INDEX

	• >= 501 units - Rs. 6 per unit If the type of the EB connection is commercial, calculate the amount to be paid as follows: • First 100 units - Rs. 2 per unit • 101-200 units - Rs. 4.50 per unit • 201 -500 units - Rs. 6 per unit • >= 501 units - Rs. 7 per unit constructor		
16	Develop a java application with a class Employee which consist: • Emp Name • Em Id • Address • E-Mail id • Mobile no. as members. Inherit the classes Programmer • Assistant Professor, • Associate Professor and • Professor from employee class. Add Basic Pay (BP) as the member of all the inherited classes with • 97% of BP as DA, • 10 % of BP as HRA, • 12% of BP as PF, • 0.1% of BP for staff club fund. Generate pay slips for the employees with their gross and net salary.	22-24	
17	Write a Java Program to create an abstract class named Shape that contains two integers and an empty method named print Area (). • Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. • Each one of the classes contains only the method print Area () that prints the area of the given shape.	25-26	
18	Write the following java programs: a) Write a Java Program to describe about try and catch blocks for handling exceptions. b) Write a Java Program to demonstrate about throw and throws keywords c) Write a Java program to implement user defined exception handling	27-28	
19	Write a Java Program to demonstrate String Buffer Class and String Builder Class	29	
20	Write a Java Program to perform the following operations: a) To create threads in java by extending Thread Class b) To create threads in java by implementing Runnable Interface	30	
21	Write a java program that connects to a database using JDBC and perform the following: a) add record. b) delete record. c) modify record. d) Retrieve record.	31-32	

Program No. 01

Write a java program to perform following operations using recursion:

I. Factorial.

II. LCM of the three numbers.

PROGRAM:

```
import java.util.Scanner;
public class Recursion {
    public static int factorial(int n) {
        if (n == 0 || n == 1)
            return 1;
        else
            return n * factorial(n - 1);
    }
    public static int lcm(int a, int b, int c) {
        return lcm(lcm(a, b), c);
    }
    public static int lcm(int a, int b) {
        return (a * b) / gcd(a, b);
    }
    public static int gcd(int a, int b) {
        if (b == 0)
            return a;
        else
            return gcd(b, a % b);
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to calculate factorial: ");
        int num = scanner.nextInt();
        System.out.println("Factorial of " + num + " is: " + factorial(num));
        System.out.print("Enter three numbers to calculate LCM: ");
        int num1 = scanner.nextInt();
        int num2 = scanner.nextInt();
        int num3 = scanner.nextInt();
        System.out.println("LCM of " + num1 + ", " + num2 + ", and " + num3 + " is: " +
         lcm(num1, num2, num3));
        scanner.close();
    }
}
```

OUTPUT:

Enter a number to calculate factorial: 4

Factorial of 4 is: 24

Enter three numbers to calculate LCM: 12 24 8

LCM of 12, 24, and 8 is: 24

Program No. - 02

Write a java program to find the square root of its own number.

PROGRAM:

```
import java.util.Scanner;
public class SquareRoot {
    public static double squareRoot(double num) {
        return squareRootHelper(num, num / 2);
    }
    private static double squareRootHelper(double num, double guess) {
        double newGuess = (guess + num / guess) / 2;
        if (Math.abs(newGuess - guess) < 0.00001) {
            return newGuess;
        } else {
            return squareRootHelper(num, newGuess);
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to find its square root: ");
        double number = scanner.nextDouble();
        double result = squareRoot(number);
        System.out.println("Square root of " + number + " is: " + result);
        scanner.close();
    }
}
```

OUTPUT:

Enter a number to find its square root: 144

Square root of 144.0 is: 12.0

Program No. - 03

Write a java program to perform:
I. Switch case II. Call by value III. Call by reference

PROGRAM:

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

    public static void switchCaseDemo(int choice) {
        switch (choice) {
            case 1:
                System.out.println("You chose option 1.");
                break;
            case 2:
                System.out.println("You chose option 2.");
                break;
            case 3:
                System.out.println("You chose option 3.");
                break;
            default:
                System.out.println("Invalid choice.");
        }
    }

    public static void callByValue(int num) {
        num += 10;
        System.out.println("Inside callByValue method: " + num);
    }

    public static void callByReference(int[] arr) {
        for (int i = 0; i < arr.length; i++) {
            arr[i] += 10;
        }

        System.out.println("Inside callByReference method: " + java.util.Arrays.toString(arr));
    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your choice (1, 2, or 3): ");
        int choice = scanner.nextInt();
        switchCaseDemo(choice);

        int num = 5;
        System.out.println("Before calling callByValue method: " + num);
        callByValue(num);

        System.out.println("After calling callByValue method: " + num);
```

```
        int[] array = {1, 2, 3, 4, 5};
        System.out.println("Before calling callByReference method: " +
         java.util.Arrays.toString(array));

        callByReference(array);

        System.out.println("After calling callByReference method: " +
         java.util.Arrays.toString(array));
        scanner.close();
    }
}
```

OUTPUT:

Enter your choice (1, 2, or 3): 1

You chose option 1.

Before calling callByValue method: 5

Inside callByValue method: 15

After calling callByValue method: 5

Before calling callByReference method: [1, 2, 3, 4, 5]

Inside callByReference method: [11, 12, 13, 14, 15]

After calling callByReference method: [11, 12, 13, 14, 15]

Program No. - 04

Write a java program to perform operation of Matrix Multiplication.

PROGRAM:

```
import java.util.Scanner;

public class MatrixMultiplication
{

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

        System.out.println("Enter the number of rows and columns for the first matrix:");

        int rows1 = scanner.nextInt();
        int cols1 = scanner.nextInt();
        int[][] matrix1 = new int[rows1][cols1];

        System.out.println("Enter the elements of the first matrix:");
        inputMatrix(scanner, matrix1);

        System.out.println("Enter the number of rows and columns for the second matrix:");
        int rows2 = scanner.nextInt();
        int cols2 = scanner.nextInt();
        int[][] matrix2 = new int[rows2][cols2];

        System.out.println("Enter the elements of the second matrix:");
        inputMatrix(scanner, matrix2);

        if (cols1 != rows2) {
            System.out.println("Matrix multiplication is not possible.");
            return;
        }

        int[][] resultMatrix = multiplyMatrices(matrix1, matrix2);

        System.out.println("Result of matrix multiplication:");
        displayMatrix(resultMatrix);

        scanner.close();
    }

    public static void inputMatrix(Scanner scanner, int[][] matrix)
    {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }
    }
```

```
    public static int[][] multiplyMatrices(int[][] matrix1, int[][] matrix2) {
        int rows1 = matrix1.length;
        int cols1 = matrix1[0].length;
        int cols2 = matrix2[0].length;

        int[][] resultMatrix = new int[rows1][cols2];

        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols2; j++) {
                for (int k = 0; k < cols1; k++) {
                    resultMatrix[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }

        return resultMatrix;
    }

    public static void displayMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int num : row) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }
}
```

OUTPUT:

Enter the number of rows and columns for the first matrix:

2 3

Enter the elements of the first matrix:

1 2 3 4 5 6

Enter the number of rows and columns for the second matrix:

3 2

Enter the elements of the second matrix:

6 5 4 3 2 1

Result of matrix multiplication:

20 14

56 41

Program No. - 05

Write a JAVA program to implement class mechanism.
1. Create a class 2. Create methods and 3. Invoke them inside main method

PROGRAM:

```
public class MyClass {
    public int add(int a, int b) {
        return a + b;
    }

    public int subtract(int a, int b) {
        return a - b;
    }

    public int multiply(int a, int b) {
        return a * b;
    }

    public double divide(int a, int b) {
        if (b == 0) {
            System.out.println("Error: Division by zero!");
            return Double.NaN;
        }
        return (double) a / b;
    }

    public static void main(String[] args) {

        MyClass myClass = new MyClass();

        int resultAddition = myClass.add(5, 3);
        System.out.println("Addition result: " + resultAddition);

        int resultSubtraction = myClass.subtract(10, 4);
        System.out.println("Subtraction result: " + resultSubtraction);

        int resultMultiplication = myClass.multiply(6, 7);
        System.out.println("Multiplication result: " + resultMultiplication);

        double resultDivision = myClass.divide(20, 5);
        System.out.println("Division result: " + resultDivision);
    }
}
```

OUTPUT:

Addition result: 8
Subtraction result: 6
Multiplication result: 42
Division result: 4.0

Program No. - 06

Write a java program to demonstrate the following Class:
I. Scanner class. II. Static class.

PROGRAM:

```
import java.util.Scanner;

public class ScannerStatic {

    static class MathUtils {

        public static int factorial(int n) {
            if (n == 0 || n == 1)
                return 1;
            else
                return n * factorial(n - 1);
        }
    }

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

        System.out.print("Enter a number to calculate its factorial: ");
        int num = scanner.nextInt();

        int fact = MathUtils.factorial(num);
        System.out.println("Factorial of " + num + " is: " + fact);

        scanner.close();
    }
}
```

OUTPUT:

```
Enter a number to calculate its factorial: 5
Factorial of 5 is: 120
```

Program No. - 07

Write Java program to demonstrate the concept of following:

a) dynamic binding,
b) differentiating method,
c) overloading method and,
d) overriding method.

PROGRAM:

```
class Animal {
    // Method that will be overridden
    void makeSound() {
        System.out.println("Animal makes a sound");
    }
}
class Dog extends Animal {
    // Method overriding
    @Override
    void makeSound() {
        System.out.println("Dog barks");
    }
    // Method overloading
    void makeSound(int numTimes) {
        for (int i = 0; i < numTimes; i++) {
            System.out.println("Woof");
        }
    }
}
public class Main {
    public static void main(String[] args) {
        // Dynamic binding
        Animal animal = new Dog(); //Creating an instance of Dog,but referred by Animal
                                   reference
        animal.makeSound(); // Calls Dog's overridden method
        // Method differentiation
        Dog dog = new Dog();
        dog.makeSound(); // Calls overridden method in Dog class
        dog.makeSound(3); // Calls overloaded method in Dog class
    }
}
```

OUTPUT:

```
Dog barks
Dog barks
Woof
Woof
Woof
```

Program No. - 08

Write a java program to perform the following inheritance:

a) Single b) Multi-level c) Hierarchical

PROGRAM:

```
// Single Inheritance
class Vehicle {
    void move() {
        System.out.println("Vehicle is moving");
}}
class Car extends Vehicle {
    void drive() {
        System.out.println("Car is being driven");
}}
// Multi-level Inheritance
class SportsCar extends Car {
    void speedUp() {
        System.out.println("Sports car is speeding up");
}}
// Hierarchical Inheritance
class Truck extends Vehicle {
    void carryLoad() {
        System.out.println("Truck is carrying a load");
}}
public class Inheritance {
    public static void main(String[] args) {
        // Single Inheritance
        Car car = new Car();
        car.move(); // inherited from Vehicle
        car.drive();
        // Multi-level Inheritance
        SportsCar sportsCar = new SportsCar();
        sportsCar.move(); // inherited from Vehicle
        sportsCar.drive(); // inherited from Car
        sportsCar.speedUp();
        // Hierarchical Inheritance
        Truck truck = new Truck();
        truck.move(); // inherited from Vehicle
        truck.carryLoad();
}}
```

OUTPUT:

```
Vehicle is moving
Car is being driven
Vehicle is moving
Car is being driven
Sports car is speeding up
Vehicle is moving
Truck is carrying a load
```

Program No. - 09

Write the following java programs:

a) Write a java program to operate super keyword in java.

PROGRAM:

```
class Person {

    String name;
    int age;

    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

class Student extends Person {
    String rollNumber;

    Student(String name, int age, String rollNumber) {
        super(name, age);
        this.rollNumber = rollNumber;
    }

    void display() {
        super.display();
        System.out.println("Roll Number: " + rollNumber);
    }
}

public class SuperKeyword {

    public static void main(String[] args) {
        Student student = new Student("Hassan", 20, "22BTCS012HY");
        student.display();
    }
}
```

OUTPUT:

```
Name: Hassan
Age: 20
Roll Number: 22BTCS012HY
```

b) Write a java program to demonstrate the final method.

PROGRAM:

```
class Parent {

    // Final method
    final void display() {
        System.out.println("This is the final method from Parent class method");
    }

}

class Child extends Parent {

    // Attempting to override a final method will result in a compile-time error
    /*void display() {
        System.out.println("Child class method");
    }*/

}

public class FinalMethod {

    public static void main(String[] args) {
        Parent parent = new Parent();
        parent.display();
    }

}
```

OUTPUT:

This is the final method from Parent class method

Program No. - 10

Write the following java programs:

a) Write a java program to define and implement an interface

PROGRAM:

```
interface Shape {

    double area();
    double perimeter();
}

class Circle implements Shape {

    double radius;
    Circle(double r) { radius = r; }
    public double area() { return Math.PI * radius * radius; }
    public double perimeter() { return 2 * Math.PI * radius; }
}

class Rectangle implements Shape {

    double length, width;
    Rectangle(double l, double w) { length = l; width = w; }
    public double area() { return length * width; }
    public double perimeter() { return 2 * (length + width); }
}

class Interface {

    public static void main(String[] args) {

        Circle circle = new Circle(5);
        System.out.println("Circle Area: " + circle.area());
        System.out.println("Circle Perimeter: " + circle.perimeter());
        Rectangle rectangle = new Rectangle(4, 6);
        System.out.println("Rectangle Area: " + rectangle.area());
        System.out.println("Rectangle Perimeter: " + rectangle.perimeter());

    }
}
```

OUTPUT:

```
Circle Area: 78.53981633974483
Circle Perimeter: 31.41592653589793
Rectangle Area: 24.0
Rectangle Perimeter: 20.0
```

b) Write a java program to implement Interface using extends keyword

PROGRAM:

```
interface Person {
    void introduce();
}

class Student implements Person {
    private String name;
    private int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public void introduce() {
        System.out.println("Hi, I'm " + name + " and I'm " + age + " years old.");
    }
}

public class ExtendKeyword {
    public static void main(String[] args) {
        Student student = new Student("Hassan", 20);
        student.introduce();
    }
}
```

OUTPUT:

Hi, I'm Hassan and I'm 20 years old.

Program No. - 11

Write a java program to demonstrate this keyword

- Write a java program to demonstrate this () method.

PROGRAM:

```
class Box {
    private int length;
    private int width;
    private int height;

    public Box(int length, int width, int height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }

    public Box(int side) {
        this(side, side, side);
    }

    public int calculateVolume() {
        return length * width * height;
    }
}

public class This {
    public static void main(String[] args) {
        Box box1 = new Box(5, 4, 3);
        System.out.println("Volume of box1: " + box1.calculateVolume());

        Box box2 = new Box(5);
        System.out.println("Volume of box2: " + box2.calculateVolume());
    }
}
```

OUTPUT:

Volume of box1: 60

Volume of box2: 125

Program No. - 12

Write a java program to create user defined package.

PROGRAM:

```
package UserPackage;

public class MyPackageDirect {
    public void sayHello(){
        System.out.println("Hello from the user-defined package!");
    }
}
```

```
import UserPackage.MyPackageDirect;

public class PackageDir {
    public static void main(String[] args) {
        MyPackageDirect obj = new MyPackageDirect();
        obj.sayHello();
    }
}
```

OUTPUT:

Hello from the user-defined package!

Program No. - 13

Write a java program to demonstrate.

a) Constructor
b) Default constructor
c) Copy constructor

PROGRAM:

```
public class Constructor {
    private int value;

    // a) Constructor
    public Constructor(int value) {
        this.value = value;
    }

    // b) Default constructor
    public Constructor() {
        this.value = 0; // Default value
    }

    // c) Copy constructor
    public Constructor(Constructor other) {
        this.value = other.value; // Copying value from another object
    }

    // Method to get the value
    public int getValue() {
        return this.value;
    }

    public static void main(String[] args) {
        // Creating objects using different constructors
        Constructor obj1 = new Constructor(10); // Constructor
        Constructor obj2 = new Constructor();    // Default constructor
        Constructor obj3 = new Constructor(obj1); // Copy constructor

        // Displaying values
        System.out.println("Value of obj1: " + obj1.getValue());
        System.out.println("Value of obj2: " + obj2.getValue());
        System.out.println("Value of obj3: " + obj3.getValue());
    }
}
```

OUTPUT:

Value of obj1: 10

Value of obj2: 0

Value of obj3: 10

Program No. - 14

Write the following java programs:

a) Method overloading

PROGRAM:

```
public class MethodOverloading {

    // Method with same name but different parameters
    public void display(int a) {
        System.out.println("Method with int parameter: " + a);
    }

    public void display(int a, int b) {
        System.out.println("Method with two int parameters: " + a + ", " + b);
    }

    public void display(double a) {
        System.out.println("Method with double parameter: " + a);
    }

    public static void main(String[] args) {

        MethodOverloading obj = new MethodOverloading();
        obj.display(5);
        obj.display(5, 10);
        obj.display(5.5);

    }
}
```

OUTPUT:

Value of obj1: 10

Value of obj2: 0

Value of obj3: 10

b) Constructor overloading

PROGRAM:

```
public class ConstructorOverloading {
    private int num1;
    private int num2;

    // Default constructor
    public ConstructorOverloading() {
        num1 = 0;
        num2 = 0;
    }

    // Parameterized constructor with one parameter
    public ConstructorOverloading(int a) {
        num1 = a;
        num2 = 0;
    }

    // Parameterized constructor with two parameters
    public ConstructorOverloading(int a, int b) {
        num1 = a;
        num2 = b;
    }

    public void display() {
        System.out.println("num1: " + num1 + ", num2: " + num2);
    }

    public static void main(String[] args) {
        ConstructorOverloading obj1 = new ConstructorOverloading();
        ConstructorOverloading obj2 = new ConstructorOverloading(5);
        ConstructorOverloading obj3 = new ConstructorOverloading(5, 10);

        obj1.display();
        obj2.display();
        obj3.display();
    }
}
```

OUTPUT:

num1: 0, num2: 0

num1: 5, num2: 0

num1: 5, num2: 10

Program No. – 15

Develop a Java application to generate Electricity bill. Create a class with the following members:

- Consumer no.
- Consumer name
- Previous month reading
- Current month reading
- Type of EB connection (domestic or commercial).

Compute the bill amount using the following tariff. If the type of the EB connection is domestic, calculate the amount to be paid as follows:

- First 100 units - Rs. 1 per unit
- 101-200 units - Rs. 2.50 per unit
- 201 -500 units - Rs. 4 per unit
- >= 501 units - Rs. 6 per unit

If the type of the EB connection is commercial, calculate the amount to be paid as follows:

- First 100 units - Rs. 2 per unit
- 101-200 units - Rs. 4.50 per unit
- 201 -500 units - Rs. 6 per unit
- >= 501 units - Rs. 7 per unit constructor

PROGRAM:

```
import java.util.Scanner;
class ElectricityBill {
    private int consumerNo;
    private String consumerName;
    private double prevMonthReading;
    private double currMonthReading;
    private String connectionType;
    public ElectricityBill(int consumerNo, String consumerName, double prevMonthReading, double
                           currMonthReading, String connectionType) {
        this.consumerNo = consumerNo;
        this.consumerName = consumerName;
        this.prevMonthReading = prevMonthReading;
        this.currMonthReading = currMonthReading;
        this.connectionType = connectionType;
    }
    public void calculateBill() {
        double unitsConsumed = currMonthReading - prevMonthReading;
        double billAmount = 0;
        if (connectionType.equalsIgnoreCase("domestic")) {
            if (unitsConsumed <= 100) {
                billAmount = unitsConsumed * 1;
            } else if (unitsConsumed <= 200) {
                billAmount = 100 * 1 + (unitsConsumed - 100) * 2.5;
            } else if (unitsConsumed <= 500) {
                billAmount = 100 * 1 + 100 * 2.5 + (unitsConsumed - 200) * 4;
            } else {
                billAmount = 100 * 1 + 100 * 2.5 + 300 * 4 + (unitsConsumed - 500) * 6;
            }
        } else if (connectionType.equalsIgnoreCase("commercial")) {
            if (unitsConsumed <= 100) {
                billAmount = unitsConsumed * 2;
            } else if (unitsConsumed <= 200) {
```

```
                billAmount = 100 * 2 + (unitsConsumed - 100) * 4.5;
            } else if (unitsConsumed <= 500) {
                billAmount = 100 * 2 + 100 * 4.5 + (unitsConsumed - 200) * 6;
            } else {
                billAmount = 100 * 2 + 100 * 4.5 + 300 * 6 + (unitsConsumed - 500) * 7;
            }
        } else {
            System.out.println("Invalid connection type. Please enter 'domestic' or
                                'commercial'.");
            return;
        }
        System.out.println("\nElectricity Bill");
        System.out.println("Consumer No.: " + consumerNo);
        System.out.println("Consumer Name: " + consumerName);
        System.out.println("Units Consumed: " + unitsConsumed);
        System.out.println("Bill Amount: Rs. " + billAmount);
    }
}
public class ElectricityBillGenerator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Consumer No.: ");
        int consumerNo = scanner.nextInt();
        System.out.print("Enter Consumer Name: ");
        String consumerName = scanner.next();
        System.out.print("Enter Previous Month Reading: ");
        double prevMonthReading = scanner.nextDouble();
        System.out.print("Enter Current Month Reading: ");
        double currMonthReading = scanner.nextDouble();
        System.out.print("Enter Type of EB Connection (domestic/commercial): ");
        String connectionType = scanner.next();
        ElectricityBill bill = new ElectricityBill(consumerNo, consumerName, prevMonthReading,
                                    currMonthReading, connectionType);
        bill.calculateBill();
        scanner.close();
    }
}
```

OUTPUT:

Enter Consumer No.: 1212
Enter Consumer Name: Hassan
Enter Previous Month Reading: 512
Enter Current Month Reading: 786
Enter Type of EB Connection (domestic/commercial): Commercial

Electricity Bill
Consumer No.: 1212
Consumer Name: Hassan
Units Consumed: 274.0
Bill Amount: Rs. 1094.0

Program No. – 16

Develop a java application with a class Employee which consist:

- Emp Name
- Em Id
- Address
- E-Mail id
- Mobile no. as members.

Inherit the classes Programmer

- Assistant Professor,
- Associate Professor and
- Professor from employee class.

Add Basic Pay (BP) as the member of all the inherited classes with

- 97% of BP as DA,
- 10 % of BP as HRA,
- 12% of BP as PF,
- 0.1% of BP for staff club fund.

Generate pay slips for the employees with their gross and net salary.

PROGRAM:

```
import java.util.Scanner;

class Employee {
    protected String empName;
    protected int empId;
    protected String address;
    protected String emailId;
    protected String mobileNo;

    public Employee(String empName, int empId, String address, String emailId, String
                            mobileNo) {
        this.empName = empName;
        this.empId = empId;
        this.address = address;
        this.emailId = emailId;
        this.mobileNo = mobileNo;
    }
}

class Programmer extends Employee {
    protected double basicPay;

    public Programmer(String empName, int empId, String address, String emailId, String
                        mobileNo, double basicPay) {
        super(empName, empId, address, emailId, mobileNo);
        this.basicPay = basicPay;
    }


    public double calculateDA() {
```

```
            return 0.97 * basicPay;
        }

        public double calculateHRA() {
            return 0.10 * basicPay;
        }

        public double calculatePF() {
            return 0.12 * basicPay;
        }

        public double calculateStaffClubFund() {
            return 0.001 * basicPay;
        }

        public double calculateGrossSalary() {
            return basicPay + calculateDA() + calculateHRA() + calculateStaffClubFund();
        }

        public double calculateNetSalary() {
            return calculateGrossSalary() - calculatePF();
        }

        public void generatePaySlip() {
            System.out.println("\u001B[1m\u001B[4m \n\tPay Slip\u001B[0m:");
            System.out.println("Employee Name: " + empName);
            System.out.println("Employee ID: " + empId);
            System.out.println("Basic Pay: " + basicPay);
            System.out.println("DA: " + calculateDA());
            System.out.println("HRA: " + calculateHRA());
            System.out.println("PF: " + calculatePF());
            System.out.println("Staff Club Fund: " + calculateStaffClubFund());
            System.out.println("Gross Salary: " + calculateGrossSalary());
            System.out.println("Net Salary: " + calculateNetSalary());
        }
    }


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

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

            System.out.print("Enter Employee ID: ");
            int empId = scanner.nextInt();
            scanner.nextLine();

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

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

```
        String emailId = scanner.nextLine();

        System.out.print("Enter Mobile No.: ");
        String mobileNo = scanner.nextLine();

        System.out.print("Enter Basic Pay: ");
        double basicPay = scanner.nextDouble();

        Programmer programmer = new Programmer(empName, empId, address, emailId, mobileNo,
basicPay);
        programmer.generatePaySlip();

        scanner.close();
    }
}
```

OUTPUT:

Enter Employee Name: Hassan

Enter Employee ID: 12

Enter Address: Hyderabad

Enter Email ID: hassan@gmail.com

Enter Mobile No.: 6299310210

Enter Basic Pay: 100000

Pay Slip:

Employee Name: Hassan

Employee ID: 12

Basic Pay: 100000.0

DA: 97000.0

HRA: 10000.0

PF: 12000.0

Staff Club Fund: 100.0

Gross Salary: 207100.0

Net Salary: 195100.0

Program No. - 17

Write a Java Program to create an abstract class named Shape that contains two integers and an empty method named print Area ().

- Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape.
- Each one of the classes contains only the method print Area () that prints the area of the given shape.

PROGRAM:

```
abstract class Shape {
    protected int base;
    protected int height;

    public abstract void printArea();
}

class Rectangle extends Shape {
    public Rectangle(int base, int height) {
        this.base = base;
        this.height = height;
    }

    @Override
    public void printArea() {
        int area = base * height;
        System.out.println("Area of Rectangle: " + area);
    }
}

class Triangle extends Shape {
    public Triangle(int base, int height) {
        this.base = base;
        this.height = height;
    }

    @Override
    public void printArea() {
        double area = 0.5 * base * height;
        System.out.println("Area of Triangle: " + area);
    }
}

class Circle extends Shape {
    private final double PI = 3.14159;
    private int radius;

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

    @Override
```

```
    public void printArea() {
        double area = PI * radius * radius;
        System.out.println("Area of Circle: " + area);
    }
}

public class ShapeTester {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(5, 10);
        rectangle.printArea();

        Triangle triangle = new Triangle(6, 12);
        triangle.printArea();

        Circle circle = new Circle(7);
        circle.printArea();
    }
}
```

OUTPUT:

Area of Rectangle: 50

Area of Triangle: 36.0

Area of Circle: 153.93791

Program No. - 18

Write the following java programs:

a) Write a Java Program to describe about try and catch blocks for handling exceptions.

PROGRAM:

```
public class ExceptionHandling {
    public static void main(String[] args) {
        try {
            int result = divide(10, 0);
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("An arithmetic exception occurred: " + e.getMessage());
        } finally {
             System.out.println("Finally block executed.");
        }
    }
    public static int divide(int num, int denom) {
        return num / denom;
    }
}
```

OUTPUT:

An arithmetic exception occurred: / by zero
Finally block executed.

b) Write a Java Program to demonstrate about throw and throws keywords

PROGRAM:

```
import java.io.IOException;
public class ThrowAndThrows {
    public static void main(String[] args) {
        try {
            readDataFromFile("example.txt");
        } catch (IOException e) {
            System.out.println("An IO exception occurred: " + e.getMessage());
        }
    }
    public static void readDataFromFile(String filename) throws IOException {
        if (!filename.equals("example.txt")) {
            throw new IOException("File not found: " + filename);
        } else {
            System.out.println("Reading data from file: " + filename);
        }
    }
    public static void validateAge(int age) {
        if (age < 0) {
            throw new IllegalArgumentException("Age cannot be negative");
        } else {
            System.out.println("Age: " + age);
        }
    }
}
```

OUTPUT:

Reading data from file: example.txt

c) Write a Java program to implement user defined exception handling

PROGRAM:

```
class CustomException extends Exception {
    public CustomException(String message) {
        super(message);
    }
}
public class UserDefinedExceptionHandling {
    public static void main(String[] args) {
        try {
            validateInput(9);
        } catch (CustomException e) {
            System.out.println("Custom exception occurred: " + e.getMessage());
        }
    }
    public static void validateInput(int value) throws CustomException {
        if (value < 10) {
            throw new CustomException("Input value should be greater than or equal to 10");
        } else {
            System.out.println("Input value is valid: " + value);
        }
    }
}
```

OUTPUT:

Custom exception occurred: Input value should be greater than or equal to 10

Program No. - 19

Write a Java Program to demonstrate String Buffer Class and String Builder Class

PROGRAM:

```
public class StringBufferStringBuilder {

    public static void main(String[] args) {

        // StringBuffer Example
        StringBuffer stringBuffer = new StringBuffer("Hello!");
        System.out.println("StringBuffer: " + stringBuffer);

        // Append method
        stringBuffer.append(" Hassan");
        System.out.println("After appending: " + stringBuffer);

        // StringBuilder Example
        StringBuilder stringBuilder = new StringBuilder("Hello!");
        System.out.println("StringBuilder: " + stringBuilder);

        // Insert method
        stringBuilder.insert(5, " Java");
        System.out.println("After inserting: " + stringBuilder);

    }

}
```

OUTPUT:

StringBuffer: Hello!

After appending: Hello! Hassan

StringBuilder: Hello!

After inserting: Hello Java!

Program No. - 20

Write a Java Program to perform the following operations:

a) To create threads in java by extending Thread Class
b) To create threads in java by implementing Runnable Interface

PROGRAM:

```
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread created by extending Thread class is running.");
    }
}

class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Thread created by implementing Runnable interface is running.");
    }
}

public class ThreadCreation {
    public static void main(String[] args) {

        MyThread thread1 = new MyThread();
        thread1.start();

        MyRunnable myRunnable = new MyRunnable();
        Thread thread2 = new Thread(myRunnable);
        thread2.start();
    }
}
```

OUTPUT:

Thread created by extending Thread class is running.

Thread created by implementing Runnable interface is running.

Program No. - 21

Write a java program that connects to a database using JDBC and perform the following:

a) add record.
b) delete record.
c) modify record.
d) Retrieve record.

PROGRAM:

```
import java.sql.*;
public class DatabaseOperations {
    // JDBC URL, username, and password of MySQL server
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/your_database_name";
    private static final String USERNAME = "your_username";
    private static final String PASSWORD = "your_password";
    public static void main(String[] args) {
        try (Connection connection = DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD))
                                {
            System.out.println("Connected to the database.");

            // a) Add record
            addRecord(connection, "Rehan", 18);

            // b) Delete record
            deleteRecord(connection, "Rehan");

            // c) Modify record
            modifyRecord(connection, "Hassan", 20);

            // d) Retrieve record
            retrieveRecord(connection, "Hassan");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    // Add record
    private static void addRecord(Connection connection, String name, int age) throws
                        SQLException {
        String query = "INSERT INTO your_table_name (name, age) VALUES (?, ?)";
        try (PreparedStatement statement = connection.prepareStatement(query)) {
            statement.setString(1, name);
            statement.setInt(2, age);
            int rowsInserted = statement.executeUpdate();
            if (rowsInserted > 0) {
                System.out.println("Record added successfully.");
            }
        }
    }
    // Delete record
    private static void deleteRecord(Connection connection, String name) throws SQLException {
        String query = "DELETE FROM your_table_name WHERE name=?";
```

```
            try (PreparedStatement statement = connection.prepareStatement(query)) {
                statement.setString(1, name);
                int rowsDeleted = statement.executeUpdate();
                if (rowsDeleted > 0) {
                    System.out.println("Record deleted successfully.");
                }
            }
        }
        // Modify record
        private static void modifyRecord(Connection connection, String name, int newAge) throws
                                                SQLException {
            String query = "UPDATE your_table_name SET age=? WHERE name=?";
            try (PreparedStatement statement = connection.prepareStatement(query)) {
                statement.setInt(1, newAge);
                statement.setString(2, name);
                int rowsUpdated = statement.executeUpdate();
                if (rowsUpdated > 0) {
                    System.out.println("Record modified successfully.");
                }
            }
        }
        // Retrieve record
        private static void retrieveRecord(Connection connection, String name) throws SQLException
{
            String query = "SELECT * FROM your_table_name WHERE name=?";
            try (PreparedStatement statement = connection.prepareStatement(query)) {
                statement.setString(1, name);
                ResultSet resultSet = statement.executeQuery();
                while (resultSet.next()) {
                    String retrievedName = resultSet.getString("name");
                    int retrievedAge = resultSet.getInt("age");
                    System.out.println("Name: " + retrievedName + ", Age: " + retrievedAge);
                }
            }
        }
}
```

OUTPUT:

Connected to the database.
Record added successfully.

Records before deletion:
ID: 1, Name: Rehan, Age: 18

Record deleted successfully.
Records after deletion:

Records after modification:
ID: 2, Name: Hassan, Age: 20

Connection closed.