```
/*PRO 1.5 Write a java application that will take two different
String objects and perform different operations on them like
checking the equality of two strings, reverse the string,
change case etc.*/

import java.util.Scanner;

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

        // Input two different strings
        System.out.println("Enter the first string:");
        String str1 = scanner.nextLine();

        System.out.println("Enter the second string:");
        String str2 = scanner.nextLine();

        // Perform operations
        System.out.println("Operations on String 1:");
        performOperations(str1);

        System.out.println("\nOperations on String 2:");
        performOperations(str2);

        // Check equality of strings
        boolean isEqual = str1.equals(str2);
```

```
        System.out.println("\nAre the strings equal? " +
isEqual);

        // Concatenate the strings
        String concatenated = str1.concat(str2);
        System.out.println("Concatenated string: " +
concatenated);

        scanner.close();
    }

    // Method to perform operations on a string
    public static void performOperations(String str) {
        System.out.println("Length of the string: " +
str.length());
        System.out.println("String in uppercase: " +
str.toUpperCase());
        System.out.println("String in lowercase: " +
str.toLowerCase());
        System.out.println("Reversed string: " +
reverseString(str));
    }

    // Method to reverse a string
    public static String reverseString(String str) {
        StringBuilder reversed = new StringBuilder();
        for (int i = str.length() - 1; i >= 0; i--) {
            reversed.append(str.charAt(i));
        }
```

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


 /*PRO2.1 Define a class called Student. Each Student has a
 rollno, name, marks and percentage.
  Define variables, methods and constructor for the Student
 class. Also write a class called TestStudent;
  with the main method to test the methods and constructors of
the Student class.*/


import java.util.Scanner;


class Student {
    private int rollNo;
    private String name;
    private double[] marks;
    private double percentage;

    // Constructor to initialize the Student object
    public Student(int rollNo, String name, double[] marks) {
        this.rollNo = rollNo;
        this.name = name;
        this.marks = marks;
        calculatePercentage();
    }

    // Method to calculate the percentage
```

```
    private void calculatePercentage() {
        double total = 0;
        for (double mark : marks) {
            total += mark;
        }
        this.percentage = total / marks.length;
    }

    // Method to display student details
    public void displayStudentDetails() {
        System.out.println("Roll No: " + rollNo);
        System.out.println("Name: " + name);
        System.out.println("Marks: ");
        for (int i = 0; i < marks.length; i++) {
            System.out.println("Subject " + (i + 1) + ": " +
marks[i]);
        }
        System.out.println("Percentage: " + percentage + "%");
    }
}

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

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

```
        for (int i = 0; i < numStudents; i++)
{
            System.out.println("Enter details for student " + (i
+ 1) + ":");
            System.out.print("Roll No: ");
            int rollNo = scanner.nextInt();
            scanner.nextLine(); // Consume newline character
            System.out.print("Name: ");
            String name = scanner.nextLine();
            System.out.print("Number of subjects: ");
            int numSubjects = scanner.nextInt();
            double[] marks = new double[numSubjects];
            for (int j = 0; j < numSubjects; j++) {
                System.out.print("Enter marks for Subject " + (j
+ 1) + ": ");
                marks[j] = scanner.nextDouble();
            }
            Student student = new Student(rollNo, name, marks);
            student.displayStudentDetails();
            System.out.println();
        }

        scanner.close();
    }
}

/*PRO 2.2 Define a class called Cartesian Point, which has two
instance variables, x and y.
```

```
  Provide the methods get X() and getY() to return the values of
the x and y values respectively,
  a method called move() which would take two integers as
parameters and change the values of x and y
  respectively. A method called display() which would display
the current values of x and y.
  Now overload the method move() to work with a single
parameter, which would set both x and y to the same values.
  Provide constructors with two parameters and overload to work
with one parameter as well.
  Now define a class called Test Cartesian Point, with the main
method to test the various methods in the Cartesian Point
class.*/

class CartesianPoint {
    private int x;
    private int y;

    // Constructor with two parameters
    public CartesianPoint(int x, int y) {
        this.x = x;
        this.y = y;
    }

    // Constructor with one parameter (overloaded)
    public CartesianPoint(int xy) {
        this.x = xy;
        this.y = xy;
    }
```

```
    // Method to get x value
    public int getX() {
        return x;
    }

    // Method to get y value
    public int getY() {
        return y;
    }

    // Method to move point by given x and y values
    public void move(int newX, int newY) {
        x = newX;
        y = newY;
    }

    // Overloaded move method with single parameter to set both
x and y to the same value
    public void move(int newXY) {
        x = newXY;
        y = newXY;
    }

    // Method to display current values of x and y
    public void display() {
        System.out.println("Current coordinates: (" + x + ", " +
y + ")");
```

```
    }
}

public class TestCartesianPoint {
    public static void main(String[] args) {
        // Test with constructor taking two parameters
        CartesianPoint point1 = new CartesianPoint(3, 5);
        System.out.println("Point 1:");
        point1.display();

        // Test with constructor taking one parameter
        CartesianPoint point2 = new CartesianPoint(2);
        System.out.println("\nPoint 2:");
        point2.display();

        // Test move method with two parameters
        System.out.println("\nMoving Point 1 to (7, 9)");
        point1.move(7, 9);
        point1.display();

        // Test move method with one parameter
        System.out.println("\nMoving Point 2 to (10, 10)");
        point2.move(10);
        point2.display();
    }
}
 /*PRO 2.3 write a java program An educational institution
wishes to maintain a database of its employees.
```

```
 The database is divided into a number of classes whose
hierarchical relationships are as follows:
 Staff (code, name): the base class
 Teacher (subject, publication):child class of the Staff
 Officer (grade): child class of the Staff.
 Note that the information given in brackets specifies the
minimum information required for each class.
 Specify all classes and define functions and constructors to
create the database and retrieve information as and when
needed.*/

import java.util.ArrayList;

// Base class Staff
class Staff {
    private int code;
    private String name;

    // Constructor for Staff class
    public Staff(int code, String name) {
        this.code = code;
        this.name = name;
    }

    // Getters
    public int getCode() {
        return code;
    }
```

```
    public String getName() {
        return name;
    }
}

// Child class Teacher
class Teacher extends Staff {
    private String subject;
    private String publication;

    // Constructor for Teacher class
    public Teacher(int code, String name, String subject, String
publication) {
        super(code, name);
        this.subject = subject;
        this.publication = publication;
    }

    // Getters
    public String getSubject() {
        return subject;
    }

    public String getPublication() {
        return publication;
    }
}
```

```
// Child class Officer
class Officer extends Staff {
    private String grade;

    // Constructor for Officer class
    public Officer(int code, String name, String grade) {
        super(code, name);
        this.grade = grade;
    }

    // Getter
    public String getGrade() {
        return grade;
    }
}

class EmployeeDatabase {
    private ArrayList<Staff> staffList;

    // Constructor for EmployeeDatabase
    public EmployeeDatabase() {
        staffList = new ArrayList<Staff>();
    }

    // Method to add staff to the database
    public void addStaff(Staff staff) {
```

```
        staffList.add(staff);
    }

    // Method to retrieve information about all staff
    public void displayStaffInfo() {
        System.out.println("Employee Database:");
        for (Staff staff : staffList) {
            if (staff instanceof Teacher) {
                Teacher teacher = (Teacher) staff;
                System.out.println("Teacher - Code: " +
teacher.getCode() + ", Name: " + teacher.getName() +
                        ", Subject: " + teacher.getSubject() +
", Publication: " + teacher.getPublication());
            } else if (staff instanceof Officer) {
                Officer officer = (Officer) staff;
                System.out.println("Officer - Code: " +
officer.getCode() + ", Name: " + officer.getName() +
                        ", Grade: " + officer.getGrade());
            }
        }
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating EmployeeDatabase object
        EmployeeDatabase database = new EmployeeDatabase();
```

```
        // Adding staff to the database
        Teacher teacher1 = new Teacher(101, "John",
"Mathematics", "Mathematics Textbook");
        Teacher teacher2 = new Teacher(102, "Alice", "Science",
"Science Journal");
        Officer officer1 = new Officer(201, "Bob", "Grade A");
        Officer officer2 = new Officer(202, "Emily", "Grade B");

        database.addStaff(teacher1);
        database.addStaff(teacher2);
        database.addStaff(officer1);
        database.addStaff(officer2);

        // Displaying information about all staff
        database.displayStaffInfo();
    }
}
/* PRO 2.4 Write a java  program that demonstrates the concept
of abstract methods and class.*/

// Abstract class
abstract class Shape {
    // Abstract method (does not have a body)
    public abstract double calculateArea();

    // Concrete method
    public void display() {
        System.out.println("This is a shape.");
```

```
    }
}

// Concrete subclass Circle
class Circle extends Shape {
    private double radius;

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

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

// Concrete subclass Rectangle
class Rectangle extends Shape {
    private double length;
    private double width;

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

```
    // Implementing abstract method Override
    public double calculateArea() {
        return length * width;
    }
}

public class Main1 {
    public static void main(String[] args) {
        // Cannot instantiate an abstract class directly
        // Shape shape = new Shape(); // This will result in
compilation error

        // Creating instances of concrete subclasses
        Circle circle = new Circle(5);
        Rectangle rectangle = new Rectangle(4, 6);

        // Polymorphism: Using Shape reference to refer to
Circle object
        Shape shape1 = circle;
        System.out.println("Area of Circle: " +
shape1.calculateArea());

        // Polymorphism: Using Shape reference to refer to
Rectangle object
        Shape shape2 = rectangle;
        System.out.println("Area of Rectangle: " +
shape2.calculateArea());
```

```
        // Accessing concrete method
        shape1.display();
        shape2.display();
    }
}
```

/*PRO2.5 Write a program that imports the user defined package and access the members of the classes that are contained by the package.*/

```
 // MyClass1.java
package myPackage;

public class MyClass1 {
    public void method1() {
        System.out.println("Method 1 from MyClass1");
    }
}
  // MyClass2.java
package myPackage;

public class MyClass2 {
    public void method2() {
        System.out.println("Method 2 from MyClass2");
    }
}
```

```
  // Main2.java
import myPackage.MyClass1;
import myPackage.MyClass2;

public class Main2 {
    public static void main(String[] args) {
        // Creating objects of classes from the imported package
        MyClass1 obj1 = new MyClass1();
        MyClass2 obj2 = new MyClass2();

        // Accessing methods from MyClass1 and MyClass2
        obj1.method1();
        obj2.method2();
    }
}
```

```
 /*PRO 2.6 Write a java application that takes the current date
from the system and performs the following.
Display the date in format like "dd-MM-yy".Apply comparison of
two dates.*/



import java.text.SimpleDateFormat;
import java.util.Date;

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

```
        // Get current date
        Date currentDate = new Date();

        // Display current date in format "dd-MM-yy"
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd-
MM-yy");
        String formattedDate = dateFormat.format(currentDate);
        System.out.println("Current Date: " + formattedDate);

        // Create another date for comparison (let's say 10 days
from now)
        long millisecondsInADay = 1000 * 60 * 60 * 24;
        Date futureDate = new Date(currentDate.getTime() + 10 *
millisecondsInADay);

        // Display the future date in the same format
        String formattedFutureDate =
dateFormat.format(futureDate);
        System.out.println("Future Date: " +
formattedFutureDate);

        // Compare the two dates
        int comparisonResult =
currentDate.compareTo(futureDate);
        if (comparisonResult < 0) {
            System.out.println("Current Date is before Future
Date.");
        } else if (comparisonResult > 0) {
```

```
            System.out.println("Current Date is after Future
Date.");
        } else {
            System.out.println("Current Date is equal to Future
Date.");
        }
    }
}
 /*PRO 2.7 Write a java program that demonstrates the concept of
Vector class constructors and its all methods.*/
import java.util.Vector;

public class VectorExample {
    public static void main(String[] args) {
        // Creating a Vector using the default constructor
        Vector<String> vector1 = new Vector<>();

        // Adding elements to the Vector using add() method
        vector1.add("Apple");
        vector1.add("Banana");
        vector1.add("Orange");
        vector1.add("Grapes");

        // Displaying the Vector elements
        System.out.println("Vector 1: " + vector1);

        // Creating a Vector using the constructor with initial
capacity
```

```
        Vector<Integer> vector2 = new Vector<>(5);

        // Adding elements to the Vector using addElement()
method
        vector2.addElement(10);
        vector2.addElement(20);
        vector2.addElement(30);
        vector2.addElement(40);
        vector2.addElement(50);

        // Displaying the Vector elements
        System.out.println("Vector 2: " + vector2);

        // Creating a Vector using the constructor with initial
capacity and capacity increment
        Vector<Double> vector3 = new Vector<>(3, 2);

        // Adding elements to the Vector using addElement()
method
        vector3.addElement(3.14);
        vector3.addElement(6.28);

        // Displaying the Vector elements
        System.out.println("Vector 3: " + vector3);

        // Accessing elements using elementAt() method
        System.out.println("Element at index 1 in Vector 1: " +
vector1.elementAt(1));
```

```
        // Removing elements using removeElementAt() method
        vector1.removeElementAt(2);
        System.out.println("Vector 1 after removing element at
index 2: " + vector1);


        // Getting the size of the Vector using size() method
        System.out.println("Size of Vector 2: " +
vector2.size());


        // Checking if the Vector is empty using isEmpty()
method
        System.out.println("Is Vector 3 empty? " +
vector3.isEmpty());


        // Getting the index of an element using indexOf()
method
        System.out.println("Index of element 30 in Vector 2: " +
vector2.indexOf(30));


        // Checking if an element is present using contains()
method
        System.out.println("Does Vector 3 contain 3.14? " +
vector3.contains(3.14));


        // Setting an element at a specific index using set()
method
        vector2.set(3, 45);
        System.out.println("Vector 2 after setting element at
index 3 to 45: " + vector2);
```

```
        // Clearing the Vector using clear() method
        vector3.clear();
        System.out.println("Vector 3 after clearing: " +
vector3);
    }
}
```