.util.Scanner;

public class P7_3_transpose_matrix {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Enter the number of rows of the matrix: ");
		int rows = scanner.nextInt();
		System.out.print("Enter the number of columns of the matrix: ");
		int cols = scanner.nextInt();

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

		// Input matrix elements
		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();
			}
		}

		System.out.println("Original Matrix:");
		printMatrix(matrix);

		// Transpose the matrix
		int[][] transposeMatrix = findTranspose(matrix);

		System.out.println("Transpose of the Matrix:");
		printMatrix(transposeMatrix);

		scanner.close();
	}

	// Function to find the transpose of a matrix
	public static int[][] findTranspose(int[][] matrix) {
		int rows = matrix.length;
		int cols = matrix[0].length;

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

		for (int i = 0; i < rows; i++) {
			for (int j = 0; j < cols; j++) {
				transpose[j][i] = matrix[i][j];
			}
		}
```

```
        return transpose;
    }

    // Function to print a matrix
    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int num : row) {
                System.out.print(num + "\t");
            }
            System.out.println();
        }
    }
}
```

OUTPUT :

```
PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>

Enter the number of rows of the matrix: 2
Enter the number of columns of the matrix: 2
Enter the elements of the matrix:
4
5
6
7
Original Matrix:
4       5
6       7
Transpose of the Matrix:
4       6
5       7
PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>
```

Program 1.4	Date: 05/01/2024

Write a Java program to find the second smallest element in an array.

PROGRAM :

```
import java.util.*;

public class P7_1_palindrome {
	public static void main(String args[]) {
		Scanner s = new Scanner(System.in);
		int i;
		boolean flag = true;
		try {
			System.out.println("\nEnter a string to check for palindrome : ");
			String str = s.nextLine();
			str = str.toLowerCase(); // coverts entered string to lowercase

			// Comparing one character at a time till middle of the string is reached
			for (i = 0; i < (str.length() / 2); i++) {
				if (str.charAt(i) != str.charAt(str.length() - i - 1)) {
					flag = false;
					break;
				}
			}
			if (flag) {
				System.out.println("\nstring " + str + " is Palindrome.");
			} else {
				System.out.println("\nstring " + str + " is Not Palindrome.");
			}

		} catch (Exception e) {
			System.out.println("\nError : " + e);
		}
	}
}
```

OUTPUT :

```
PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>

Enter the number of elements in the array: 5
Enter the elements of the array:
4
3
7
8
1
The second smallest element in the array is: 3
PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>
```

Program 1.5	Date: 06/01/2024
Write a Java program to check whether a number is prime or not.	

PROGRAM :

```
import java.util.Scanner;

public class P7_5_prime {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);

		System.out.print("Enter a number to check if it's prime: ");
		int number = scanner.nextInt();

		if (isPrime(number)) {
			System.out.println(number + " is a prime number.");
		} else {
			System.out.println(number + " is not a prime number.");
		}

		scanner.close();
	}

	public static boolean isPrime(int number) {
		if (number <= 1) {
			return false;
		}

		if (number <= 3) {
			return true;
		}

		if (number % 2 == 0 || number % 3 == 0) {
			return false;
		}

		for (int i = 5; i * i <= number; i += 6) {
			if (number % i == 0 || number % (i + 2) == 0) {
				return false;
			}
		}

		return true;
	}
}
```

OUTPUT :

```
PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>

Enter a number to check if it's prime: 7
7 is a prime number.
PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>
```

Program 1.6	Date: 06/01/2024

Write a java program to demonstrate Bitwise logical operators, left shift and right shift operators.

PROGRAM :

```
import java.util.*;

public class P7_6_bitwise {
	public static void main(String[] args) {
		Scanner s = new Scanner(System.in);
		try {
			System.out.println("\nEnter the first integer : ");//Eg:12 -> Binary: 1100
			int num1 = s.nextInt();
			System.out.println("\nEnter the second integer : ");//Eg:7 -> Binary: 0111
			int num2 = s.nextInt();
			// Bitwise AND operator (&)
			int resultAnd = num1 & num2; // Result: 4 (Binary: 0100)
			System.out.println("Bitwise AND of " + num1 + " and " + num2 + " is: " +
resultAnd);
			// Bitwise OR operator (|)
			int resultOr = num1 | num2; // Result: 15 (Binary: 1111)
			System.out.println("Bitwise OR of " + num1 + " and " + num2 + " is: " +
resultOr);

			// Bitwise XOR operator (^)
			int resultXor = num1 ^ num2; // Result: 11 (Binary: 1011)
			System.out.println("Bitwise XOR of " + num1 + " and " + num2 + " is: " +
resultXor);

			// Bitwise NOT operator (~)
			int resultNotNum1 = ~num1; // Result: -13 (Binary: 11111111 11111111
11111111 11110011)
			System.out.println("Bitwise NOT of " + num1 + " is: " + resultNotNum1);

			// Left shift operator (<<)
			int resultLeftShift = num1 << 2; // Result: 48 (Binary: 110000)
			System.out.println("Left shift of " + num1 + " by 2 is: " + resultLeftShift);

			// Right shift operator (>>)
			int resultRightShift = num2 >> 2; // Result: 1 (Binary: 0001)
			System.out.println("Right shift of " + num2 + " by 2 is: " + resultRightShift);

		} catch (Exception e) {
			System.out.println("\nError : " + e);
		}
		s.close();
	}
}
```

LEARN • SERVE • EXCEL
RAJAGIRI

OUTPUT :

```
PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>

Enter the first integer :
12

Enter the second integer :
7
Bitwise AND of 12 and 7 is: 4
Bitwise OR of 12 and 7 is: 15
Bitwise XOR of 12 and 7 is: 11
Bitwise NOT of 12 is: -13
Left shift of 12 by 2 is: 48
Right shift of 7 by 2 is: 1
PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>
```

Program 1.7	Date: 08/01/2024
Write a java program to find the roots of a quadratic equation.	

PROGRAM :

```
import java.util.Scanner;
public class P7_7_roots_of_quadratic {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);

		System.out.println("Enter the coefficients of the quadratic equation (ax^2 + bx + c
= 0):");
		System.out.print("Enter the coefficient a: ");
		double a = scanner.nextDouble();

		System.out.print("Enter the coefficient b: ");
		double b = scanner.nextDouble();

		System.out.print("Enter the coefficient c: ");
		double c = scanner.nextDouble();

		double discriminant = b * b - 4 * a * c;

		if (discriminant > 0) {
			double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
			double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
			System.out.println("Roots are real and different.");
			System.out.println("Root 1 = " + root1);
			System.out.println("Root 2 = " + root2);
		} else if (discriminant == 0) {
			double root = -b / (2 * a);
			System.out.println("Roots are real and equal.");
			System.out.println("Root = " + root);
		} else {
			double realPart = -b / (2 * a);
			double imaginaryPart = Math.sqrt(-discriminant) / (2 * a);
			System.out.println("Roots are complex and different.");
			System.out.println("Root 1 = " + realPart + " + " + imaginaryPart + "i");
			System.out.println("Root 2 = " + realPart + " - " + imaginaryPart + "i");
		}
		scanner.close();
	}
}
```

OUTPUT :

```
PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>

Enter the coefficients of the quadratic equation (ax^2 + bx + c =
0):
Enter the coefficient a: 1
Enter the coefficient b: -3
Enter the coefficient c: 2
Roots are real and different.
Root 1 = 2.0
Root 2 = 1.0

PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>
```

CYCLE 2: Object Oriented Concepts

Program 2.1	Date: 16/01/2024

Write a Java program to calculate the area of different shapes namely circle, rectangle, trapezoid and triangle. (Use the concepts of JAVA like *this* keyword, constructor overloading and method overloading)

PROGRAM :

```
import java.util.Scanner;

class AreaCalculator {
	// Circle
	public double calculateArea(double radius) {
		return Math.PI * radius * radius;
	}

	// Rectangle
	public double calculateArea(double length, double width) {
		return length * width;
	}

	// Trapezoid
	public double calculateArea(double base1, double base2, double height) {
		return (base1 + base2) * height / 2;
	}

	// Triangle
	public double calculateTriangleArea(double base, double height) {
		return 0.5 * base * height;
	}
}

public class P8_1_area_of_shapes {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		AreaCalculator areaCalculator = new AreaCalculator();
		int choice = 0;
		do {
			System.out.println("\nChoose a shape to calculate its area:");
			System.out.println("1. Circle");
			System.out.println("2. Rectangle");
			System.out.println("3. Trapezoid");
			System.out.println("4. Triangle");
			System.out.println("5. Exit.\nSelect any one : ");

			choice = scanner.nextInt();
```

```
                switch (choice) {
                    case 1:
                        System.out.println("Enter the radius of the circle:");
                        double radius = scanner.nextDouble();
                        System.out.println("Area of the circle: " +
areaCalculator.calculateArea(radius));
                        break;
                    case 2:
                        System.out.println("Enter the length and width of the rectangle:");
                        double length = scanner.nextDouble();
                        double width = scanner.nextDouble();
                        System.out.println("Area of the rectangle: " +
areaCalculator.calculateArea(length, width));
                        break;
                    case 3:
                        System.out.println("Enter the lengths of the two bases and the
height of the trapezoid:");
                        double base1 = scanner.nextDouble();
                        double base2 = scanner.nextDouble();
                        double heightTrapezoid = scanner.nextDouble();
                        System.out.println(
                                "Area of the trapezoid: " +
areaCalculator.calculateArea(base1, base2, heightTrapezoid));
                        break;
                    case 4:
                        System.out.println("Enter the base and height of the triangle:");
                        double baseTriangle = scanner.nextDouble();
                        double heightTriangle = scanner.nextDouble();
                        System.out.println("Area of the triangle: "
                                + areaCalculator.calculateTriangleArea(baseTriangle,
heightTriangle));
                        break;
                    case 5:
                        System.out.println("Exiting....");
                        break;
                        // System.exit(0);
                    default:
                        System.out.println("Invalid choice!");
                }
            } while (choice != 5);

            scanner.close();
        }
}
```

OUTPUT :

```
PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>

Choose a shape to calculate its area:
1. Circle
2. Rectangle
3. Trapezoid
4. Triangle
5. Exit.
Select any one :
1
Enter the radius of the circle:
3
Area of the circle: 28.274333882308138

Choose a shape to calculate its area:
1. Circle
2. Rectangle
3. Trapezoid
4. Triangle
5. Exit.
Select any one :
2
Enter the length and width of the rectangle:
5
8
Area of the rectangle: 40.0

Choose a shape to calculate its area:
1. Circle
2. Rectangle
3. Trapezoid
4. Triangle
5. Exit.
Select any one :
3
Enter the lengths of the two bases and the height of the trapezoid:
5
6
7
Area of the trapezoid: 38.5

Choose a shape to calculate its area:
1. Circle
2. Rectangle
3. Trapezoid
4. Triangle
5. Exit.
Select any one :
4
```

```
Enter the base and height of the triangle:
7
4
Area of the triangle: 14.0

Choose a shape to calculate its area:
1. Circle
2. Rectangle
3. Trapezoid
4. Triangle
5. Exit.
Select any one :
5
Exiting....
PS E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)>
```

Program 2.2	Date: 16/01/2024

Define a class called Rectangle with member variables length and width. Use appropriate member functions to calculate the perimeter and area of the rectangle. Define another member function *int sameArea(Rectangle)* that has one parameter of type Rectangle. *sameArea* returns 1 if the two Rectangles have the same area, and returns 0 if they don't. Use appropriate constructors to initialize the member variables(Use both default and parameterized constructor)

PROGRAM :

```
import java.util.Scanner;

class Rectangle {
	private double length;
	private double width;

	// Default constructor
	public Rectangle() {
		this.length = 0;
		this.width = 0;
	}

	// Parameterized constructor
	public Rectangle(double length, double width) {
		this.length = length;
		this.width = width;
	}

	// Getter methods
	public double getLength() {
		return length;
	}

	public double getWidth() {
		return width;
	}

	// Setter methods
	public void setLength(double length) {
		this.length = length;
	}

	public void setWidth(double width) {
		this.width = width;
	}

	// Method to calculate perimeter
```

```
    public double calculatePerimeter() {
        return 2 * (length + width);
    }

    // Method to calculate area
    public double calculateArea() {
        return length * width;
    }

    // Method to check if two rectangles have the same area
    public int sameArea(Rectangle otherRectangle) {
        if (this.calculateArea() == otherRectangle.calculateArea()) {
            return 1;
        } else {
            return 0;
        }
    }
}

public class P8_2_member_functions {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking input for first rectangle
        System.out.println("Enter length and width for rectangle 1:");
        double length1 = scanner.nextDouble();
        double width1 = scanner.nextDouble();
        Rectangle rectangle1 = new Rectangle(length1, width1);

        // Taking input for second rectangle
        System.out.println("Enter length and width for rectangle 2:");
        double length2 = scanner.nextDouble();
        double width2 = scanner.nextDouble();
        Rectangle rectangle2 = new Rectangle(length2, width2);

        // Calculating and printing perimeter and area of rectangle 1
        System.out.println("Perimeter of rectangle 1: " + rectangle1.calculatePerimeter());
        System.out.println("Area of rectangle 1: " + rectangle1.calculateArea());

        // Calculating and printing perimeter and area of rectangle 2
        System.out.println("Perimeter of rectangle 2: " + rectangle2.calculatePerimeter());
        System.out.println("Area of rectangle 2: " + rectangle2.calculateArea());

        // Checking if rectangle 1 and rectangle 2 have the same area
        if (rectangle1.sameArea(rectangle2) == 1) {
            System.out.println("Rectangle 1 and Rectangle 2 have the same area.");
        } else {
            System.out.println("Rectangle 1 and Rectangle 2 don't have the same area.");
        }
        scanner.close();
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P8_2_member_functions
Enter length and width for rectangle 1:
4
2
Enter length and width for rectangle 2:
4
2
Perimeter of rectangle 1: 12.0
Area of rectangle 1: 8.0
Perimeter of rectangle 2: 12.0
Area of rectangle 2: 8.0
Rectangle 1 and Rectangle 2 have the same area.

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 2.3	Date: 16/01/2024

Write a main function to create two rectangle objects and display its area and perimeter. Check whether the two Rectangles have the same area and print a message indicating the result. (Use the concept of *this* pointer too)

PROGRAM :

```
import java.util.Scanner;

class Rectangle {
	private double length;
	private double width;

	// Default constructor
	public Rectangle() {
		this.length = 0;
		this.width = 0;
	}

	// Parameterized constructor
	public Rectangle(double length, double width) {
		this.length = length;
		this.width = width;
	}

	// Getter methods
	public double getLength() {
		return length;
	}

	public double getWidth() {
		return width;
	}

	// Setter methods
	public void setLength(double length) {
		this.length = length;
	}

	public void setWidth(double width) {
		this.width = width;
	}

	// Method to calculate perimeter
	public double calculatePerimeter() {
		return 2 * (this.length + this.width); // using this pointer to refer to the instance variables
```

```
    }

    // Method to calculate area
    public double calculateArea() {
        return this.length * this.width; // using this pointer to refer to the instance variables
    }

    // Method to check if two rectangles have the same area
    public boolean sameArea(Rectangle otherRectangle) {
        return this.calculateArea() == otherRectangle.calculateArea();
    }
}

public class P8_3_two_rectangleobjects {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Creating rectangle objects
        Rectangle rectangle1 = new Rectangle();
        Rectangle rectangle2 = new Rectangle();

        // Taking input for rectangle 1
        System.out.println("Enter length and width for rectangle 1:");
        double length1 = scanner.nextDouble();
        double width1 = scanner.nextDouble();
        rectangle1.setLength(length1);
        rectangle1.setWidth(width1);

        // Taking input for rectangle 2
        System.out.println("Enter length and width for rectangle 2:");
        double length2 = scanner.nextDouble();
        double width2 = scanner.nextDouble();
        rectangle2.setLength(length2);
        rectangle2.setWidth(width2);

        // Displaying area and perimeter of rectangle 1
        System.out.println("Area of rectangle 1: " + rectangle1.calculateArea());
        System.out.println("Perimeter of rectangle 1: " + rectangle1.calculatePerimeter());

        // Displaying area and perimeter of rectangle 2
        System.out.println("Area of rectangle 2: " + rectangle2.calculateArea());
        System.out.println("Perimeter of rectangle 2: " + rectangle2.calculatePerimeter());

        // Checking if rectangle 1 and rectangle 2 have the same area
        if (rectangle1.sameArea(rectangle2)) {
            System.out.println("Rectangle 1 and Rectangle 2 have the same area.");
        } else {
            System.out.println("Rectangle 1 and Rectangle 2 don't have the same area.");
        }
        scanner.close();
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P8_3_two_rectangleobjects
Enter length and width for rectangle 1:
16
6
Enter length and width for rectangle 2:
16
6
Area of rectangle 1: 96.0
Perimeter of rectangle 1: 44.0
Area of rectangle 2: 96.0
Perimeter of rectangle 2: 44.0
Rectangle 1 and Rectangle 2 have the same area.

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 2.4	Date: 17/01/2024

Write the definition for a class called Complex that has floating point data members for storing real and imaginary parts. Define a function *Complex sum(Complex)* to add two complex numbers

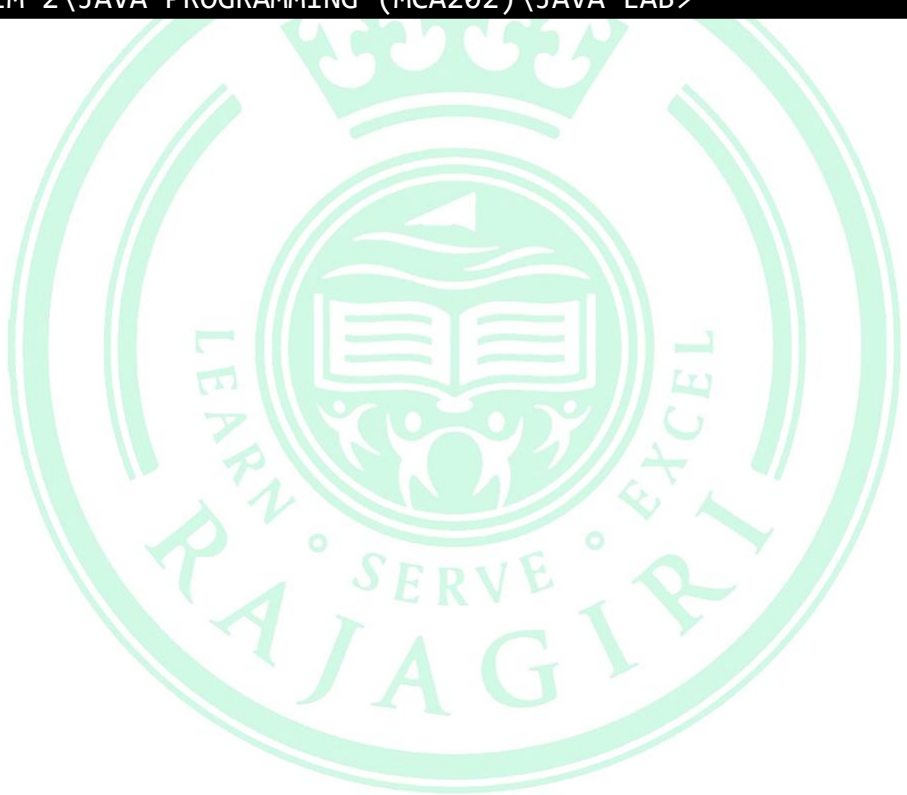
PROGRAM :

```
import java.util.Scanner;

class Complex {
	private double real;
	private double imaginary;

	// Default constructor
	public Complex() {
		this.real = 0;
		this.imaginary = 0;
	}

	// Parameterized constructor
	public Complex(double real, double imaginary) {
		this.real = real;
		this.imaginary = imaginary;
	}

	// Getter methods
	public double getReal() {
		return real;
	}

	public double getImaginary() {
		return imaginary;
	}

	// Setter methods
	public void setReal(double real) {
		this.real = real;
	}

	public void setImaginary(double imaginary) {
		this.imaginary = imaginary;
	}

	// Method to add two complex numbers
	public Complex sum(Complex other) {
		double realPart = this.real + other.getReal();
		double imaginaryPart = this.imaginary + other.getImaginary();
		return new Complex(realPart, imaginaryPart);
```

```
    }

    // Method to display complex number
    public void display() {
        System.out.println(this.real + " + " + this.imaginary + "i");
    }
}

public class P8_4_complex {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Creating three complex number objects
        Complex complex1 = new Complex();
        Complex complex2 = new Complex();
        Complex complex3;

        // Taking input for complex number 1
        System.out.println("Enter real and imaginary parts for complex number 1:");
        double real1 = scanner.nextDouble();
        double imaginary1 = scanner.nextDouble();
        complex1.setReal(real1);
        complex1.setImaginary(imaginary1);

        // Taking input for complex number 2
        System.out.println("Enter real and imaginary parts for complex number 2:");
        double real2 = scanner.nextDouble();
        double imaginary2 = scanner.nextDouble();
        complex2.setReal(real2);
        complex2.setImaginary(imaginary2);

        // Calculating sum and assigning it to complex number 3
        complex3 = complex1.sum(complex2);

        // Displaying all complex numbers
        System.out.println("Complex number 1:");
        complex1.display();
        System.out.println("Complex number 2:");
        complex2.display();
        System.out.println("Sum of complex number 1 and complex number 2:");
        complex3.display();

        scanner.close();
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P8_4_complex
Enter real and imaginary parts for complex number 1:
3.5
2.7
Enter real and imaginary parts for complex number 2:
1.2
-4.5
Complex number 1:
3.5 + 2.7i
Complex number 2:
1.2 + -4.5i
Sum of complex number 1 and complex number 2:
4.7 + -1.7999999999999998i

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```



Program 2.5	Date: 18/01/2024

Define a class called Time that has hours and minutes as integer. The class has the following member function: *Time sum(Time)* to sum two time object & return time a. Use the concept of constructor overloading to initialize the time

2.5.1 Write the definitions for each of the above member functions.

2.5.2 Write main function to create three time objects. Set the value in two objects and call sum() to calculate sum and assign it in third object. Display all time objects. (Use the concept of *this* pointer too)

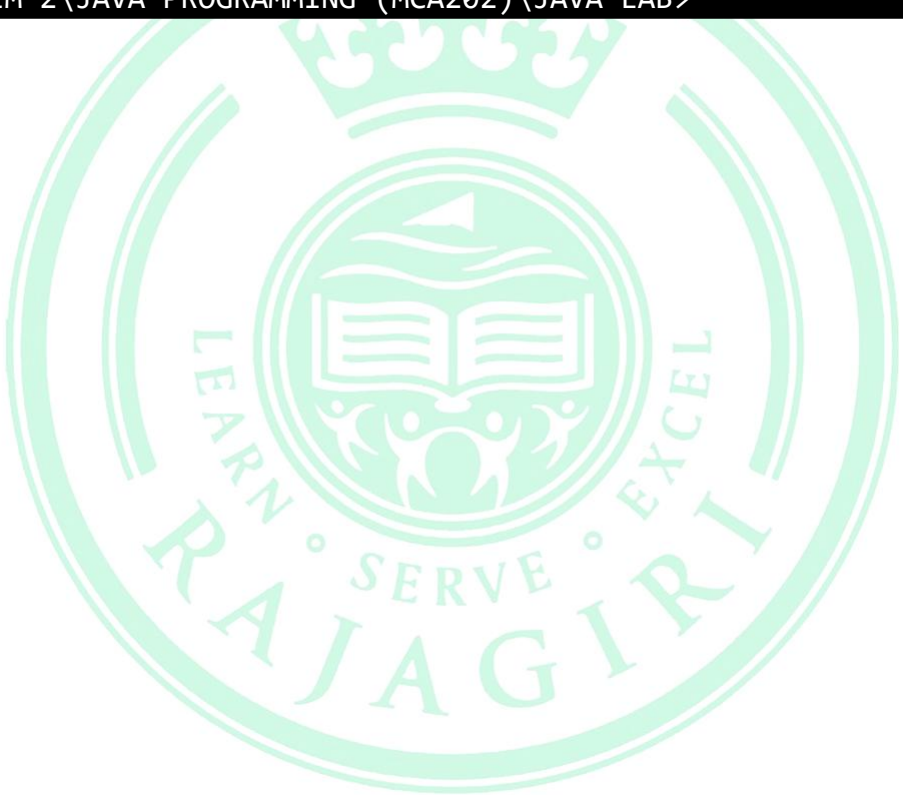
PROGRAM :

```
import java.util.Scanner;

class Time {
	private int hours;
	private int minutes;

	// Default constructor
	public Time() {
		this.hours = 0;
		this.minutes = 0;
	}

	// Parameterized constructor with hours and minutes
	public Time(int hours, int minutes) {
		this.hours = hours;
		this.minutes = minutes;
	}

	// Getter methods
	public int getHours() {
		return hours;
	}

	public int getMinutes() {
		return minutes;
	}

	// Setter methods
	public void setHours(int hours) {
		this.hours = hours;
	}

	public void setMinutes(int minutes) {
		this.minutes = minutes;
	}

	// Method to sum two Time objects
```

```
    public Time sum(Time other) {
        int totalHours = this.hours + other.getHours();
        int totalMinutes = this.minutes + other.getMinutes();

        // Adjust minutes if they exceed 60
        if (totalMinutes >= 60) {
            totalHours += totalMinutes / 60;
            totalMinutes %= 60;
        }

        return new Time(totalHours, totalMinutes);
    }
    // Method to display the time
    public void display() {
        System.out.println("Time: " + hours + " hours " + minutes + " minutes");
    }
}
public class P8_5_time {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Creating three time objects
        Time time1 = new Time();
        Time time2 = new Time();
        Time time3;

        // Taking input for time object 1
        System.out.println("Enter hours and minutes for time object 1:");
        int hours1 = scanner.nextInt();
        int minutes1 = scanner.nextInt();
        time1.setHours(hours1);
        time1.setMinutes(minutes1);

        // Taking input for time object 2
        System.out.println("Enter hours and minutes for time object 2:");
        int hours2 = scanner.nextInt();
        int minutes2 = scanner.nextInt();
        time2.setHours(hours2);
        time2.setMinutes(minutes2);
        // Calculating sum and assigning it to time object 3
        time3 = time1.sum(time2);
        // Displaying all time objects
        System.out.println("Time object 1:");
        time1.display();
        System.out.println("Time object 2:");
        time2.display();
        System.out.println("Sum of time object 1 and time object 2:");
        time3.display();

        scanner.close();
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P8_5_time
Enter hours and minutes for time object 1:
6
35
Enter hours and minutes for time object 2:
7
45
Time object 1:
Time: 6 hours 35 minutes
Time object 2:
Time: 7 hours 45 minutes
Sum of time object 1 and time object 2:
Time: 14 hours 20 minutes

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```



Program 2.6	Date: 19/01/2024

Create a class 'Account' with two overloaded constructors. The first constructor is used for initializing the name of account holder, the account number and the initial amount in the account. The second constructor is used for initializing the name of the account holder, the account number, the addresses, the type of account and the current balance. The Account class is having methods Deposit (), Withdraw (), and Get_Balance(). Make the necessary assumption for data members and return types of the methods. Create objects of Account class and use them.

PROGRAM :

```
import java.util.Scanner;

class Account {
	private String accountHolderName;
	private int accountNumber;
	private String address;
	private String accountType;
	private double balance;

	// First constructor
	public Account(String accountHolderName, int accountNumber, double initialAmount)
{
		this.accountHolderName = accountHolderName;
		this.accountNumber = accountNumber;
		this.balance = initialAmount;
	}

	// Second constructor
	public Account(String accountHolderName, int accountNumber, String address, String
accountType, double currentBalance) {
		this.accountHolderName = accountHolderName;
		this.accountNumber = accountNumber;
		this.address = address;
		this.accountType = accountType;
		this.balance = currentBalance;
	}

	// Method to deposit money
	public void deposit(double amount) {
		if (amount > 0) {
			balance += amount;
			System.out.println("Deposit successful. Current balance: " + balance);
		} else {
			System.out.println("Invalid deposit amount.");
		}
	}

	// Method to withdraw money
```

```
    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawal successful. Current balance: " + balance);
        } else {
            System.out.println("Insufficient funds or invalid withdrawal amount.");
        }
    }

    // Method to get current balance
    public double getBalance() {
        return balance;
    }
}

public class P8_6_account {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Account objects
        Account account1 = null;
        Account account2 = null;
        // Taking input for account 1
        System.out.println("Enter details for Account 1:");
        System.out.print("Account holder name: ");
        String name1 = scanner.nextLine();
        System.out.print("Account number: ");
        int number1 = scanner.nextInt();
        System.out.print("Initial balance: ");
        double initialBalance1 = scanner.nextDouble();
        account1 = new Account(name1, number1, initialBalance1);
        scanner.nextLine(); // Consume newline character
        // Taking input for account 2
        System.out.println("\nEnter details for Account 2:");
        System.out.print("Account holder name: ");
        String name2 = scanner.nextLine();
        System.out.print("Account number: ");
        int number2 = scanner.nextInt();
        scanner.nextLine(); // Consume newline character
        System.out.print("Address: ");
        String address2 = scanner.nextLine();
        System.out.print("Account type: ");
        String type2 = scanner.nextLine();
        System.out.print("Current balance: ");
        double currentBalance2 = scanner.nextDouble();
        account2 = new Account(name2, number2, address2, type2, currentBalance2);

        // Menu-driven loop
        boolean exit = false;
        while (!exit) {
            System.out.println("\nMenu:");
            System.out.println("1. Deposit to Account 1");
            System.out.println("2. Deposit to Account 2");
            System.out.println("3. Withdraw from Account 1");
```

```
                System.out.println("4. Withdraw from Account 2");
                System.out.println("5. Exit");
                System.out.print("Enter your choice: ");
                int choice = scanner.nextInt();
                switch (choice) {
                    case 1:
                        if (account1 != null) {
                            System.out.println("Enter deposit amount for account 1:");
                            double depositAmount1 = scanner.nextDouble();
                            account1.deposit(depositAmount1);
                        } else {
                            System.out.println("Account 1 not created yet.");
                        }
                        break;
                    case 2:
                        if (account2 != null) {
                            System.out.println("Enter deposit amount for account 2:");
                            double depositAmount2 = scanner.nextDouble();
                            account2.deposit(depositAmount2);
                        } else {
                            System.out.println("Account 2 not created yet.");
                        }
                        break;
                    case 3:
                        if (account1 != null) {
                            System.out.println("Enter withdrawal amount for account 1:");
                            double withdrawalAmount1 = scanner.nextDouble();
                            account1.withdraw(withdrawalAmount1);
                        } else {
                            System.out.println("Account 1 not created yet.");
                        }
                        break;
                    case 4:
                        if (account2 != null) {
                            System.out.println("Enter withdrawal amount for account 2:");
                            double withdrawalAmount2 = scanner.nextDouble();
                            account2.withdraw(withdrawalAmount2);
                        } else {
                            System.out.println("Account 2 not created yet.");
                        }
                        break;
                    case 5:
                        exit = true;
                        break;
                    default:
                        System.out.println("Invalid choice. Please enter a number between 1
and 5.");
                }
            }
            scanner.close();
        }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P8_6_account
Enter details for Account 1:
Account holder name: Anshad
Account number: 1001
Initial balance: 35000

Enter details for Account 2:
Account holder name: Nihal
Account number: 1005
Address: Nihal house,Kannur
Account type: Savings
Current balance: 55000

Menu:
1. Deposit to Account 1
2. Deposit to Account 2
3. Withdraw from Account 1
4. Withdraw from Account 2
5. Exit
Enter your choice: 1
Enter deposit amount for account 1:
25000
Deposit successful. Current balance: 60000.0

Menu:
1. Deposit to Account 1
2. Deposit to Account 2
3. Withdraw from Account 1
4. Withdraw from Account 2
5. Exit
Enter your choice: 2
Enter deposit amount for account 2:
7000
Deposit successful. Current balance: 62000.0

Menu:
1. Deposit to Account 1
2. Deposit to Account 2
3. Withdraw from Account 1
4. Withdraw from Account 2
5. Exit
Enter your choice: 3
Enter withdrawal amount for account 1:
6500
Withdrawal successful. Current balance: 53500.0

Menu:
1. Deposit to Account 1
2. Deposit to Account 2
```

```
3. Withdraw from Account 1
4. Withdraw from Account 2
5. Exit
Enter your choice: 4
Enter withdrawal amount for account 2:
100
Withdrawal successful. Current balance: 61900.0

Menu:
1. Deposit to Account 1
2. Deposit to Account 2
3. Withdraw from Account 1
4. Withdraw from Account 2
5. Exit
Enter your choice: 5

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

CYCLE 3: Inheritance, method overloading and overriding, Polymorphism

Program 3.1	Date: 20/01/2024

Write a Java program which creates a class named 'Employee' having the following members: Name, Age, Phone number, Address, Salary. It also has a method named 'print- Salary( )' which prints the salary of the Employee. Two classes 'Officer' and 'Manager' inherits the 'Employee' class. The 'Officer' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an officer and a manager by making an object of both of these classes and print the same.

PROGRAM :

```
import java.util.Scanner;

class Employee {
	String name;
	int age;
	String phoneNumber;
	String address;
	double salary;

	public Employee(String name, int age, String phoneNumber, String address, double salary) {
		this.name = name;
		this.age = age;
		this.phoneNumber = phoneNumber;
		this.address = address;
		this.salary = salary;
	}

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

class Officer extends Employee {
	String specialization;

	public Officer(String name, int age, String phoneNumber, String address, double salary, String specialization) {
		super(name, age, phoneNumber, address, salary);
		this.specialization = specialization;
	}
}

class Manager extends Employee {
```

```
    String department;

    public Manager(String name, int age, String phoneNumber, String address, double salary, String department) {
        super(name, age, phoneNumber, address, salary);
        this.department = department;
    }
}

public class P9_1_employee {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter details for Officer:");
        System.out.print("Name: ");
        String officerName = scanner.nextLine();
        System.out.print("Age: ");
        int officerAge = scanner.nextInt();
        scanner.nextLine(); // Consume newline character
        System.out.print("Phone Number: ");
        String officerPhoneNumber = scanner.nextLine();
        System.out.print("Address: ");
        String officerAddress = scanner.nextLine();
        System.out.print("Salary: ");
        double officerSalary = scanner.nextDouble();
        scanner.nextLine(); // Consume newline character
        System.out.print("Specialization: ");
        String officerSpecialization = scanner.nextLine();

        System.out.println("\nEnter details for Manager:");
        System.out.print("Name: ");
        String managerName = scanner.nextLine();
        System.out.print("Age: ");
        int managerAge = scanner.nextInt();
        scanner.nextLine(); // Consume newline character
        System.out.print("Phone Number: ");
        String managerPhoneNumber = scanner.nextLine();
        System.out.print("Address: ");
        String managerAddress = scanner.nextLine();
        System.out.print("Salary: ");
        double managerSalary = scanner.nextDouble();
        scanner.nextLine(); // Consume newline character
        System.out.print("Department: ");
        String managerDepartment = scanner.nextLine();

        Officer officer = new Officer(officerName, officerAge, officerPhoneNumber, officerAddress, officerSalary, officerSpecialization);
        Manager manager = new Manager(managerName, managerAge, managerPhoneNumber, managerAddress, managerSalary, managerDepartment);

        System.out.println("\nOfficer Details:");
        System.out.println("Name: " + officer.name);
        System.out.println("Age: " + officer.age);
```

```
            System.out.println("Phone Number: " + officer.phoneNumber);
            System.out.println("Address: " + officer.address);
            System.out.println("Specialization: " + officer.specialization);
            officer.printSalary();

            System.out.println("\nManager Details:");
            System.out.println("Name: " + manager.name);
            System.out.println("Age: " + manager.age);
            System.out.println("Phone Number: " + manager.phoneNumber);
            System.out.println("Address: " + manager.address);
            System.out.println("Department: " + manager.department);
            manager.printSalary();

            scanner.close();
        }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P9_1_employee
Enter details for Officer:
Name: Anshad
Age: 23
Phone Number: 8157847663
Address: Anshad House,wayanad
Salary: 45000
Specialization: IT

Enter details for Manager:
Name: Nihal
Age: 22
Phone Number: 9946552244
Address: NIhal manzil,kannur
Salary: 65000
Department: HR

Officer Details:
Name: Anshad
Age: 23
Phone Number: 8157847663
Address: Anshad House,wayanad
Specialization: IT
Salary: 45000.0

Manager Details:
Name: Nihal
Age: 22
Phone Number: 9946552244
Address: NIhal manzil,kannur
Department: HR
Salary: 65000.0

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 3.2	Date: 21/01/2024

Write two Java classes Employee and Engineer. Engineer should inherit from Employee class. Employee class to have two methods display() and calcSalary(). Write a program to display the engineer salary and to display from Employee class using a single object instantiation (i.e., only one object creation is allowed).

- **display() only prints the name of the class and does not return any value. Ex. " Name of class is Employee."**
- **calcSalary() in Employee displays "Salary of employee is 10000" and calcSalary() in Engineer displays "Salary of employee is 20000."**

PROGRAM :

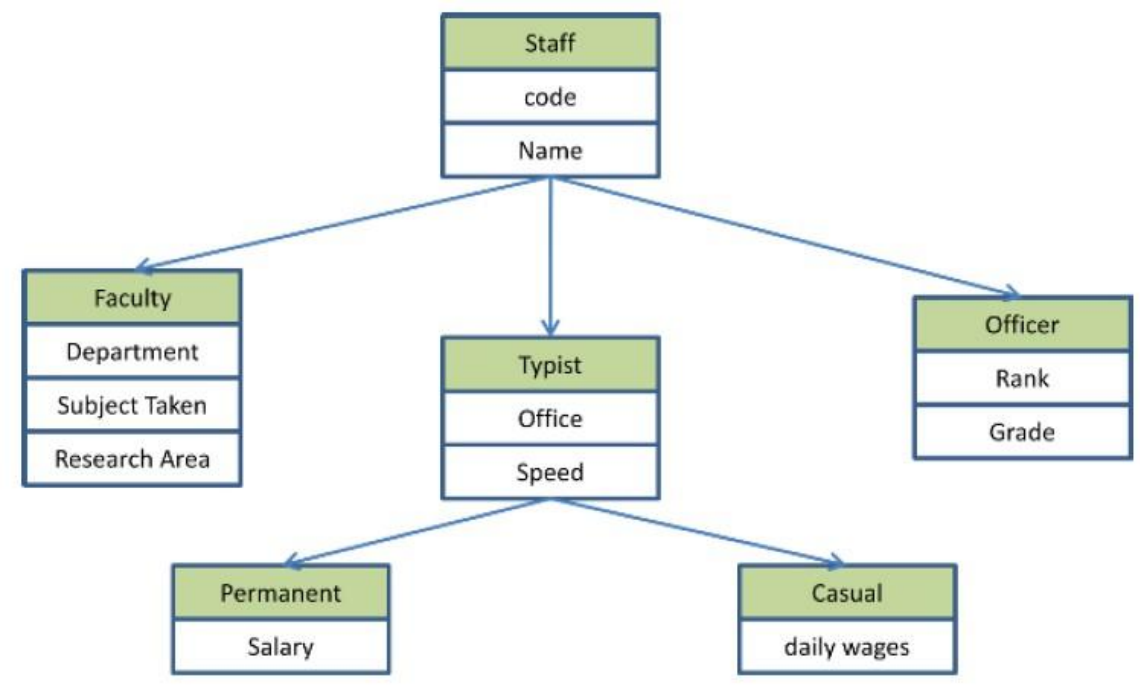
```
import java.util.Scanner;

class Employee {
	public void display() {
		System.out.println("Name of class is Employee.");
	}
	public void calcSalary() {
		System.out.println("Salary of employee is 10000.");
	}
}

class Engineer extends Employee {
	public void calcSalary() {
		System.out.println("Salary of employee is 20000.");
	}
}

public class P9_2_emp_engineer {
	public static void main(String[] args) {
		Employee emp = new Engineer(); // Polymorphism: Employee reference, Engineer
object
		emp.display(); // Calls display method of Employee class
		emp.calcSalary(); // Calls calcSalary method of Engineer class
	}
}
```

OUTPUT :

```
E:\MCA\SEM     2\JAVA     PROGRAMMING     (MCA202)\JAVA     LAB>java
P9_2_emp_engineer
Name of class is Employee.
Salary of employee is 20000.
```

Program 3.3	Date: 21/01/2024

Write a Java program to implement the following level of inheritance.



PROGRAM :

```
class Staff {
	String code;
	String name;

	public Staff(String code, String name) {
		this.code = code;
		this.name = name;
	}

	public void display() {
		System.out.println("Code: " + code);
		System.out.println("Name: " + name);
	}
}

class Faculty extends Staff {
	String department;
	String subjectTaken;
	String researchArea;

	public Faculty(String code, String name, String department, String subjectTaken, String
researchArea) {
		super(code, name);
		this.department = department;
```

```
        this.subjectTaken = subjectTaken;
        this.researchArea = researchArea;
    }

    public void display() {
        super.display();
        System.out.println("Department: " + department);
        System.out.println("Subject Taken: " + subjectTaken);
        System.out.println("Research Area: " + researchArea);
    }
}

class Typist extends Staff {
    String office;
    int speed;

    public Typist(String code, String name, String office, int speed) {
        super(code, name);
        this.office = office;
        this.speed = speed;
    }

    public void display() {
        super.display();
        System.out.println("Office: " + office);
        System.out.println("Speed: " + speed);
    }
}

class Officer extends Staff {
    String rank;
    String grade;

    public Officer(String code, String name, String rank, String grade) {
        super(code, name);
        this.rank = rank;
        this.grade = grade;
    }

    public void display() {
        super.display();
        System.out.println("Rank: " + rank);
        System.out.println("Grade: " + grade);
    }
}

class Permanent extends Typist {
    double salary;

    public Permanent(String code, String name, String office, int speed, double salary) {
        super(code, name, office, speed);
        this.salary = salary;
    }
```

```
    public void display() {
        super.display();
        System.out.println("Salary: " + salary);
    }
}

class Casual extends Typist {
    double dailyWages;

    public Casual(String code, String name, String office, int speed, double dailyWages) {
        super(code, name, office, speed);
        this.dailyWages = dailyWages;
    }

    public void display() {
        super.display();
        System.out.println("Daily Wages: " + dailyWages);
    }
}

public class P9_3_inheritance_levels {
    public static void main(String[] args) {
        Faculty faculty = new Faculty("F101", "Ansahd Muhammad", "Computer
Science", "Java Programming", "Machine Learning");
        Officer officer = new Officer("O201", "Majo", "Manager", "Grade A");
        Permanent permanent = new Permanent("T301", "Nihal", "Front Office", 50,
50000.0);
        Casual casual = new Casual("T401", "Hari", "Back Office", 40, 1000.0);

        System.out.println("Faculty Details:");
        faculty.display();

        System.out.println("\nOfficer Details:");
        officer.display();

        System.out.println("\nPermanent Typist Details:");
        permanent.display();

        System.out.println("\nCasual Typist Details:");
        casual.display();
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P9_3_inheritance_levels
Faculty Details:
Code: F101
Name: Ansahd Muhammad
Department: Computer Science
Subject Taken: Java Programming
Research Area: Machine Learning

Officer Details:
Code: O201
Name: Majo
Rank: Manager
Grade: Grade A

Permanent Typist Details:
Code: T301
Name: Nihal
Office: Front Office
Speed: 50
Salary: 50000.0

Casual Typist Details:
Code: T401
Name: Hari
Office: Back Office
Speed: 40
Daily Wages: 1000.0

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 3.4	Date: 22/01/2024

Write a java program to create an abstract class named Shape that contains an empty method named numberOfSides(). Provide three classes named Rectangle, Triangle and Hexagon such that each one of the classes extends the class Shape. Each one of the classes contains only the method numberOfSides( ) that shows the number of sides in the given geometrical structures.

PROGRAM :

```
import java.util.Scanner;

abstract class Shape {
	abstract void numberOfSides();
}

class Rectangle extends Shape {
	@Override
	void numberOfSides() {
		System.out.println("A rectangle has 4 sides.");
	}
}

class Triangle extends Shape {
	@Override
	void numberOfSides() {
		System.out.println("A triangle has 3 sides.");
	}
}

class Hexagon extends Shape {
	@Override
	void numberOfSides() {
		System.out.println("A hexagon has 6 sides.");
	}
}

public class P9_4_abstract_class {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);

		System.out.println("Enter the name of the shape (Rectangle, Triangle, or Hexagon):");
		String shapeName = scanner.nextLine().toLowerCase();

		Shape shape = null;
		switch (shapeName) {
```

```
                case "rectangle":
                        shape = new Rectangle();
                        break;
                case "triangle":
                        shape = new Triangle();
                        break;
                case "hexagon":
                        shape = new Hexagon();
                        break;
                default:
                        System.out.println("Invalid shape name!");
        }

        if (shape != null) {
                System.out.println("Details of " + shapeName + ":");
                shape.numberOfSides();
        }

        scanner.close();
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P9_4_abstract_class
Enter the name of the shape (Rectangle, Triangle, or Hexagon):
Rectangle
Details of rectangle:
A rectangle has 4 sides.

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P9_4_abstract_class
Enter the name of the shape (Rectangle, Triangle, or Hexagon):
Triangle
Details of triangle:
A triangle has 3 sides.

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P9_4_abstract_class
Enter the name of the shape (Rectangle, Triangle, or Hexagon):
Hexagon
Details of hexagon:
A hexagon has 6 sides.

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 3.5	Date: 22/01/2024

write a program to represent geometric shapes and some operations that can be performed on them. The idea here is that shapes in higher dimensions inherit data from lower dimensional shapes. For example a cube is a three dimensional square. A sphere is a three dimensional circle and a glome is a four dimensional circle. A cylinder is another kind of three dimensional circle. The circle, sphere, cylinder, and glome all share the attribute radius. The square and cube share the attribute side length. There are various ways to use inheritance to relate these shapes but please follow the inheritance described in the table below.
All shapes inherit getName() from the superclass Shape.

Class	Class Variable	Constructor	Extends	Implements
Shape	String name	Shape()		
Circle	double radius	Circle(double r, String n)	Shape	Area
Square	double side	Square(double s, String n)	Shape	Area
Cylinder	double height	Cylinder(double h, double r, String n)	Circle	Volume
Sphere	None	Sphere(double r, String n)	Circle	Volume
Cube	None	Cube(double s, String n)	Square	Volume
Glome	None	Glome(double r, String n)	Sphere	Volume

Note: the volume of a glome is $0.5(\pi^2)r^4$ where r is the radius.

Specification:

Your program will consist of the following classes: Shape, Circle, Square, Cube, Sphere, Cylinder,and Glome and two interfaces Area and Volume

Your classes may only have the class variable specified in the table below and the methods defined in the two interfaces Area and Volume. You will implement the methods specified in the Area and Volume interfaces and have them return the appropriate value for each shape. Class Shape will have a single public method called getName that returns a string.

PROGRAM :

```
import java.util.Scanner;

// Area interface
interface Area {
	double calculateArea();
}

// Volume interface
interface Volume {
	double calculateVolume();
}

// Shape class
class Shape {
	String name;

	public Shape() {
		this.name = "Generic Shape";
	}

	public String getName() {
		return name;
	}
}

// Circle class
class Circle extends Shape implements Area {
	double radius;

	public Circle(double r, String n) {
		this.radius = r;
		this.name = n;
	}

	@Override
	public double calculateArea() {
		return Math.PI * radius * radius;
	}
}

// Square class
class Square extends Shape implements Area {
	double side;

	public Square(double s, String n) {
		this.side = s;
		this.name = n;
	}
```

```
    @Override
    public double calculateArea() {
        return side * side;
    }
}

// Cylinder class
class Cylinder extends Circle implements Volume {
    double height;

    public Cylinder(double h, double r, String n) {
        super(r, n);
        this.height = h;
    }

    @Override
    public double calculateVolume() {
        return Math.PI * radius * radius * height;
    }
}

// Sphere class
class Sphere extends Circle implements Volume {
    public Sphere(double r, String n) {
        super(r, n);
    }

    @Override
    public double calculateVolume() {
        return (4.0 / 3.0) * Math.PI * Math.pow(radius, 3);
    }
}

// Cube class
class Cube extends Square implements Volume {
    public Cube(double s, String n) {
        super(s, n);
    }

    @Override
    public double calculateVolume() {
        return side * side * side;
    }
}

// Glome class
class Glome extends Sphere implements Volume {
    public Glome(double r, String n) {
        super(r, n);
    }
    @Override
    public double calculateVolume() {
        return 0.5 * Math.PI * Math.PI * radius * radius * radius * radius;
```

```
    }
}
public class P9_5_geometric_shapes {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input for creating Circle
        System.out.println("Enter radius for Circle:");
        double radius = scanner.nextDouble();
        Circle circle = new Circle(radius, "Circle");
        System.out.println(circle.getName() + " - Area: " + circle.calculateArea());

        // Input for creating Square
        System.out.println("Enter side length for Square:");
        double side = scanner.nextDouble();
        Square square = new Square(side, "Square");
        System.out.println(square.getName() + " - Area: " + square.calculateArea());

        // Input for creating Cylinder
        System.out.println("Enter radius for Cylinder:");
        radius = scanner.nextDouble();
        System.out.println("Enter height for Cylinder:");
        double height = scanner.nextDouble();
        Cylinder cylinder = new Cylinder(height, radius, "Cylinder");
        System.out.println(cylinder.getName() + " - Area: " + cylinder.calculateArea() + ",
Volume: " + cylinder.calculateVolume());
        // Input for creating Sphere
        System.out.println("Enter radius for Sphere:");
        radius = scanner.nextDouble();
        Sphere sphere = new Sphere(radius, "Sphere");
        System.out.println(sphere.getName() + " - Area: " + sphere.calculateArea() + ",
Volume: " + sphere.calculateVolume());

        // Input for creating Cube
        System.out.println("Enter side length for Cube:");
        side = scanner.nextDouble();
        Cube cube = new Cube(side, "Cube");
        System.out.println(cube.getName() + " - Area: " + cube.calculateArea() + ",
Volume: " + cube.calculateVolume());

        // Input for creating Glome
        System.out.println("Enter radius for Glome:");
        radius = scanner.nextDouble();
        Glome glome = new Glome(radius, "Glome");
        System.out.println(glome.getName() + " - Area: " + glome.calculateArea() + ",
Volume: " + glome.calculateVolume());

        scanner.close();
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P9_5_geometric_shapes
Enter radius for Circle:
3
Circle - Area: 28.274333882308138
Enter side length for Square:
3
Square - Area: 9.0
Enter radius for Cylinder:
5
Enter height for Cylinder:
10
Cylinder - Area: 78.53981633974483, Volume: 785.3981633974483
Enter radius for Sphere:
3
Sphere - Area: 28.274333882308138, Volume: 113.09733552923254
Enter side length for Cube:
4
Cube - Area: 16.0, Volume: 64.0
Enter radius for Glome:
3
Glome - Area: 28.274333882308138, Volume: 399.71897824411906

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 3.6	Date: 23/01/2024

Define an interface "Operations" which has method area(), volume(). Define a constant PI having value 3.14. Create class a Cylinder(with member variable height) which implements this interface. Create one object and calculate area and volume. Add Required Constructors.

PROGRAM :

```
import java.util.Scanner;

// Define interface Operations
interface Operations {
	double PI = 3.14; // constant

	double area(); // method to calculate area
	double volume(); // method to calculate volume
}

// Implementing class Cylinder
class Cylinder implements Operations {
	double height; // member variable

	// Constructor
	public Cylinder(double height) {
		this.height = height;
	}

	// Method to calculate area
	public double area() {
		return 2 * PI * height;
	}

	// Method to calculate volume
	public double volume() {
		return PI * height * height;
	}
}

public class P9_6_interface_operations {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);

		System.out.print("Enter the height of the cylinder: ");
		double height = scanner.nextDouble();

		Cylinder cylinder = new Cylinder(height); // Creating an object of Cylinder

		// Calculating area and volume
```

```
            double area = cylinder.area();
            double volume = cylinder.volume();

            // Displaying the results
            System.out.println("Area of the cylinder: " + area);
            System.out.println("Volume of the cylinder: " + volume);

            scanner.close();
        }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P9_6_interface_operations
Enter the height of the cylinder: 10
Area of the cylinder: 62.800000000000004
Volume of the cylinder: 314.0

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB
```

Program 3.7	Date: 23/01/2024

Write a program that illustrates interface inheritance. Interface P is extended by P1 and P2. Interface P12 inherits from both P1 and P2.Each interface declares one constant and one method. class Q implements P12.Instantiate Q and invoke each of its methods. Each method displays one of the constants.

PROGRAM :

```
import java.util.*;

interface P {
	void methodP(); // method declaration
}

interface P1 extends P {
	void methodP1(); // method declaration
}

interface P2 extends P {
	void methodP2(); // method declaration
}

interface P12 extends P1, P2 {
	// No additional constants or methods here
}

class Q implements P12 {
	private int constantP;
	private int constantP1;
	private int constantP2;

	// Constructor to receive constants
	public Q(int constantP, int constantP1, int constantP2) {
		this.constantP = constantP;
		this.constantP1 = constantP1;
		this.constantP2 = constantP2;
	}

	// Implementing method from P1 interface
	public void methodP1() {
		System.out.println("Constant from P1: " + constantP1);
	}

	public void methodP2() {
		System.out.println("Constant from P2: " + constantP2);
	}

	public void methodP() {
		System.out.println("Constant from P: " + constantP);
```

```
    }
}

public class P9_7_interface_inheritance {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter constant value for P: ");
        int constantP = scanner.nextInt();

        System.out.print("Enter constant value for P1: ");
        int constantP1 = scanner.nextInt();

        System.out.print("Enter constant value for P2: ");
        int constantP2 = scanner.nextInt();

        // Instantiating Q with constants passed to its constructor
        Q q = new Q(constantP, constantP1, constantP2);

        q.methodP();
        q.methodP1();
        q.methodP2();

        scanner.close();
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P9_7_interface_inheritance
Enter constant value for P: 5
Enter constant value for P1: 7
Enter constant value for P2: 8
Constant from P: 5
Constant from P1: 7
Constant from P2: 8

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Cycle 4: Multithreading

Program 4.1	Date: 25/01/2024

Write a Java program to create two threads: One for displaying all odd number between 1 and 100 and second thread for displaying all even numbers between 1 and 100. Create a multithreaded program by creating a subclass of Thread and then creating, initializing, and starting two Thread objects from your class. The threads will execute concurrently Main thread should wait until all the other thread terminates its execution(using join()).

PROGRAM :

```
class OddThread extends Thread {
	public void run() {
		System.out.println("Odd numbers between 1 and 100:");
		for (int i = 1; i <= 100; i += 2) {
			System.out.print(i + " ");
		}
		System.out.println();
	}
}
class EvenThread extends Thread {
	public void run() {
		System.out.println("Even numbers between 1 and 100:");
		for (int i = 2; i <= 100; i += 2) {
			System.out.print(i + " ");
		}
		System.out.println();
	}
}
public class P10_1_threads {
	public static void main(String[] args) {
		Thread oddThread = new OddThread();
		Thread evenThread = new EvenThread();
		oddThread.start();
		evenThread.start();
		try{
		oddThread.join();
		evenThread.join();
		}catch(InterruptedException e) {
			System.out.println("Main thread interrupted");
		}
		System.out.println("Main thread exiting.");
	}
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P10_1_threads
Even numbers between 1 and 100:
Odd numbers between 1 and 100:
1 3 5 2 4 7 9 6 11 13 8 10 15 12 17 14 19 16 21 23 18 25 27 20 29 22
31 24 33 26 35 28 37 30 39 41 32 43 34 45 36 47 49 38 51 53 40 55 42
57 59 44 61 63 46 65 67 48 69 50 71 52 73 54 75 56 77 58 79 60 81 62
83 64 85 87 89 91 93 66 95 68 97 70 72 74 99 76
78 80 82 84 86 88 90 92 94 96 98 100
Main thread exiting.

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P10_1_threads
Odd numbers between 1 and 100:
Even numbers between 1 and 100:
2 1 4 3 6 5 7 8 9 11 10 13 12 14 15 17 16 19 18 21 20 22 24 23 26 25
28 27 30 29 32 34 31 36 33 38 35 40 37 42 44 39 46 41 48 43 50 52 45
54 47 56 49 58 51 60 53 62 55 64 57 66 59 68 70 61 72 74 76 63 78 65
80 67 82 69 84 71 86 88 73 90 75 92 94 96 98 100 77
79 81 83 85 87 89 91 93 95 97 99
Main thread exiting.

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 4.2	Date: 25/01/2024

Write a Java program that set thread priorities and display the priority.

PROGRAM :

```
class MyThread extends Thread {
	public MyThread(String name) {
		super(name);
	}

	public void run() {
		System.out.println("Thread: " + getName() + ", Priority: " + getPriority());
	}
}

public class    P10_2_thread_priority{
	public static void main(String[] args) {
		MyThread thread1 = new MyThread("Thread 1");
		MyThread thread2 = new MyThread("Thread 2");
		MyThread thread3 = new MyThread("Thread 3");

		// Set priorities for threads
		thread1.setPriority(Thread.MIN_PRIORITY); // Minimum priority
		thread2.setPriority(Thread.NORM_PRIORITY); // Normal priority
		thread3.setPriority(Thread.MAX_PRIORITY); // Maximum priority

		// Start threads
		thread1.start();
		thread2.start();
		thread3.start();
	}
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P10_2_thread_priority
Thread: Thread 2, Priority: 5
Thread: Thread 3, Priority: 10
Thread: Thread 1, Priority: 1

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 4.3	Date: 25/01/2024

Write a java program that implements a multi-thread application that has three threads. The first thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number

PROGRAM :

```
import java.util.Random;

class RandomNumberGenerator extends Thread {
	public void run() {
		Random random = new Random();
		while (true) {
			int randomNumber = random.nextInt(100); // Generating random integer between 0 and 99
			System.out.println("Generated Number: " + randomNumber);
			try {
				Thread.sleep(1000); // Sleep for 1 second
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
			if (randomNumber % 2 == 0) {
				// If the random number is even, notify the square thread
				synchronized (SquareThread.lock) {
					SquareThread.number = randomNumber;
					SquareThread.lock.notify();
				}
			} else {
				// If the random number is odd, notify the cube thread
				synchronized (CubeThread.lock) {
					CubeThread.number = randomNumber;
					CubeThread.lock.notify();
				}
			}
		}
	}
}

class SquareThread extends Thread {
	public static final Object lock = new Object();
	public static int number;

	public void run() {
		while (true) {
```

```
                synchronized (lock) {
                    try {
                        lock.wait();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    int square = number * number;
                    System.out.println("Square of " + number + " is: " + square);
                }
            }
        }
}

class CubeThread extends Thread {
    public static final Object lock = new Object();
    public static int number;

    public void run() {
        while (true) {
            synchronized (lock) {
                try {
                    lock.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                int cube = number * number * number;
                System.out.println("Cube of " + number + " is: " + cube);
            }
        }
    }
}

public class P10_3_multithread {
    public static void main(String[] args) {
        RandomNumberGenerator randomNumberGenerator = new
RandomNumberGenerator();
        SquareThread squareThread = new SquareThread();
        CubeThread cubeThread = new CubeThread();

        // Start all threads
        randomNumberGenerator.start();
        squareThread.start();
        cubeThread.start();
    }
}
```

OUTPUT :

```
E:\MCA\SEM     2\JAVA     PROGRAMMING     (MCA202)\JAVA     LAB>java
P10_3_multithread
Generated Number: 77
Generated Number: 40
Cube of 77 is: 456533
Generated Number: 37
Square of 40 is: 1600
Cube of 37 is: 50653
Generated Number: 91
Generated Number: 60
Cube of 91 is: 753571
Generated Number: 79
Square of 60 is: 3600
Generated Number: 94
Cube of 79 is: 493039
Square of 94 is: 8836
Generated Number: 73
Generated Number: 95
Cube of 73 is: 389017
Cube of 95 is: 857375
```

Program 4.4	Date: 25/01/2024

Write a program to illustrate creation of threads using runnable interface. (start method start each of the newly created thread. Inside the run method there is sleep() for suspend the thread for 500 milliseconds). Main thread should wait until all the other thread terminates its execution (using join()).

PROGRAM :

```
import java.util.Scanner;

class MyRunnable implements Runnable {
	public void run() {
		try {
			System.out.println(Thread.currentThread().getName() + " is running.");
			Thread.sleep(500);
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
	}
}

public class P10_4_thread_runnable {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Enter the number of threads: ");
		int numThreads = scanner.nextInt();
		scanner.close();

		Thread[] threads = new Thread[numThreads];

		for (int i = 0; i < numThreads; i++) {
			threads[i] = new Thread(new MyRunnable());
			threads[i].start();
		}

		for (Thread thread : threads) {
```

```
                try {
                    thread.join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }

            System.out.println("All threads have terminated.");
        }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P10_4_thread_runnable
Enter the number of threads: 10
Thread-7 is running.
Thread-1 is running.
Thread-8 is running.
Thread-3 is running.
Thread-4 is running.
Thread-6 is running.
Thread-9 is running.
Thread-2 is running.
Thread-5 is running.
Thread-0 is running.
All threads have terminated.

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 4.5	Date: 25/01/2024

Write a java program showing a typical invocation of banking operations via multiple threads. Create three threads and 2 methods deposit and withdraw methods to add the amount to the account and withdraw an amount from the account respectively. As the threads concurrently run the method, avoid the unpredictable behavior. (Use synchronization).

PROGRAM :

```
import java.util.Scanner;

class BankAccount {
	private int balance;
	public BankAccount(int initialBalance) {
		balance = initialBalance;
	}

	// Synchronized method to deposit amount into the account
	public synchronized void deposit(int amount) {
		System.out.println(Thread.currentThread().getName() + " is depositing $" + amount);
		balance += amount;
		System.out.println("New balance after deposit by " + Thread.currentThread().getName() + ": $" + balance);
	}

	// Synchronized method to withdraw amount from the account
	public synchronized void withdraw(int amount) {
		System.out.println(Thread.currentThread().getName() + " is withdrawing $" + amount);
		if (balance >= amount) {
			balance -= amount;
			System.out.println("New balance after withdrawal by " + Thread.currentThread().getName() + ": $" + balance);
		} else {
			System.out.println("Insufficient balance for withdrawal by " + Thread.currentThread().getName());
		}
	}
}
```

```
public class P10_5_banking {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Enter initial balance: ");
		int initialBalance = scanner.nextInt();

		// Creating a bank account with initial balance entered by the user
		BankAccount account = new BankAccount(initialBalance);

		System.out.print("Enter deposit amount: ");
		int depositAmount = scanner.nextInt();
		System.out.print("Enter withdrawal amount: ");
		int withdrawalAmount = scanner.nextInt();
		scanner.close();

		// Creating three threads performing deposit and withdrawal operations
		Thread thread1 = new Thread(() -> {
			account.deposit(depositAmount);
		}, "Thread-1");

		Thread thread2 = new Thread(() -> {
			account.withdraw(withdrawalAmount);
		}, "Thread-2");

		Thread thread3 = new Thread(() -> {
			account.deposit(depositAmount);
		}, "Thread-3");

		// Starting all threads
		thread1.start();
		thread2.start();
		thread3.start();
	}
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P10_5_banking
Enter initial balance: 50000
Enter deposit amount: 25000
Enter withdrawal amount: 3000
Thread-1 is depositing $25000
New balance after deposit by Thread-1: $75000
Thread-3 is depositing $25000
New balance after deposit by Thread-3: $100000
Thread-2 is withdrawing $3000
New balance after withdrawal by Thread-2: $97000

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 5.1	Date: 27/01/2024

Write a Java Program to merge data from two files into a third file. (Handle all file related exceptions)

PROGRAM :

```
import java.io.*;

public class  11_1_merge {
      public static void main(String[] args) {
            try (BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in))) {
                  System.out.print("Enter the first input file name: ");
                  String inputFile1 = reader.readLine();

                  System.out.print("Enter the second input file name: ");
                  String inputFile2 = reader.readLine();

                  System.out.print("Enter the output file name: ");
                  String outputFile = reader.readLine();

                  mergeFiles(inputFile1, inputFile2, outputFile);
            } catch (IOException e) {
                  System.err.println("An error occurred while reading user input: "
+ e.getMessage());
                  e.printStackTrace();
            }
      }

      public static void mergeFiles(String inputFile1, String inputFile2, String
outputFile) {
            try (BufferedReader reader1 = new BufferedReader(new
FileReader(new File(inputFile1)));
                  BufferedReader reader2 = new BufferedReader(new
FileReader(new File(inputFile2)));
```

```
            BufferedWriter writer = new BufferedWriter(new
FileWriter(new File(outputFile)))) {

                String line;

                // Merge data from the first input file
                while ((line = reader1.readLine()) != null) {
                    writer.write(line);
                    writer.newLine();
                }

                // Merge data from the second input file
                while ((line = reader2.readLine()) != null) {
                    writer.write(line);
                    writer.newLine();
                }

                System.out.println("Files merged successfully.");
            } catch (IOException e) {
                System.err.println("An error occurred while merging files: " +
e.getMessage());
                e.printStackTrace();
            }
        }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P11_1_merge
Enter the first input file name: anshad1.txt
Enter the second input file name: anshad2.txt
Enter the output file name: out1.txt
Files merged successfully.

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 5.2	Date: 27/01/2024

Write a Java Program to perform file merge operation where merging should be done by line by line alternatively. (Handle all file related exceptions)

PROGRAM :

```
import java.io.*;

public class P11_2_linebyline_merge {
	public static void main(String[] args) {
		try (BufferedReader reader = new BufferedReader(new InputStreamReader(System.in))) {
			System.out.print("Enter the first input file name: ");
			String inputFile1 = reader.readLine();

			System.out.print("Enter the second input file name: ");
			String inputFile2 = reader.readLine();

			System.out.print("Enter the output file name: ");
			String outputFile = reader.readLine();

			mergeFiles(inputFile1, inputFile2, outputFile);
		} catch (IOException e) {
			System.err.println("An error occurred while reading user input: " + e.getMessage());
			e.printStackTrace();
		}
	}

	public static void mergeFiles(String inputFile1, String inputFile2, String outputFile) {
		try (BufferedReader reader1 = new BufferedReader(new FileReader(new File(inputFile1)));
			BufferedReader reader2 = new BufferedReader(new FileReader(new File(inputFile2)));
			BufferedWriter writer = new BufferedWriter(new
```

```
FileWriter(new File(outputFile))) {

                String line1,line2=null;

                // Merge data from the first input file
                while ((line1 = reader1.readLine()) != null||(line2 =
reader2.readLine()) != null) {
            if(line1!=null){
                    writer.write(line1);
                    writer.newLine();
            }
            if(line2!=null){
                    writer.write(line2);
                    writer.newLine();
            }
              }


                System.out.println("Files merged successfully.");
          } catch (IOException e) {
                System.err.println("An error occurred while merging files: " +
e.getMessage());
                e.printStackTrace();
          }
      }
}
```

OUTPUT :

```
java P11_2_linebyline_merge
Enter the first input file name: anshad1.txt
Enter the second input file name: out1.txt
Enter the output file name: newout.txt
Files merged successfully.

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 5.3	Date: 27/01/2024

Write a Java program that reads a set of real numbers from a file and displays the minimum,maximum, average, and range of the numbers in the file. The user should be able to enter the name of the input file from the keyboard.

PROGRAM :

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

public class P11_3_realnum {

	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Enter the name of the input file: ");
		String fileName = scanner.nextLine();

		try {
			File file = new File(fileName);
			Scanner fileScanner = new Scanner(file);

			double sum = 0;
			double min = Double.MAX_VALUE;
			double max = Double.MIN_VALUE;

			while (fileScanner.hasNextDouble()) {
				double num = fileScanner.nextDouble();
				sum += num;
				if (num < min) {
					min = num;
				}
				if (num > max) {
					max = num;
				}
			}
```

```
            fileScanner.close();

            int count = 0;
            double range = max - min;

            System.out.println("Minimum: " + min);
            System.out.println("Maximum: " + max);
            System.out.println("Average: " + (sum / count));
            System.out.println("Range: " + range);

        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + fileName);
        }
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P11_3_realnum
Enter the name of the input file: newout.txt
Minimum: 1.0
Maximum: 5.0
Average: Infinity
Range: 4.0

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 5.4	Date: 27/01/2024

Write a program that reads the contents of a file and creates an exact copy of the file, except that each line is numbered. For example, if the input file contains the following text:

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

then the output file should appear something like this:

1: Two roads diverged in a yellow wood,
2: And sorry I could not travel both
3: And be one traveler, long I stood
4: And looked down one as far as I could
5: To where it bent in the undergrowth;

PROGRAM :

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;

public class P11_4_copy {
	public static void main(String[] args) {
		try (BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in))) {
			System.out.print("Enter the input file name: ");
			String inputFileName = reader.readLine();

			System.out.print("Enter the output file name: ");
			String outputFileName = reader.readLine();
```

```
                createNumberedCopy(inputFileName, outputFileName);
            } catch (IOException e) {
                System.err.println("An error occurred while reading user input: "
+ e.getMessage());
                e.printStackTrace();
            }
        }

        public static void createNumberedCopy(String inputFileName, String
outputFileName) {
            try (BufferedReader reader = new BufferedReader(new
FileReader(inputFileName));
                 BufferedWriter writer = new BufferedWriter(new
FileWriter(outputFileName))) {

                String line;
                int lineNumber = 1;
                while ((line = reader.readLine()) != null) {
                    writer.write(lineNumber + ": " + line);
                    writer.newLine();
                    lineNumber++;
                }

                System.out.println("Numbered copy created successfully.");
            } catch (IOException e) {
                System.err.println("An error occurred while creating numbered
copy: " + e.getMessage());
                e.printStackTrace();
            }
        }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P11_4_copy
Enter the input file name: out1.txt
Enter the output file name: out1copy.txt
Numbered copy created successfully.
```

Program 5.5	Date: 27/01/2024

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

PROGRAM :

```
import java.util.Scanner;
import java.util.StringTokenizer;

public class P11_5_lineof_integer {

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

		// Prompt the user to enter a line of integers
		System.out.print("Enter a line of integers separated by spaces: ");
		String inputLine = scanner.nextLine();

		// StringTokenizer to tokenize the input line
		StringTokenizer tokenizer = new StringTokenizer(inputLine);
		int sum = 0;
		// Iterate through tokens and calculate sum
		while (tokenizer.hasMoreTokens()) {
			// Convert token to integer and add to sum
			String token = tokenizer.nextToken();
			int number = Integer.parseInt(token);
			System.out.println("Integer: " + number);
			sum += number;
		}
		// Display the sum of all integers
		System.out.println("Sum of all integers: " + sum);

		scanner.close();
	}
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P11_5_lineof_integer
Enter a line of integers separated by spaces:  5 6 7 8 9 10
Integer: 5
Integer: 6
Integer: 7
Integer: 8
Integer: 9
Integer: 10
Sum of all integers: 45

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 5.6	Date: 27/01/2024

Write a Java program that displays the number of characters, lines and words in a text file.

PROGRAM :

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class P11_6_no_of_char {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Enter the path of the text file: ");
		String filePath = scanner.nextLine();

		try {
			FileReader fileReader = new FileReader(filePath);
			BufferedReader bufferedReader = new BufferedReader(fileReader);

			int charCount = 0;
			int wordCount = 0;
			int lineCount = 0;

			String line;
			while ((line = bufferedReader.readLine()) != null) {
				lineCount++;
				String[] words = line.trim().split("\\s+");
				wordCount += words.length;
				for (String word : words) {
					charCount += word.length();
				}
			}

			System.out.println("Number of characters: " + charCount);
```

```
            System.out.println("Number of words: " + wordCount);
            System.out.println("Number of lines: " + lineCount);

            bufferedReader.close();
        } catch (IOException e) {
            System.out.println("Error reading the file: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P11_6_no_of_char
Enter the path of the text file: anshad1.txt
Number of characters: 16
Number of words: 5
Number of lines: 1

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 5.7	Date: 27/01/2024

Write a Java Program to merge data from two files into a third file. (Handle all file related exceptions)

PROGRAM :

```
import java.util.Scanner;

class AgeOutOfRangeException extends Exception {
	public AgeOutOfRangeException(String message) {
		super(message);
	}
}

public class P11_7_studentclass {
	private int rollNo;
	private String name;
	private int age;
	private String course;

	public P11_7_studentclass(int rollNo, String name, int age, String course)
throws AgeOutOfRangeException {
		if (age < 15 || age > 21) {
			throw new AgeOutOfRangeException("Age must be between 15
and 21");
		}
		this.rollNo = rollNo;
		this.name = name;
		this.age = age;
		this.course = course;
	}

	public void display() {
		System.out.println("Roll No: " + rollNo);
		System.out.println("Name: " + name);
		System.out.println("Age: " + age);
```

```
            System.out.println("Course: " + course);
        }

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

            System.out.print("Enter Roll No: ");
            int rollNo = scanner.nextInt();
            scanner.nextLine(); // Consume newline character

            System.out.print("Enter Name: ");
            String name = scanner.nextLine();

            System.out.print("Enter Age: ");
            int age = scanner.nextInt();

            System.out.print("Enter Course: ");
            String course = scanner.next();

            try {
                P11_7_studentclass student = new P11_7_studentclass(rollNo,
name, age, course);
                System.out.println("\nStudent Details:");
                student.display();
            } catch (AgeOutOfRangeException e) {
                System.out.println("Error: " + e.getMessage());
            } finally {
                scanner.close();
            }
        }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P11_7_studentclass
Enter Roll No: 36
Enter Name: Anshad Muhammad
Enter Age: 21
Enter Course: MCA

Student Details:
Roll No: 36
Name: Anshad Muhammad
Age: 21
Course: MCA

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 5.8	Date: 27/01/2024

Write a Java program to define a class salesman with the attributes name, salesman code, sales amount and commission(use user inputs). The Company calculates the commission of a salesman according to the following formula:

(i) 8% if sales &<=2000
(ii) 10% sales if sales>=2000 and but <=5000
(iii)12% if sales exceeds 5000

Create salesman objects and find the commission of sales. Generate and handle exceptions if sales amount is less than 0.

PROGRAM :

```
import java.util.Scanner;

class P11_8_salesman {
	private String name;
	private int salesmanCode;
	private double salesAmount;
	private double commission;

	public P11_8_salesman(String name, int salesmanCode, double salesAmount) {
		this.name = name;
		this.salesmanCode = salesmanCode;
		this.salesAmount = salesAmount;
		calculateCommission();
	}

	private void calculateCommission() {
		if (salesAmount < 2000) {
			commission = salesAmount * 0.08; // 8% commission
		} else if (salesAmount >= 2000 && salesAmount <= 5000) {
			commission = salesAmount * 0.10; // 10% commission
		} else {
			commission = salesAmount * 0.12; // 12% commission
```

```
        }
    }

    public void display() {
        System.out.println("Salesman Name: " + name);
        System.out.println("Salesman Code: " + salesmanCode);
        System.out.println("Sales Amount: " + salesAmount);
        System.out.println("Commission: " + commission);
    }

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

        System.out.print("Enter Salesman Name: ");
        String name = scanner.nextLine();

        System.out.print("Enter Salesman Code: ");
        int salesmanCode = scanner.nextInt();

        System.out.print("Enter Sales Amount: ");
        double salesAmount = scanner.nextDouble();

        P11_8_salesman salesman = new P11_8_salesman(name,
salesmanCode, salesAmount);
        System.out.println("\nSalesman Details:");
        salesman.display();

        scanner.close();
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P11_8_salesman
Enter Salesman Name: Anshad
Enter Salesman Code: 1001
Enter Sales Amount: 5256

Salesman Details:
Salesman Name: Anshad
Salesman Code: 1001
Sales Amount: 5256.0
Commission: 630.72

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Cycle 6: Networking

Program 6.1	Date: 02/03/2024
Download the content of file from the internet	

PROGRAM :

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

class P12_1_download
{
        public static void main(String args[]) throws Exception{
                int c;
                URL u = new URL
("https://cs2113f18.github.io/java/JavaCheatSheet.pdf");
                URLConnection uc= u.openConnection();
                System.out.println("Date: " + new Date(uc.getDate()));
                System.out.println("Content-type: " + uc.getContentType());
                System.out.println("Expires: " + uc.getExpiration());
                System.out.println("Last-modified:          "          +          new
Date(uc.getLastModified()));

                int len =uc.getContentLength();
                System.out.println("content-length: " +len);

                if(len>0){
                        FileOutputStream fout = new FileOutputStream("test.pdf");
                        System.out.println("-----content-----");
                        InputStream input =uc.getInputStream();
                        int i = 0;
                        while(((c = input.read()) !=-1) && i<len){
                                fout.write((char)c);
                                i++;
                        }
```

```
                input.close();
                fout.close();
            }
            else{
                System.out.println("No content Available");
            }
        }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P12_1_download
Date: Sun Mar 24 11:48:07 IST 2024
Content-type: application/pdf
Expires: 1711198206000
Last-modified: Wed Dec 12 13:31:53 IST 2018
content-length: 163431
-----content-----

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 6.2	Date: 02/03/2024
Make a public chatting program using TCP/IP	

PROGRAM :

Server.java

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

public class P12_2_server {
    public static void main(String args[]){
        ServerSocket ss;
        Socket as;
        DataInputStream sin;
        DataOutputStream sout;
        try {
            ss = new ServerSocket(1234);//listens for incoming connections from clients on this port
            System.out.println("\nServer started waiting for client....");
            as = ss.accept(); //wait for request from client
            System.out.println("\nClient Connected");
            sin = new DataInputStream(as.getInputStream());
            sout = new DataOutputStream(as.getOutputStream());
            Scanner s = new Scanner(System.in);

            String received;
            String toSend;

            while(true){
                received = sin.readUTF();
                System.out.println("\nClient Says : "+received);

                if(received.equals("quit")){
```

```
                    System.out.println("\nClient is Closing.....");
                    break;
                }
                System.out.print("\nServer : ");
                toSend = s.nextLine();
                sout.writeUTF(toSend);

                if(toSend.equals("quit")){
                    System.out.println("\nServer is closing....");
                    break;
                }

            }
            ss.close();
            as.close();
            s.close();

        } catch (Exception e) {
            System.out.println("\nError : "+e);
        }

    }
}
```

Client.java

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

public class P12_2_client {
    public static void main(String args[]){
        Socket as;
        DataInputStream sin;
        DataOutputStream sout;
        try {
            as = new Socket("localhost",1234);
            System.out.println("\nConnected to Server.");
            sin = new DataInputStream(as.getInputStream());
```

```
                sout = new DataOutputStream(as.getOutputStream());
                Scanner s = new Scanner(System.in);

                String received;
                String toSend;

                while(true){
                    System.out.print("\nClient : ");
                    toSend = s.nextLine();
                    sout.writeUTF(toSend);
                    if(toSend.equals("quit")){
                        System.out.println("\nClient is closing.....");
                        break;
                    }

                    received = sin.readUTF();
                    System.out.println("\nServer says : "+received);
                    if(received.equals("quit")){
                        System.out.println("\nServer is stopping.....");
                        break;
                    }
                }
                as.close();
                s.close();

            } catch (Exception e) {
                System.out.println("\nError : "+e);
            }
        }
    }
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P12_2_server

Server started waiting for client....

Client Connected

Client Says : Hi Server

Server : Hello Client

Client Says : Have a nice day

Server : Thanks

Client Says : quit

Client is Closing.....

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P12_2_client

Connected to Server.

Client : Hi Server

Server says : Hello Client

Client : Have a nice day

Server says : Thanks

Client : quit

Client is closing......

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
```

Program 6.3	Date: 02/03/2024
Make a peer to peer messaging program using UDP.	

PROGRAM :

P12_3_udpserver.java

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

public class P12_3_udpserver {
	public static void main(String args[]) {
		DatagramSocket ds = null;
		DatagramPacket dp = null, reply = null;
		Scanner scanner = new Scanner(System.in);
		try {
			ds = new DatagramSocket(1234);
			byte[] buffer = new byte[1000];
			dp = new DatagramPacket(buffer, buffer.length);

			System.out.println("\nWaiting for client.....");
			ds.receive(dp);

			System.out.println("\nFrom client : " + (new
String(dp.getData())).trim());
			System.out.println("\nClient PORT : " + dp.getPort());
			reply = new DatagramPacket("From Server OK".getBytes(),
"From Server OK".length(), dp.getAddress(),
					dp.getPort());

			ds.send(reply);

			// Receiving response from client
			buffer = new byte[1000];
			reply = new DatagramPacket(buffer, buffer.length);
			ds.receive(reply);
			System.out.println("\nFrom client : " + (new
String(reply.getData())).trim());

			// Sending reply to client
			System.out.print("\nEnter your reply: ");
			String serverMessage = scanner.nextLine();
```

```
                reply = new DatagramPacket(serverMessage.getBytes(),
serverMessage.length(), dp.getAddress(),
                        dp.getPort());
                ds.send(reply);
            } catch (SocketException e) {
                System.out.println("\nSocket : " + e.getMessage());
            } catch (IOException e) {
                System.out.println("\nIO : " + e.getMessage());
            } finally {
                if (ds != null) {
                    ds.close();
                }
                scanner.close();
            }
        }
}
```

P12_3_udpclient.java

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

public class    P12_3_udpclient {
        public static void main(String args[]) {
            DatagramSocket ds = null;
            DatagramPacket dp = null, reply = null;
            InetAddress shost = null;
            Scanner scanner = new Scanner(System.in);
            try {
                ds = new DatagramSocket();
                byte[] m = "Bye".getBytes();
                shost = InetAddress.getByName("localhost");
                dp = new DatagramPacket(m, 3, shost, 1234);
                ds.send(dp);

                // Receiving reply from server
                byte[] buffer = new byte[1000];
                reply = new DatagramPacket(buffer, buffer.length);
                ds.receive(reply);
                System.out.println("\nReply : " + new
String(reply.getData()).trim());

                // Sending response back to server
                System.out.print("\nEnter your message: ");
```

```
                String clientMessage = scanner.nextLine();
                m = clientMessage.getBytes();
                dp = new DatagramPacket(m, m.length, shost, reply.getPort());
                ds.send(dp);

            } catch (SocketException e) {
                System.out.println("\nSocket : " + e.getMessage());
            } catch (IOException e) {
                System.out.println("\nIO : " + e.getMessage());
            } finally {
                if (ds != null) {
                    ds.close();
                }
                scanner.close();
            }
        }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java P12_3_udpserver

Waiting for client.......

From client : Bye

Client PORT : 59639

From client : Hi

Enter your reply: Hello

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>\JAVA PROGRAMMING
(MCA202)\JAVA LAB>
```

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
P12_3_udpclient

Reply : From Server OK

Enter your message: Hi

E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>CA202)\JAVA LAB>
```

Cycle 7: Database PRogramming

Program 7.1	Date: 13/03/2024

Construct the following tables:

Department (don(Primary), dname, dloc)

Emp (eno(Primary), ename, esal ,don(Foreign))

PROGRAM :

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateTableExample {
	public static void main(String[] args) {
		String url = "jdbc:oracle:thin:@localhost:1521:orcl";
		String username = "mca";
		String password = "mca";

		try {
			// Load the Oracle JDBC driver
			Class.forName("oracle.jdbc.driver.OracleDriver");

			// Establish connection
			Connection con = DriverManager.getConnection(url, username,
password);

			// Create tables
			Statement stmt = con.createStatement();

			// Create Department table
			String createDeptTableQuery = "CREATE TABLE Department
(" +
                                                "don INT PRIMARY KEY,
" +
                                                "dname VARCHAR(255), "
```

```
+
                                                    "dloc VARCHAR(255))";
                stmt.executeUpdate(createDeptTableQuery);

                // Create Emp table
                String createEmpTableQuery = "CREATE TABLE Emp (" +
                                                    "eno INT PRIMARY KEY, "
+
                                                    "ename VARCHAR(255), " +
                                                    "esal DECIMAL(10, 2), " +
                                                    "don INT, " +
                                                    "FOREIGN KEY (don)
REFERENCES Department(don))";
                stmt.executeUpdate(createEmpTableQuery);

                System.out.println("Tables created successfully.");

                // Close connection
                stmt.close();
                con.close();
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            }
        }
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
java CreateTableExample

Tables created successfully.
```

Program 7.2	Date: 13/03/2024

Write a program for displaying information in the following order from the above tables:

eno	ename	esal	dnam	dloc
101	Chetan	10,000	Civil	Kochi
102	Amish	20,000	Accounts	Delhi

PROGRAM :

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class DisplayInfoExample {
	public static void main(String[] args) {
		String url = "jdbc:oracle:thin:@localhost:1521:orcl";
		String username = "mca";
		String password = "mca";

		try {
			// Load the Oracle JDBC driver
			Class.forName("oracle.jdbc.driver.OracleDriver");

			// Establish connection
			Connection con = DriverManager.getConnection(url, username,
password);

			// Prepare SQL query
			String query = "SELECT e.eno, e.ename, e.esal, d.dname, d.dloc
" +
					"FROM Emp e " +
					"JOIN Department d ON e.don = d.don " +
					"WHERE e.eno IN (101, 102)";

			// Create prepared statement
```

```
                PreparedStatement pstmt = con.prepareStatement(query);

                // Execute query
                ResultSet rs = pstmt.executeQuery();

                // Display results
                System.out.println("eno\t ename\t    esal\t      dnam\t    dloc");
                while (rs.next()) {
                        int eno = rs.getInt("eno");
                        String ename = rs.getString("ename");
                        double esal = rs.getDouble("esal");
                        String dname = rs.getString("dname");
                        String dloc = rs.getString("dloc");
                        System.out.println(eno + "\t " + ename + "\t " + esal + "\t "
+ dname + "\t " + dloc);
                }

                // Close resources
                rs.close();
                pstmt.close();
                con.close();
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            }
        }
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>java
DisplayInfoExample

Eno     ename      esal         dnam        dloc
101     Anshad     10000.0     Civil       Kochi
102     Nihal      20000.0     Accounts    Delhi
```

Program 7.3	Date: 12/03/2024

Program to implement database connectivity using object oriented concepts.

PROGRAM :

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class DatabaseConnector {
	private Connection connection;
	private String url;
	private String username;
	private String password;

	// Constructor to initialize database connection parameters
	public DatabaseConnector(String url, String username, String password) {
		this.url = url;
		this.username = username;
		this.password = password;
	}

	// Method to establish database connection
	public void connect() throws SQLException {
		connection = DriverManager.getConnection(url, username,
password);
		System.out.println("Connected to database.");
	}

	// Method to close database connection
	public void disconnect() throws SQLException {
		if (connection != null && !connection.isClosed()) {
			connection.close();
			System.out.println("Disconnected from database.");
		}
	}
```

```
    // Method to execute a SQL query and return the ResultSet
    public ResultSet executeQuery(String query) throws SQLException {
        PreparedStatement statement = connection.prepareStatement(query);
        return statement.executeQuery();
    }

    // Main method to demonstrate usage
    public static void main(String[] args) {
        String url = "jdbc:oracle:thin:@localhost:1521:orcl";
        String username = "mca";
        String password = "mca";

        DatabaseConnector connector = new DatabaseConnector(url, username, password);

        try {
            connector.connect();
            ResultSet resultSet = connector.executeQuery("SELECT * FROM Emp");
            while (resultSet.next()) {
                // Process rows from the result set
                int eno = resultSet.getInt("eno");
                String ename = resultSet.getString("ename");
                double esal = resultSet.getDouble("esal");
                System.out.println("Employee: " + eno + ", " + ename + ", " + esal);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                connector.disconnect();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
java DatabaseConnector

Connected to database.
Employee : 101, Anshad , 10000.0
Employee : 102, Nihal  ,  20000.0
Employee : 103, Majo   , 15000.0
Employee : 104, Hari   , 18000.0
Employee : 105, Jibin  , 22000.0
Disconnected from database.
```

Program 7.4	Date: 12/03/2024

Write a JDBC program with Parametrized queries to update a given record (Rani's salary to 15,000) in the Emp table

PROGRAM :

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class UpdateRecordExample {
	public static void main(String[] args) {
		String url = "jdbc:oracle:thin:@localhost:1521:orcl";
		String username = "mca";
		String password = "mca";

		// Rani's new salary
		double newSalary = 15000;

		try {
			// Establish connection
			Connection con = DriverManager.getConnection(url, username, password);

			// Define the SQL query with parameters
			String updateQuery = "UPDATE Emp SET esal = ? WHERE ename = ?";

			// Create prepared statement
			PreparedStatement pstmt = con.prepareStatement(updateQuery);

			// Set parameters
			pstmt.setDouble(1, newSalary); // set the new salary
			pstmt.setString(2, "Rani");	// specify the employee name

			// Execute the update operation
```

```
            int rowsUpdated = pstmt.executeUpdate();

            if (rowsUpdated > 0) {
                System.out.println("Record updated successfully.");
            } else {
                System.out.println("No record updated.");
            }

            // Close resources
            pstmt.close();
            con.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
Java UpdateRecordExample

Record updated successfully.
```

Program 7.5	Date: 12/03/2024

Write a JDBC program with Parametrized queries to list the records of Emp table which has records whose names start with the alphabet “R”.

PROGRAM :

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class ListRecordsExample {
	public static void main(String[] args) {
		String url = "jdbc:oracle:thin:@localhost:1521:orcl";
		String username = "mca";
		String password = "mca";

		try {
			// Establish connection
			Connection con = DriverManager.getConnection(url, username, password);

			// Define the SQL query with parameter
			String selectQuery = "SELECT * FROM Emp WHERE ename LIKE ?";

			// Create prepared statement
			PreparedStatement pstmt = con.prepareStatement(selectQuery);

			// Set parameter value
			pstmt.setString(1, "R%"); // The "%" acts as a wildcard for any characters after "R"

			// Execute query
			ResultSet rs = pstmt.executeQuery();

			// Display results
```

```
                System.out.println("eno\t ename\t    esal\t      don");
                while (rs.next()) {
                        int eno = rs.getInt("eno");
                        String ename = rs.getString("ename");
                        double esal = rs.getDouble("esal");
                        int don = rs.getInt("don");
                        System.out.println(eno + "\t " + ename + "\t " + esal + "\t "
+ don);
                }

                // Close resources
                rs.close();
                pstmt.close();
                con.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
Java ListRecordsExample

eno     ename     esal     dno
103     Anshad    15000    2
```

Program 7.6	Date: 12/03/2024

Write a JDBC program with PreparedStatement to delete the records of Emp table which has records whose salary is less than 10,000.

PROGRAM :

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class DeleteRecordsExample {
	public static void main(String[] args) {
		String url = "jdbc:oracle:thin:@localhost:1521:orcl";
		String username = "mca";
		String password = "mca";

		try {
			// Establish connection
			Connection con = DriverManager.getConnection(url, username, password);

			// Define the SQL query with parameter
			String deleteQuery = "DELETE FROM Emp WHERE esal < ?";

			// Create prepared statement
			PreparedStatement pstmt = con.prepareStatement(deleteQuery);

			// Set parameter value
			pstmt.setDouble(1, 10000); // Deleting records with salary less than 10,000

			// Execute the delete operation
			int rowsDeleted = pstmt.executeUpdate();

			System.out.println(rowsDeleted + " records deleted successfully.");

			// Close resources
```

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

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
Java DeleteRecordsExample

0 records deleted successfully.
```

Program 7.7	Date: 12/03/2024

Implement a JDBC program which uses a Stored Procedure to insert records into Department table.

PROGRAM :

PROCEDURE:

```
/*CREATE OR REPLACE PROCEDURE insert_department(
	dno_param IN INT,
	dname_param IN VARCHAR2,
	dloc_param IN VARCHAR2
)
AS
BEGIN
	INSERT INTO Department(dno, dname, dloc) VALUES(dno_param, dname_param, dloc_param);
	COMMIT;
END;
/*/
```

SOURCE CODE:

```
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class InsertDepartmentExample {
	public static void main(String[] args) {
		String url = "jdbc:oracle:thin:@localhost:1521:orcl";
		String username = "mca";
		String password = "mca";

		try {
			// Establish connection
			Connection con = DriverManager.getConnection(url, username, password);
```

```
            // Define the SQL call to stored procedure
            String sql = "{ call insert_department(?, ?, ?) }";

            // Create CallableStatement
            CallableStatement cstmt = con.prepareCall(sql);

            // Set parameter values
            cstmt.setInt(1, 1);                 // dno
            cstmt.setString(2, "IT");       // dname
            cstmt.setString(3, "New York"); // dloc

            // Execute stored procedure
            cstmt.execute();

            System.out.println("Record inserted successfully.");

            // Close resources
            cstmt.close();
            con.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
Java InsertDepartmentExample

Record inserted successfully.
```

Program 7.8	Date: 12/03/2024

Use Callable statement to implement a Stored Procedure to display the Ename and Salary of all employees.

PROGRAM :

PROCEDURE:

```
/*CREATE OR REPLACE PROCEDURE display_employees_info
AS
BEGIN
	FOR emp_rec IN (SELECT ename, esal FROM Emp) LOOP
		DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_rec.ename ||
', Salary: ' || emp_rec.esal);
	END LOOP;
END;
/*/
```

SOURCE CODE:

```
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DisplayEmployeesInfo {
	public static void main(String[] args) {
		String url = "jdbc:oracle:thin:@localhost:1521:orcl";
		String username = "mca";
		String password = "mca";

		try {
			// Establish connection
			Connection con = DriverManager.getConnection(url, username,
password);

			// Define the SQL call to stored procedure
			String sql = "{ call display_employees_info }";
```

```
            // Create CallableStatement
            CallableStatement cstmt = con.prepareCall(sql);

            // Execute stored procedure
            cstmt.execute();

            // Close resources
            cstmt.close();
            con.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
Java DisplayEmployeesInfo

Stored procedure executed successfully.
```

Program 7.9	Date: 12/03/2024

Write a JDBC program to implement Transaction Management in the Department table.

PROGRAM :

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class TransactionExample {
	public static void main(String[] args) {
		String url = "jdbc:oracle:thin:@localhost:1521:orcl";
		String username = "mca";
		String password = "mca";

		Connection con = null;
		Statement stmt = null;

		try {
			// Establish connection
			con = DriverManager.getConnection(url, username, password);
			con.setAutoCommit(false); // Disable auto-commit mode

			// Create a Statement object
			stmt = con.createStatement();

			// Perform multiple database operations within the transaction
			stmt.executeUpdate("INSERT INTO Department VALUES
(101, 'IT', 'New York')");
			stmt.executeUpdate("INSERT INTO Department VALUES
(102, 'HR', 'London')");

			// Commit the transaction
			con.commit();
			System.out.println("Transaction committed successfully.");
		} catch (SQLException e) {
```

```
                try {
                    // Rollback the transaction if any exception occurs
                    if (con != null) {
                        con.rollback();
                        System.out.println("Transaction rolled back
successfully.");
                    }
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
                e.printStackTrace();
            } finally {
                try {
                    // Close resources
                    if (stmt != null) {
                        stmt.close();
                    }
                    if (con != null) {
                        con.close();
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
Java TransactionExample

Transaction committed successfully.
```

Program 7.10	Date: 12/03/2024

Write a JDBC program to depict the usage of SQLException Class and SQLWarning Class

PROGRAM :

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class SQLExceptionExample {
	public static void main(String[] args) {
		String url = "jdbc:oracle:thin:@localhost:1521:orcl";
		String username = "mca";
		String password = "mca";

		try {
			// Establish connection
			Connection con = DriverManager.getConnection(url, username,
password);

			// Create a Statement object
			Statement stmt = con.createStatement();

			// Execute a query that may cause SQLException
			ResultSet rs = stmt.executeQuery("SELECT * FROM
NonExistentTable");

			// Handle SQLWarning
			SQLWarning warning = stmt.getWarnings();
			if (warning != null) {
				System.out.println("SQLWarning occurred:");
				while (warning != null) {
					System.out.println("Message: " +
warning.getMessage());
					System.out.println("SQLState: " +
warning.getSQLState());
```

```
                    System.out.println("Error code: " +
warning.getErrorCode());
                    warning = warning.getNextWarning();
                }
            }

            // Close resources
            rs.close();
            stmt.close();
            con.close();
        } catch (SQLException e) {
            // Handle SQLException
            System.out.println("SQLException occurred:");
            System.out.println("Message: " + e.getMessage());
            System.out.println("SQLState: " + e.getSQLState());
            System.out.println("Error code: " + e.getErrorCode());
            e.printStackTrace();
        }
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
Java SQLExceptionExample


SQLException occurred:
Message: ORA-00942: table or view does not exist
https://docs.oracle.com/error-help/db/ora-00942/
SQLState: 42000
Error code: 942
```

Cycle 8: Graphics Programming

Program 8.1	Date: 18/03/2024

Write a Java program that works as a simple calculator. Arrange Buttons for digits and the + - * % operations properly. Add a text field to display the result. Handle any possible exceptions like divide by zero. Use Java Swing.

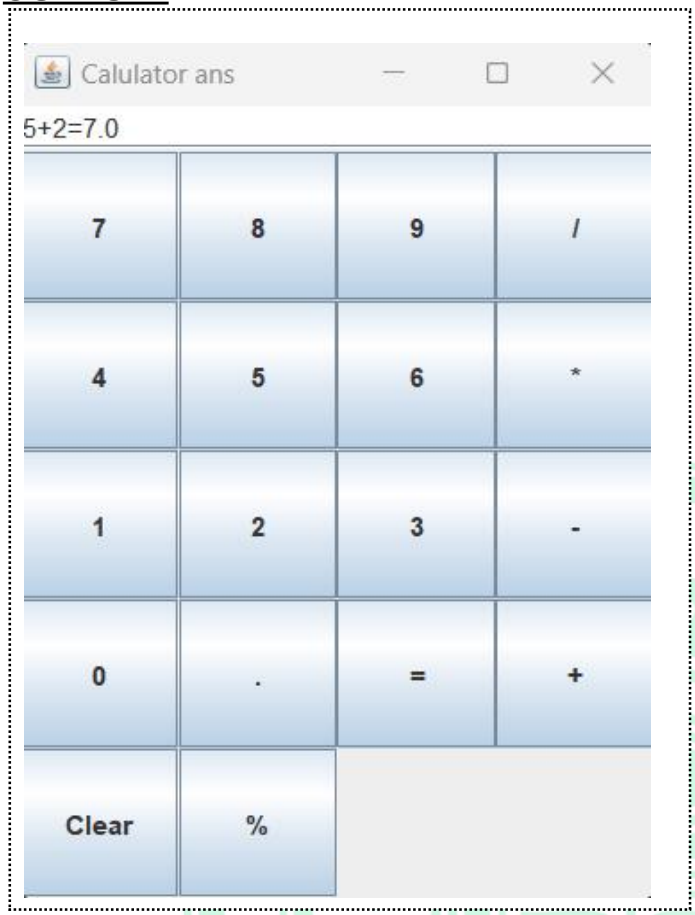
PROGRAM :

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

public class P14_1_calculator extends JFrame implements ActionListener {
	private JTextField textField;
	private JButton[] buttons;
	private String[] buttonLabels = {
			"7", "8", "9", "/",
			"4", "5", "6", "*",
			"1", "2", "3", "-",
			"0", ".", "=", "+",
			"Clear", "%"
	};
	private double num1, num2, result;
	private char operator;

	public P14_1_calculator() {
		setTitle("Calulator ans");
		setSize(300, 400);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setLocationRelativeTo(null);

		textField = new JTextField();
		// textField.setEditable(false);

		JPanel buttonPanel = new JPanel(new GridLayout(5, 4));

		buttons = new JButton[buttonLabels.length];
		for (int i = 0; i < buttonLabels.length; i++) {
```

```
            buttons[i] = new JButton(buttonLabels[i]);
            buttons[i].addActionListener(this);
            buttonPanel.add(buttons[i]);
        }

        add(textField, BorderLayout.NORTH);
        add(buttonPanel, BorderLayout.CENTER);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();
        try {
            switch (command) {
                case "Clear":
                    textField.setText("");
                    break;
                case "=":
                    if (operator != '\u0000') {
                        num2 = Double.parseDouble(textField.getText());
                        result = calculate();
                        textField.setText(String.valueOf(result));
                    }
                    break;
                case "%":
                    num1 = Double.parseDouble(textField.getText());
                    result = num1 / 100;
                    textField.setText(String.valueOf(result));
                    break;
                case "+":
                case "-":
                case "*":
                case "/":
                    num1 = Double.parseDouble(textField.getText());
                    operator = command.charAt(0);
                    textField.setText("");
                    break;
                default:
                    textField.setText(textField.getText() + command);
            }
        } catch (NumberFormatException ex) {
            textField.setText("Error");
        } catch (ArithmeticException ex) {
            textField.setText("Cannot divide by zero");
        }
```

```
    }

    private double calculate() {
        switch (operator) {
            case '+':
                return num1 + num2;
            case '-':
                return num1 - num2;
            case '*':
                return num1 * num2;
            case '/':
                if (num2 == 0)
                    throw new ArithmeticException();
                return num1 / num2;
        }
        return 0;
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            P14_1_calculator calculator = new P14_1_calculator();
            calculator.setVisible(true);
        });
    }
}
```

OUTPUT :



Program 8.2	Date: 12/03/2024

Write a Java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green. When a radio button is selected, the light is turned on, and only one light can be on at a time. No light is on when the program starts.

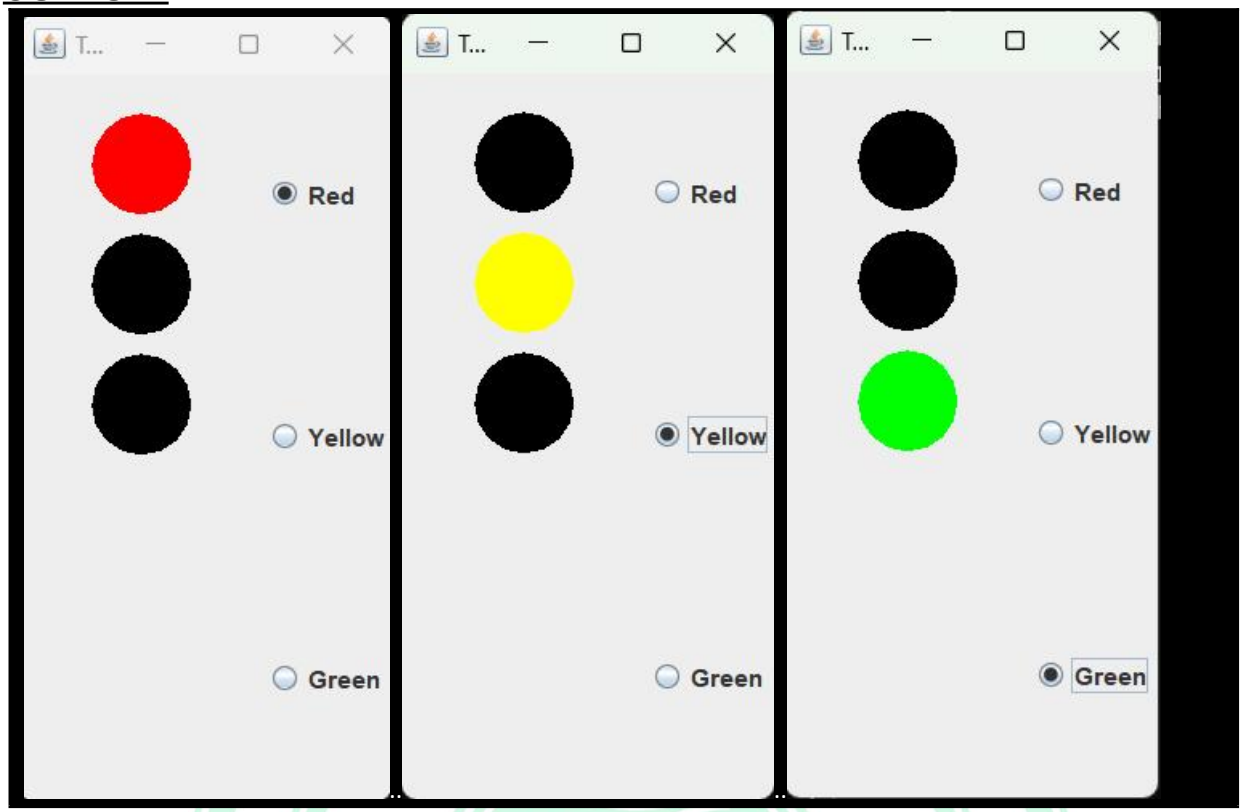
PROGRAM :

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

public class P14_2_traffic_light extends JFrame implements ActionListener
{
	private JRadioButton redButton, yellowButton, greenButton;
	private JPanel trafficPanel, buttonPanel;
	private ButtonGroup buttonGroup;

	public P14_2_traffic_light() {
		setTitle("Traffic Light Simulator");
		setSize(200, 400);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		trafficPanel = new JPanel() {
			@Override
			protected void paintComponent(Graphics g) {
				super.paintComponent(g);
				drawTrafficLight(g);
			}
		};
		trafficPanel.setPreferredSize(new Dimension(100, 300));

		buttonPanel = new JPanel();
		buttonPanel.setLayout(new GridLayout(3, 1));

		redButton = new JRadioButton("Red");
		redButton.addActionListener(this);

		yellowButton = new JRadioButton("Yellow");
		yellowButton.addActionListener(this);

		greenButton = new JRadioButton("Green");
```

```
        greenButton.addActionListener(this);

        buttonGroup = new ButtonGroup();
        buttonGroup.add(redButton);
        buttonGroup.add(yellowButton);
        buttonGroup.add(greenButton);

        buttonPanel.add(redButton);
        buttonPanel.add(yellowButton);
        buttonPanel.add(greenButton);

        add(trafficPanel, BorderLayout.CENTER);
        add(buttonPanel, BorderLayout.EAST);

        setVisible(true);
    }

    private void drawTrafficLight(Graphics g) {
        int diameter = 50;
        int centerX = (trafficPanel.getWidth() - diameter) / 2;
        int startY = 20;

        g.setColor(Color.black);
        g.fillOval(centerX, startY, diameter, diameter);
        g.fillOval(centerX, startY + diameter + 10, diameter, diameter);
        g.fillOval(centerX, startY + (diameter + 10) * 2, diameter,
diameter);

        if (redButton.isSelected()) {
            g.setColor(Color.red);
            g.fillOval(centerX, startY, diameter, diameter);
        } else if (yellowButton.isSelected()) {
            g.setColor(Color.yellow);
            g.fillOval(centerX, startY + diameter + 10, diameter,
diameter);
        } else if (greenButton.isSelected()) {
            g.setColor(Color.green);
            g.fillOval(centerX, startY + (diameter + 10) * 2, diameter,
diameter);
        }
    }

    @Override
    public void actionPerformed(ActionEvent e) {
```

```
            trafficPanel.repaint();
        }

        public static void main(String[] args) {
            SwingUtilities.invokeLater(() -> new P14_2_traffic_light());
        }
}
```

OUTPUT :



Cycle 9: Collection Framework

Program 9.1	Date: 15/03/2024

Write a Java program for the following:

4) **Create a doubly linked list of elements.**
5) **Delete a given element from the above list.**
6) **Display the contents of the list after deletion.**

PROGRAM :

```
class Node {
	int data;
	Node prev;
	Node next;

	public Node(int data) {
		this.data = data;
		this.prev = null;
		this.next = null;
	}
}

class DoublyLinkedList {
	Node head;
	Node tail;

	public DoublyLinkedList() {
		this.head = null;
		this.tail = null;
	}

	public void insert(int data) {
		Node newNode = new Node(data);
		if (head == null) {
			head = newNode;
			tail = newNode;
		} else {
```

```
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }

    public void delete(int data) {
        Node current = head;
        while (current != null) {
            if (current.data == data) {
                if (current.prev != null) {
                    current.prev.next = current.next;
                } else {
                    head = current.next;
                }
                if (current.next != null) {
                    current.next.prev = current.prev;
                } else {
                    tail = current.prev;
                }
                break;
            }
            current = current.next;
        }
    }

    public void display() {
        Node current = head;
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
        }
        System.out.println();
    }
}

public class P15_1_collection1 {
    public static void main(String[] args) {
        DoublyLinkedList list = new DoublyLinkedList();
```

```
        // Insert elements into the doubly linked list
        list.insert(1);
        list.insert(2);
        list.insert(3);
        list.insert(4);
        list.insert(5);

        // Display the contents of the list before deletion
        System.out.println("Contents of the list before deletion:");
        list.display();

        // Delete a given element from the list
        int elementToDelete = 3;
        list.delete(elementToDelete);

        // Display the contents of the list after deletion
        System.out.println("Contents of the list after deletion of " +
elementToDelete + ":");
        list.display();
    }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
Java P15_1_collection1

Contents of the list before deletion:
1 2 3 4 5
Contents of the list after deletion of 3: 1245
```

Program 9.2	Date: 15/03/2024

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

PROGRAM :

```
import java.util.Arrays;

public class P15_2_quicksort {

	public static void main(String[] args) {
		String[] names = {"John", "Alice", "Bob", "Eva", "Charlie", "David"};

		System.out.println("Original list of names: " + Arrays.toString(names));

		quickSort(names, 0, names.length - 1);

		System.out.println("Sorted list of names: " + Arrays.toString(names));
	}

	public static void quickSort(String[] arr, int low, int high) {
		if (low < high) {
			int pi = partition(arr, low, high);

			quickSort(arr, low, pi - 1);
			quickSort(arr, pi + 1, high);
		}
	}

	public static int partition(String[] arr, int low, int high) {
		String pivot = arr[high];
		int i = low - 1;
		for (int j = low; j < high; j++) {
			if (arr[j].compareTo(pivot) < 0) {
				i++;
```

```
                    String temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }
            String temp = arr[i + 1];
            arr[i + 1] = arr[high];
            arr[high] = temp;

            return i + 1;
        }
}
```

OUTPUT :

```
E:\MCA\SEM 2\JAVA PROGRAMMING (MCA202)\JAVA LAB>
Java P15_2_quicksort

Original list of names: [John, Alice, Bob, Eva, Charlie, David]
Sorted list of names: [Alice, Bob, Charlie, David, Eva, John]
```

Cycle 10: Capstone Project

PROJECT	Date: 18/03/2024
Fee Management System	

1. Introduction :

The **Fee Management System** is a Java Swing-based graphical user interface (GUI) application designed to manage fee records for students. It provides functionalities to add new fee records and make payments towards the existing fee amounts for students. The application interacts with a MySQL database to store and retrieve fee-related information.

2. Aim :

The aim of this application is to provide a simple and user-friendly interface for managing fee records of students. It allows administrators or users to add new fee records and update existing fee amounts by making payments, all through a graphical interface.

3. Technologies used :

I. **Java Swing**: Java's built-in GUI library used to create the graphical user interface.
II. **JDBC (Java Database Connectivity)**: Java API to connect and execute SQL queries with the MySQL database.
III. **MySQL**: Database system used to store and manage fee records.

4. Functionalities :

Add Fee Record:
Inputs: Student ID, Student Name, Fee Amount
Action: Adds a new fee record to the database with the provided information.
Process: The entered student ID, student name, and fee amount are inserted into the fees table in the MySQL database.
Make Payment:
Inputs: Student ID, Payment Amount
Action: Updates the existing fee amount for the specified student by deducting the payment amount.
Process: The fee amount for the student identified by the entered student **ID is** updated by subtracting the payment amount from the existing fee amount in the fees table.

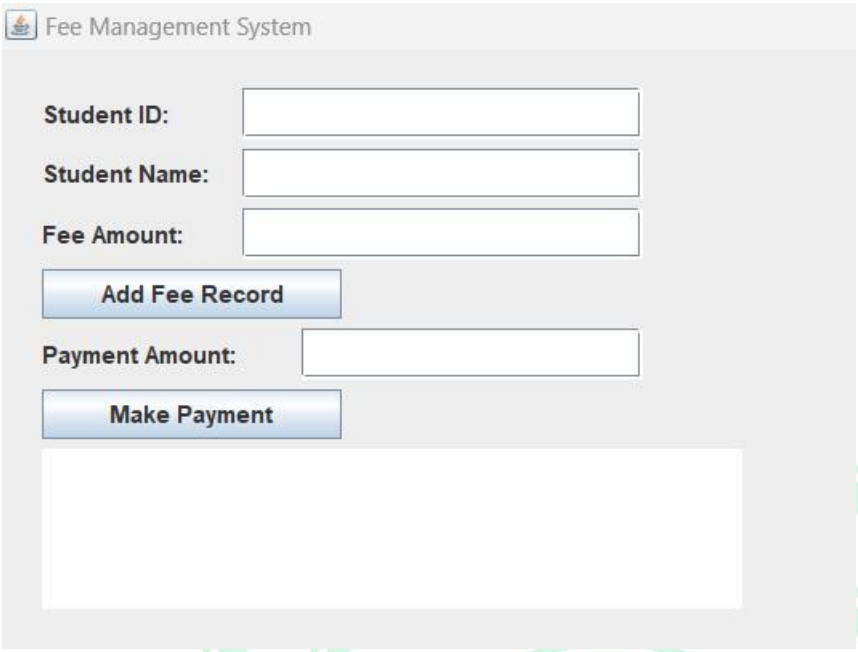
5. Source Code :

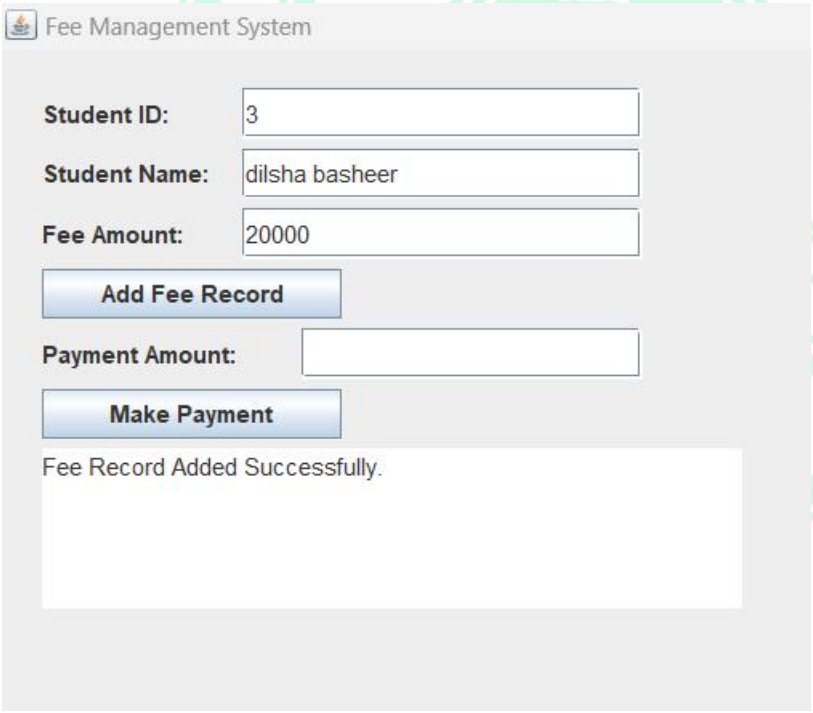
```
        import javax.swing.*;
import java.awt.event.*;
import java.sql.*;

public class FeeManagementSystemGUI extends JFrame implements ActionListener {
    private Connection con;
    private Statement st;
    private ResultSet rs;
    private JTextField studentIdField, studentNameField, feeAmountField, paymentField;
    private JTextArea outputArea;

    public FeeManagementSystemGUI() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/fees_db?characterEncoding=utf8", "root", "");
            st = con.createStatement();
        } catch (Exception e) {
            System.out.println("Error: " + e);
        }

        setTitle("Fee Management System");
        setSize(400, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(null);

        JLabel studentIdLabel = new JLabel("Student ID:");
        studentIdLabel.setBounds(20, 20, 80, 25);
        add(studentIdLabel);

        studentIdField = new JTextField();
        studentIdField.setBounds(120, 20, 200, 25);
        add(studentIdField);

        JLabel studentNameLabel = new JLabel("Student Name:");
        studentNameLabel.setBounds(20, 50, 100, 25);
        add(studentNameLabel);

        studentNameField = new JTextField();
```

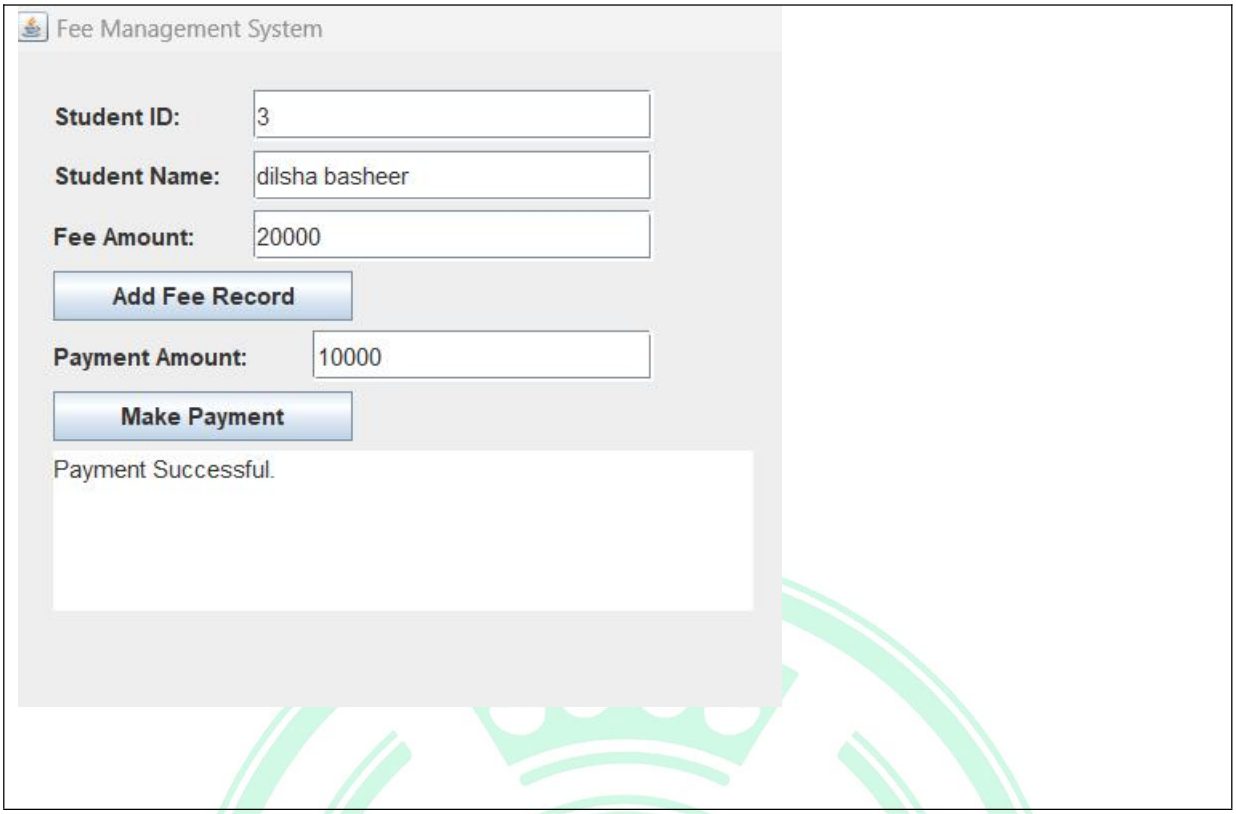
```
        studentNameField.setBounds(120, 50, 200, 25);
        add(studentNameField);

        JLabel feeAmountLabel = new JLabel("Fee Amount:");
        feeAmountLabel.setBounds(20, 80, 80, 25);
        add(feeAmountLabel);

        feeAmountField = new JTextField();
        feeAmountField.setBounds(120, 80, 200, 25);
        add(feeAmountField);

        JButton addButton = new JButton("Add Fee Record");
        addButton.setBounds(20, 110, 150, 25);
        addButton.addActionListener(this);
        add(addButton);

        JLabel paymentLabel = new JLabel("Payment Amount:");
        paymentLabel.setBounds(20, 140, 120, 25);
        add(paymentLabel);

        paymentField = new JTextField();
        paymentField.setBounds(150, 140, 170, 25);
        add(paymentField);

        JButton payButton = new JButton("Make Payment");
        payButton.setBounds(20, 170, 150, 25);
        payButton.addActionListener(this);
        add(payButton);

        outputArea = new JTextArea();
        outputArea.setBounds(20, 200, 350, 80);
        add(outputArea);

        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("Add Fee Record")) {
            try {
                int studentId = Integer.parseInt(studentIdField.getText());
                String studentName = studentNameField.getText();
                double feeAmount =
Double.parseDouble(feeAmountField.getText());

                String insertQuery = "INSERT INTO fees (student_id,
```

```
student_name, fee_amount) VALUES (" + studentId + ", '" + studentName + "',
" + feeAmount + ")";
                    st.executeUpdate(insertQuery);

                    outputArea.setText("Fee Record Added Successfully.");
                } catch (Exception ex) {
                    outputArea.setText("Error: " + ex.getMessage());
                }
            } else if (e.getActionCommand().equals("Make Payment")) {
                try {
                    int studentId = Integer.parseInt(studentIdField.getText());
                    double paymentAmount =
Double.parseDouble(paymentField.getText());

                    String updateQuery = "UPDATE fees SET fee_amount =
fee_amount - " + paymentAmount + " WHERE student_id = " + studentId;
                    int rowsUpdated = st.executeUpdate(updateQuery);

                    if (rowsUpdated > 0) {
                        outputArea.setText("Payment Successful.");
                    } else {
                        outputArea.setText("Payment Failed. Invalid Student
ID or Insufficient Balance.");
                    }
                } catch (Exception ex) {
                    outputArea.setText("Error: " + ex.getMessage());
                }
            }
        }

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

6. OUTPUT :


Fee Management System
Student ID:
Student Name:
Fee Amount:
Add Fee Record
Payment Amount:
Make Payment


Fee Management System
Student ID:
3
Student Name:
dilsha basheer
Fee Amount:
20000
Add Fee Record
Payment Amount:
Make Payment
Fee Record Added Successfully.


Fee Management System
Student ID:
3
Student Name:
dilsha basheer
Fee Amount:
20000
Add Fee Record
Payment Amount:
10000
Make Payment
Payment Successful.