Java Important Programs

1) Design an applet using Java Swing like:

Roll No. – TextBox
Name - TextBox
Address - Text Area
Submit - Button
followed by a message please enter your personal details.

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

public class PersonalDetailsApplet extends JApplet {
    private JTextField rollNoField, nameField;
    private JTextArea addressArea;

    public void init() {
        // Set layout
        setLayout(new GridLayout(4, 2));

        // Create components
        JLabel rollNoLabel = new JLabel("Roll No.: ");
        JLabel nameLabel = new JLabel("Name: ");
        JLabel addressLabel = new JLabel("Address: ");

        rollNoField = new JTextField(20);
        nameField = new JTextField(20);
        addressArea = new JTextArea(5, 20);

        JButton submitButton = new JButton("Submit");
        submitButton.addActionListener(new SubmitListener());

        // Add components to applet
        add(rollNoLabel);
        add(rollNoField);
        add(nameLabel);
        add(nameField);
        add(addressLabel);
        add(new JScrollPane(addressArea));
        add(new JLabel()); // Empty cell
        add(submitButton);

        // Set applet size
        setSize(400, 200);
    }

    class SubmitListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String rollNo = rollNoField.getText();
            String name = nameField.getText();
            String address = addressArea.getText();

            // Validate if fields are not empty
            if (rollNo.isEmpty() || name.isEmpty() || address.isEmpty()) {
                JOptionPane.showMessageDialog(null, "Please enter all personal
details.", "Error", JOptionPane.ERROR_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Personal details
submitted:\nRoll No.: " + rollNo + "\nName: " + name + "\nAddress: " +
address, "Success", JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
}
```

2) Write a program to implement Money class. This class should have fields for initializing a rupee and paisa value.

The paisa value will be in the range of 0-99 with the paisa being the same sign as that of rupees. The class should have all reasonable constructors, addition and subtraction methods, and a main) method that provides a thorough test of all the methods in the class.

```java
public class Money {
    private int rupees;
    private int paisa;

    // Constructors
    public Money() {
        this.rupees = 0;
        this.paisa = 0;
    }

    public Money(int rupees, int paisa) {
        this.rupees = rupees;
        this.paisa = paisa;
        normalize(); // Normalize paisa value
    }
    // Getters and setters
    public int getRupees() {
        return rupees;
    }

    public void setRupees(int rupees) {
        this.rupees = rupees;
    }

    public int getPaisa() {
        return paisa;
    }

    public void setPaisa(int paisa) {
        this.paisa = paisa;
        normalize(); // Normalize paisa value
    }

    // Addition method
    public Money add(Money other) {
        int newRupees = this.rupees + other.rupees;
        int newPaisa = this.paisa + other.paisa;
        return new Money(newRupees, newPaisa);
    }

    // Subtraction method
    public Money subtract(Money other) {
        int newRupees = this.rupees - other.rupees;
        int newPaisa = this.paisa - other.paisa;
        return new Money(newRupees, newPaisa);
    }

    // Normalize paisa value to be in the range 0-99
    private void normalize() {
        if (this.paisa < 0) {
            int borrow = (-this.paisa - 1) / 100 + 1;
            this.rupees -= borrow;
            this.paisa += borrow * 100;
        } else if (this.paisa >= 100) {
            int carry = this.paisa / 100;
            this.rupees += carry;
            this.paisa -= carry * 100;
        }
    }

    // Main method for testing
    public static void main(String[] args) {
        Money money1 = new Money(10, 50);
        Money money2 = new Money(5, 75);

        // Addition test
        Money sum = money1.add(money2);
        System.out.println("Sum: " + sum.getRupees() + " Rupees, " +
sum.getPaisa() + " Paisa");

        // Subtraction test
        Money difference = money1.subtract(money2);
        System.out.println("Difference: " + difference.getRupees() + " Rupees,
" + difference.getPaisa() + " Paisa");
    }
}
```

3) Create a class Order that performs order processing of a single item. The class has five data members: cust _name, cust_no, qty _ordered, unit_price and total _price. Include set and get methods for each field except the total price field.

The set method prompts the user for values for each field.

This class also needs a method to compute the total_ price and a method to display the field values. Create subclass ShippedOrder that overrides computePrice by adding a shipping and handling charge of $10.00. Write an, application UseOrder that instantiates an object of each of these classes. Prompt the user for data for the Order object and display the result. Then prompt the user for data for the ShippedOrder object and display the results.

```
import java.util.Scanner;

// Order class
class Order {
    private String custName;
    private String custNo;
    private int qtyOrdered;
    private double unitPrice;
    private double totalPrice;

    // Setters and getters
    public void setCustName(String custName) {
        this.custName = custName;
    }

    public String getCustName() {
        return custName;
```

```java
    }

    public void setCustNo(String custNo) {
        this.custNo = custNo;
    }

    public String getCustNo() {
        return custNo;
    }

    public void setQtyOrdered(int qtyOrdered) {
        this.qtyOrdered = qtyOrdered;
    }

    public int getQtyOrdered() {
        return qtyOrdered;
    }

    public void setUnitPrice(double unitPrice) {
        this.unitPrice = unitPrice;
    }

    public double getUnitPrice() {
        return unitPrice;
    }

    // Method to compute total price
    public void computePrice() {
        totalPrice = qtyOrdered * unitPrice;
    }

    // Method to display field values
    public void display() {
        System.out.println("Customer Name: " + custName);
        System.out.println("Customer Number: " + custNo);
        System.out.println("Quantity Ordered: " + qtyOrdered);
        System.out.println("Unit Price: $" + unitPrice);
        System.out.println("Total Price: $" + totalPrice);
    }
}

// ShippedOrder subclass
class ShippedOrder extends Order {
    private static final double SHIPPING_CHARGE = 10.00;

    // Override computePrice method to include shipping charge
    @Override
    public void computePrice() {
        super.computePrice();
        totalPrice += SHIPPING_CHARGE;
    }
}

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

        // Create Order object
        Order order = new Order();
        System.out.println("Enter customer name:");
        order.setCustName(scanner.nextLine());
        System.out.println("Enter customer number:");
        order.setCustNo(scanner.nextLine());
        System.out.println("Enter quantity ordered:");
        order.setQtyOrdered(scanner.nextInt());
        System.out.println("Enter unit price:");
        order.setUnitPrice(scanner.nextDouble());

        // Compute total price
        order.computePrice();

        // Display Order details
        System.out.println("\nOrder details:");
        order.display();

        // Create ShippedOrder object
        ShippedOrder shippedOrder = new ShippedOrder();
        System.out.println("\nEnter customer name for shipped order:");
        scanner.nextLine(); // Consume newline character
        shippedOrder.setCustName(scanner.nextLine());
        System.out.println("Enter customer number for shipped order:");
        shippedOrder.setCustNo(scanner.nextLine());
        System.out.println("Enter quantity ordered for shipped order:");
        shippedOrder.setQtyOrdered(scanner.nextInt());
        System.out.println("Enter unit price for shipped order:");
        shippedOrder.setUnitPrice(scanner.nextDouble());

        // Compute total price including shipping charge
        shippedOrder.computePrice();

        // Display ShippedOrder details
        System.out.println("\nShipped order details:");
        shippedOrder.display();

        scanner.close();
    }
}
```

4) Write an interface Numbers with a method int process (int x, int y). Write a class Sum in which the process method finds the sum of two numbers and returns an int value. Write another class Average in which the process method finds the average of the two numbers and returns an int value. Implement an application class Demo where the functionality of above two classes are tested that exhibit run time polymorphism.

```java
// Numbers interface
interface Numbers {
    int process(int x, int y);
}

// Sum class implementing Numbers interface
class Sum implements Numbers {
    // Method to find the sum of two numbers
    public int process(int x, int y) {
        return x + y;
    }
}

// Average class implementing Numbers interface
class Average implements Numbers {
    // Method to find the average of two numbers
    public int process(int x, int y) {
        return (x + y) / 2;
    }
}

// Demo application class
public class Demo {
    public static void main(String[] args) {
        Numbers operation;

        // Create Sum object and demonstrate runtime polymorphism
        operation = new Sum();
        System.out.println("Sum of 10 and 20: " + operation.process(10, 20));

        // Create Average object and demonstrate runtime polymorphism
        operation = new Average();
        System.out.println("Average of 10 and 20: " + operation.process(10,
20));
    }
}
```

5) Implement a package Geometry which contains a class Circle. The Circle class has one instance variable radius (double) and two methods void setRadius (double) to set the radius of the circle and double getRadius to return the radius of the circle. Also implement an application class Demo where the Geometry package is imported and the area and perimeter of the circle is calculated.

```java
package Geometry;

public class Circle {
    private double radius;

    // Method to set the radius of the circle
    public void setRadius(double radius) {
        this.radius = radius;
    }

    // Method to get the radius of the circle
    public double getRadius() {
        return radius;
    }

    // Method to calculate the area of the circle
    public double calculateArea() {
        return Math.PI * radius * radius;
    }

    // Method to calculate the perimeter (circumference) of the circle
    public double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }
}
```

```java
import Geometry.Circle;

public class Demo {
    public static void main(String[] args) {
        // Create a Circle object
        Circle circle = new Circle();

        // Set the radius of the circle
        circle.setRadius(5.0);

        // Calculate and display the area of the circle
        double area = circle.calculateArea();
        System.out.println("Area of the circle: " + area);

        // Calculate and display the perimeter of the circle
        double perimeter = circle.calculatePerimeter();
        System.out.println("Perimeter of the circle: " + perimeter);
    }
}
```

6) Implement a Triangle class having three data members a,b,c as it's sides. This class includes appropriate constructors and two methods

(1) Find_area : is used to return the area of the triangle.

(2) Find_perimeter : is used to return the perimeter of the triangle.

Implement an user defined exception class NoTriangleFormException which is thrown when object is created without legal values(a+b>c and b+c>a and cta>b). Write down the necessary java code to test the functionality of Triangle class.

```
// Custom exception class for when a triangle cannot be formed
class NoTriangleFormException extends Exception {
    public NoTriangleFormException(String message) {
        super(message);
    }
}

// Triangle class
class Triangle {
    private double a, b, c;

    // Constructor to initialize sides of the triangle
    public Triangle(double a, double b, double c) throws
NoTriangleFormException {
        if (a + b <= c || b + c <= a || c + a <= b) {
            throw new NoTriangleFormException("These sides cannot form a
triangle.");
        }
        this.a = a;
        this.b = b;
        this.c = c;
    }

    // Method to calculate area of the triangle using Heron's formula
    public double findArea() {
        double s = (a + b + c) / 2;
        return Math.sqrt(s * (s - a) * (s - b) * (s - c));
```

```java
    }

    // Method to calculate perimeter of the triangle
    public double findPerimeter() {
        return a + b + c;
    }
}

// Test class to demonstrate Triangle functionality
public class TestTriangle {
    public static void main(String[] args) {
        try {
            // Creating a triangle
            Triangle triangle = new Triangle(3, 4, 5);

            // Displaying the area and perimeter of the triangle
            System.out.println("Area of the triangle: " +
triangle.findArea());
            System.out.println("Perimeter of the triangle: " +
triangle.findPerimeter());
        } catch (NoTriangleFormException e) {
            // Catching the custom exception if sides cannot form a triangle
            System.out.println("Exception caught: " + e.getMessage());
        }
    }
}
```

7) Write a program that checks whether a string is a valid password. The password rule:

1) A password must have at least six characters

2) A password consists of only uppercase letters and digits

3) A password must contain at least three digits

Write a program that prompts the user to enter a password and display whether the password is valid or not.

```java
import java.util.Scanner;

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

        System.out.println("Enter a password:");
        String password = scanner.nextLine();
        if (isValidPassword(password)) {
            System.out.println("The password is valid.");
        } else {
            System.out.println("The password is not valid.");
        }

        scanner.close();
    }

    public static boolean isValidPassword(String password) {
        // Rule 1: A password must have at least six characters
        if (password.length() < 6) {
            return false;
        }

        // Rule 2: A password consists of only uppercase letters and digits
        for (int i = 0; i < password.length(); i++) {
            char ch = password.charAt(i);
            if (!Character.isUpperCase(ch) && !Character.isDigit(ch)) {
                return false;
            }
        }

        // Rule 3: A password must contain at least three digits
        int digitCount = 0;
        for (int i = 0; i < password.length(); i++) {
            char ch = password.charAt(i);
            if (Character.isDigit(ch)) {
                digitCount++;
            }
        }
        if (digitCount < 3) {
            return false;
        }

        // All rules passed, password is valid
        return true;
    }
}
```

8)Define Checked and Unchecked Exceptions in Java? Write a program in java which will accept the mark of a student from the keyboard. If the mark is less than zero or greater than 100, then the program will throw an user defined exception and catch the same using necessary catch handler and display the necessary message, otherwise the accepted mark will be displayed.

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

```java
class InvalidMarkException extends Exception {
    public InvalidMarkException(String message) {
        super(message);
    }
}

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

        try {
            System.out.print("Enter the mark of the student: ");
            int mark = scanner.nextInt();
            if (mark < 0 || mark > 100) {
                throw new InvalidMarkException("Invalid mark! Mark should be
between 0 and 100.");
            }
            System.out.println("Entered mark: " + mark);
        } catch (InvalidMarkException e) {
            System.out.println("Exception: " + e.getMessage());
        }

        scanner.close();
    }
}
```

9)Write a program in java having a class A. The class A will have two child classes called B and C. The class A and B will be in package pl and class C will be in package p2. Class A has four variables with different access specifiers. Illustrate their visibility in child class with suitable example.

```java
package pl;

public class A {
    public int publicVar = 10;
    protected int protectedVar = 20;
    int defaultVar = 30;
    private int privateVar = 40;
}

package pl;

public class B extends A {
    public void display() {
        System.out.println("Public variable in class B: " + publicVar);
        System.out.println("Protected variable in class B: " + protectedVar);
        System.out.println("Default variable in class B: " + defaultVar);

    }
}

package p2;

import p1.A;

public class C {
    public void display() {
        A a = new A();
        System.out.println("Public variable in class C: " + a.publicVar);
        System.out.println("Protected variable in class C: " +
a.protectedVar);

    }
}

public class VisibilityDemo {
    public static void main(String[] args) {
        B b = new B();
        b.display();

        C c = new C();
        c.display();
    }
}
```

10) Write a program to calculate the percentage of marks obtained in three subjects (each out of 100) by student A and in four subjects (each out of 100) by student B. Create an abstract class 'Marks' with an abstract method

'getPercentage'. It is inherited by two other classes 'A' and 'B' each having a method with the same name which returns the percentage,- of the students. The constructor of student A takes the marks in three subjects as its parameters and the marks in four subjects as its parameters for student B. Create an object for each of the two classes and print the percentage of marks for both the students.

```
// Abstract class Marks
abstract class Marks {
    abstract double getPercentage();
}

// Class A inherits Marks
class A extends Marks {
    private int subject1, subject2, subject3;
```

```java
    // Constructor for student A
    public A(int subject1, int subject2, int subject3) {
        this.subject1 = subject1;
        this.subject2 = subject2;
        this.subject3 = subject3;
    }

    // Method to calculate percentage for student A
    public double getPercentage() {
        return (subject1 + subject2 + subject3) / 3.0;
    }
}

// Class B inherits Marks
class B extends Marks {
    private int subject1, subject2, subject3, subject4;

    // Constructor for student B
    public B(int subject1, int subject2, int subject3, int subject4) {
        this.subject1 = subject1;
        this.subject2 = subject2;
        this.subject3 = subject3;
        this.subject4 = subject4;
    }

    // Method to calculate percentage for student B
    public double getPercentage() {
        return (subject1 + subject2 + subject3 + subject4) / 4.0;
    }
}

public class TestMarks {
    public static void main(String[] args) {
        // Create object for student A and calculate percentage
        Marks studentA = new A(80, 85, 90);
        System.out.println("Percentage of student A: " +
studentA.getPercentage());

        // Create object for student B and calculate percentage
        Marks studentB = new B(75, 85, 90, 95);
        System.out.println("Percentage of student B: " +
studentB.getPercentage());
    }
}
```

11) Write a Java program to create user defined exception for Negative ValueException if user enters negative value. In the program, check if number is negative then throw mentioned user defined exception.

Also catch that exception to display information of the raised exception using pre-defined method and finally display Exception successfully handled.

```
// User-defined exception class NegativeValueException
class NegativeValueException extends Exception {
    public NegativeValueException(String message) {
        super(message);
    }
}

public class UserDefinedExceptionExample {
    public static void main(String[] args) {
        try {
            int number = -5;

            // Check if number is negative
            if (number < 0) {
                throw new NegativeValueException("Negative value entered: " +
number);
            }

            System.out.println("No exception occurred.");
        } catch (NegativeValueException e) {
            // Catching the custom exception and displaying its information
            System.out.println("Exception occurred: " + e.getMessage());
        } finally {
            // Finally block to execute whether exception occurred or not
            System.out.println("Exception successfully handled.");
        }
    }
}
```

12) Create a swing application having 3 label RollNo, Name and Marks and three text field. The application will also have a Insert Button. When the user will enter value in all three text field and click on the Insert Button, the corresponding values will be submitted to the database table.

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

public class StudentDatabaseApp extends JFrame implements ActionListener {
    private JLabel rollNoLabel, nameLabel, marksLabel;
    private JTextField rollNoField, nameField, marksField;
    private JButton insertButton;

    // 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 StudentDatabaseApp() {
        setTitle("Student Database Application");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 200);
        setLayout(new GridLayout(4, 2));

        rollNoLabel = new JLabel("Roll No.: ");
        nameLabel = new JLabel("Name: ");
        marksLabel = new JLabel("Marks: ");

        rollNoField = new JTextField();
        nameField = new JTextField();
        marksField = new JTextField();

        insertButton = new JButton("Insert");
        insertButton.addActionListener(this);

        add(rollNoLabel);
        add(rollNoField);
        add(nameLabel);
        add(nameField);
        add(marksLabel);
        add(marksField);
        add(new JLabel()); // Placeholder for alignment
        add(insertButton);

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == insertButton) {
            String rollNo = rollNoField.getText();
            String name = nameField.getText();
            String marks = marksField.getText();

            try {
                // Create a Connection to the database
                Connection connection = DriverManager.getConnection(JDBC_URL,
USERNAME, PASSWORD);

                // Create a Statement object
                Statement statement = connection.createStatement();

                // Insert data into the table
                String query = "INSERT INTO students (roll_no, name, marks)
VALUES ('" + rollNo + "', '" + name + "', '" + marks + "')";
                statement.executeUpdate(query);

                JOptionPane.showMessageDialog(this, "Data inserted
successfully!");

                // Close the statement and connection
                statement.close();
                connection.close();
            } catch (SQLException ex) {
                JOptionPane.showMessageDialog(this, "Error inserting data into
database: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                ex.printStackTrace();
            }
        }
    }

    public static void main(String[] args) {
        // Create and display the Swing application
        SwingUtilities.invokeLater(() -> new StudentDatabaseApp());
    }
}
```

13) Create a swing application to calculate the Factorial and square of a given number. Create a text field to take the number as user input for these operations. Create two buttons namely FACTORIAL and SQUARE. When the user will click on any button, the result will be displayed on another text field.

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

public class FactorialAndSquareCalculator extends JFrame implements
ActionListener {
    private JTextField inputField, resultField;
    private JButton factorialButton, squareButton;

    public FactorialAndSquareCalculator() {
        setTitle("Factorial and Square Calculator");
```

```java
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 150);
        setLayout(new GridLayout(3, 2));

        JLabel inputLabel = new JLabel("Enter a number:");
        inputField = new JTextField();
        JLabel resultLabel = new JLabel("Result:");
        resultField = new JTextField();
        resultField.setEditable(false);

        factorialButton = new JButton("FACTORIAL");
        factorialButton.addActionListener(this);
        squareButton = new JButton("SQUARE");
        squareButton.addActionListener(this);

        add(inputLabel);
        add(inputField);
        add(resultLabel);
        add(resultField);
        add(factorialButton);
        add(squareButton);

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            int number = Integer.parseInt(inputField.getText());
            if (e.getSource() == factorialButton) {
                long factorial = calculateFactorial(number);
                resultField.setText(Long.toString(factorial));
            } else if (e.getSource() == squareButton) {
                int square = calculateSquare(number);
                resultField.setText(Integer.toString(square));
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Please enter a valid
number.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    private long calculateFactorial(int n) {
        if (n < 0) {
            throw new IllegalArgumentException("Factorial is not defined for
negative numbers.");
        }
        if (n == 0 || n == 1) {
            return 1;
        }
        long factorial = 1;
        for (int i = 2; i <= n; i++) {
            factorial *= i;
        }
        return factorial;
    }

    private int calculateSquare(int n) {
        return n * n;
    }

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

14) Create a class Rectangle having data members as length and width and a method as find Area). Derive a class Box from rectangle which will have a data member as height and a method called find Volume(). All the field must be initialize through constructor. Display the result by calling the find Area() method through the object of both the classes and the volume method through the object of Box class.

```
// Rectangle class
class Rectangle {
    protected double length;
    protected double width;

    // Constructor to initialize length and width
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Method to find area of rectangle
    public double findArea() {
        return length * width;
    }
}

// Box class derived from Rectangle
class Box extends Rectangle {
    private double height;

    // Constructor to initialize length, width, and height
    public Box(double length, double width, double height) {
        super(length, width);
        this.height = height;
    }

    // Method to find volume of box
    public double findVolume() {
        return length * width * height;
    }
}

public class TestRectangleAndBox {
    public static void main(String[] args) {
        // Create object of Rectangle class
        Rectangle rectangle = new Rectangle(5, 4);
        // Call findArea method of Rectangle class
        System.out.println("Area of Rectangle: " + rectangle.findArea());

        // Create object of Box class
        Box box = new Box(5, 4, 3);
        // Call findArea method of Rectangle class (inherited)
        System.out.println("Area of Box: " + box.findArea());
        // Call findVolume method of Box class
        System.out.println("Volume of Box: " + box.findVolume());
    }
}
```

15) Write a Java program to create a simple banking system using inheritance. Define a base class called Bank with a protected data member ROI (Rate of Interest) and a method rateOfInterest() to display the rate of interest. Derive three classes, HDFC, SBI, and Kotak, from the Bank class. The HDFC, SBI, and Kotak classes should have constructors that accept the rate of interest as a parameter and set the ROI accordingly. In the TestBank class, create objects for HDFC, SBI, and Kotak banks. Prompt the user to enter the rate of interest for each bank and display the rate of interest for all three banks.

```
import java.util.Scanner;

// Bank class
class Bank {
    protected double ROI; // Rate of Interest

    public void rateOfInterest() {
        System.out.println("Rate of Interest: " + ROI);
    }
}

// Derived classes
```

```java
class HDFC extends Bank {
    public HDFC(double ROI) {
        this.ROI = ROI;
    }
}

class SBI extends Bank {
    public SBI(double ROI) {
        this.ROI = ROI;
    }
}

class Kotak extends Bank {
    public Kotak(double ROI) {
        this.ROI = ROI;
    }
}

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

        System.out.print("Enter rate of interest for HDFC Bank: ");
        double hdfcROI = scanner.nextDouble();
        HDFC hdfc = new HDFC(hdfcROI);

        System.out.print("Enter rate of interest for SBI Bank: ");
        double sbiROI = scanner.nextDouble();
        SBI sbi = new SBI(sbiROI);

        System.out.print("Enter rate of interest for Kotak Bank: ");
        double kotakROI = scanner.nextDouble();
        Kotak kotak = new Kotak(kotakROI);

        System.out.println("\nRate of interest for different banks:");
        hdfc.rateOfInterest();
        sbi.rateOfInterest();
        kotak.rateOfInterest();

        scanner.close();
    }
}
```

16) Given two strings, A and B, create a bigger string made of the first char of A, the first char of B, the second char of A, the second char of B, and so on. Any leftover chars go at the end of the result. If the inputs of A and B strings are "Hello" and "World", then the output is "HWeolrlod".

```java
public class MergeStrings {
    public static void main(String[] args) {
        String A = "Hello";
        String B = "World";

        String mergedString = mergeStrings(A, B);
        System.out.println("Merged String: " + mergedString);
    }

    public static String mergeStrings(String A, String B) {
        StringBuilder result = new StringBuilder();
        int i = 0, j = 0;

        // Iterate over both strings simultaneously
        while (i < A.length() && j < B.length()) {
            result.append(A.charAt(i++)); // Append character from A
            result.append(B.charAt(j++)); // Append character from B
        }

        // Append remaining characters of A, if any
        while (i < A.length()) {
            result.append(A.charAt(i++));
        }

        // Append remaining characters of B, if any
        while (j < B.length()) {
            result.append(B.charAt(j++));
        }

        return result.toString();
    }
}
```

17) Write a Java program that performs operations on complex numbers. Define a class ComplexNumber with private data members real and imaginary, and methods to perform addition and subtraction of complex numbers. The ComplexNumber class should have a constructor to initialize the real and imaginary parts of a complex number, and methods add and subtract to perform addition and subtraction, respectively. Additionally, provide a method display to display the result.

```java
import java.util.Scanner;

// ComplexNumber class
class ComplexNumber {
    private double real;
    private double imaginary;
    // Constructor to initialize real and imaginary parts
    public ComplexNumber(double real, double imaginary) {
        this.real = real;
        this.imaginary = imaginary;
    }

    // Method to perform addition of two complex numbers
    public ComplexNumber add(ComplexNumber other) {
        double realResult = this.real + other.real;
        double imaginaryResult = this.imaginary + other.imaginary;
        return new ComplexNumber(realResult, imaginaryResult);
    }

    // Method to perform subtraction of two complex numbers
    public ComplexNumber subtract(ComplexNumber other) {
        double realResult = this.real - other.real;
        double imaginaryResult = this.imaginary - other.imaginary;
        return new ComplexNumber(realResult, imaginaryResult);
    }

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

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

        System.out.println("Enter the first complex number:");
        System.out.print("Real part: ");
        double real1 = scanner.nextDouble();
        System.out.print("Imaginary part: ");
        double imaginary1 = scanner.nextDouble();

        System.out.println("\nEnter the second complex number:");
        System.out.print("Real part: ");
        double real2 = scanner.nextDouble();
        System.out.print("Imaginary part: ");
        double imaginary2 = scanner.nextDouble();

        ComplexNumber complex1 = new ComplexNumber(real1, imaginary1);
        ComplexNumber complex2 = new ComplexNumber(real2, imaginary2);

        System.out.println("\nEnter the operation to perform 
(addition/subtraction):");
        String operation = scanner.next();
        ComplexNumber result;
        if (operation.equalsIgnoreCase("addition")) {
            result = complex1.add(complex2);
        } else if (operation.equalsIgnoreCase("subtraction")) {
            result = complex1.subtract(complex2);
        } else {
            System.out.println("Invalid operation!");
            return;
        }

        System.out.println("\nResult of " + operation + ":");
        result.display();

        scanner.close();
    }
}
```

18) Write a program in java to create a Frame with three labels (Enter X, Enter Y and Result), three text boxes with default message(enter x value, enter y value and result) and three buttons subtraction multiply and division. When you ent x value, y value in text boxes and press on the buttons their respective result should be displayed in the result text box.

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

public class CalculatorFrame extends JFrame implements ActionListener {
    private JLabel xLabel, yLabel, resultLabel;
    private JTextField xField, yField, resultField;
    private JButton subtractionButton, multiplyButton, divisionButton;

    public CalculatorFrame() {
        setTitle("Simple Calculator");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 200);
        setLayout(new GridLayout(4, 2));

        // Labels
        xLabel = new JLabel("Enter X: ");
        yLabel = new JLabel("Enter Y: ");
        resultLabel = new JLabel("Result: ");

        // Text Fields
        xField = new JTextField("Enter x value");
        yField = new JTextField("Enter y value");
        resultField = new JTextField("Result");
```

```java
        resultField.setEditable(false); // Make result text field non-editable

        // Buttons
        subtractionButton = new JButton("Subtraction");
        multiplyButton = new JButton("Multiply");
        divisionButton = new JButton("Division");

        // Action Listeners for Buttons
        subtractionButton.addActionListener(this);
        multiplyButton.addActionListener(this);
        divisionButton.addActionListener(this);

        // Add components to frame
        add(xLabel);
        add(xField);
        add(yLabel);
        add(yField);
        add(resultLabel);
        add(resultField);
        add(subtractionButton);
        add(multiplyButton);
        add(divisionButton);

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            double x = Double.parseDouble(xField.getText());
            double y = Double.parseDouble(yField.getText());

            if (e.getSource() == subtractionButton) {
                double result = x - y;
                resultField.setText(String.valueOf(result));
            } else if (e.getSource() == multiplyButton) {
                double result = x * y;
                resultField.setText(String.valueOf(result));
            } else if (e.getSource() == divisionButton) {
                if (y == 0) {
                    resultField.setText("Cannot divide by zero");
                } else {
                    double result = x / y;
                    resultField.setText(String.valueOf(result));
                }
            }
        } catch (NumberFormatException ex) {
            resultField.setText("Invalid input");
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(CalculatorFrame::new);
    }
}
```