Title: Create and demonstrate programs using class, object and methods.

Create a Java program to model a simple "Book" class with the following properties:

- ISBN (String)
- Title (String)
- Author (String)
- Price (double)

Include methods in the class to set and get each property, and also a method to display the details of the book. Then, create three objects of the class, set values for their properties, and display the details.

```
// Book class definition
class Book {
    // Properties
    private String isbn;
    private String title;
    private String author;
    private double price;

    // Methods to set values
    public void setIsbn(String bookIsbn) {
        this.isbn = bookIsbn;
    }

    public void setTitle(String bookTitle) {
        this.title = bookTitle;
    }

    public void setAuthor(String bookAuthor) {
        this.author = bookAuthor;
    }

    public void setPrice(double bookPrice) {
        this.price = bookPrice;
    }

    // Methods to get values
    public String getIsbn() {
        return this.isbn;
    }

    public String getTitle() {
        return this.title;
    }

    public String getAuthor() {
        return this.author;
    }

    public double getPrice() {
        return this.price;
    }

    // Method to display book details
    public void displayDetails() {
        System.out.println("ISBN: " + this.isbn);
        System.out.println("Title: " + this.title);
        System.out.println("Author: " + this.author);
        System.out.println("Price: $" + this.price);
    }
}

// Main class for demonstration
public class BookDemo {
    public static void main(String[] args) {
        // Creating three objects of Book class
        Book book1 = new Book();
        Book book2 = new Book();
        Book book3 = new Book();

        // Setting values for the properties of each book
        book1.setIsbn("978-3-16-148410-0");
        book1.setTitle("Introduction to Java Programming");
        book1.setAuthor("John Doe");
        book1.setPrice(49.99);

        book2.setIsbn("978-1-23-456789-0");
        book2.setTitle("Data Structures and Algorithms");
        book2.setAuthor("Jane Smith");
        book2.setPrice(39.95);

        book3.setIsbn("978-9-87-654321-0");
        book3.setTitle("Object-Oriented Design Patterns");
        book3.setAuthor("Bob Johnson");
        book3.setPrice(59.99);

        // Displaying details for each book
        System.out.println("Book 1 Details:");
        book1.displayDetails();

        System.out.println("\nBook 2 Details:");
        book2.displayDetails();

        System.out.println("\nBook 3 Details:");
        book3.displayDetails();
    }
}
```

In this program, the Book class represents a book with properties such as ISBN, title, author, and price. Objects of the Book class are created, values are set for their properties, and then the details of each book are displayed using the displayDetails method.

Title: Create and demonstrate programs using inheritance.

Design a Java program to demonstrate the concept of inheritance by creating a base class named "Person" and a derived class named "Employee." The base class should have the following private properties:

- name (String)
- age (int)

Implement accessors (getters) and mutators (setters) using appropriate access specifiers for each property in the base class. The derived class should inherit from the base class and have an additional private property:

- employeeId (int)

Implement accessors and mutators for the employeeId property in the derived class. Create an object of the derived class, set values for the properties using the appropriate methods, and display the details using the accessor methods from both the base and derived classes.

Java Program:

```
java
// Person class definition with private properties and public methods
class Person {
    // Private properties
    private String name;
    private int age;

    // Public methods to set values using mutators (setters)
    public void setName(String name) {
        this.name = name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    // Public methods to get values using accessors (getters)
    public String getName() {
        return this.name;
    }

    public int getAge() {
        return this.age;
    }

    // Method to display person details
    public void displayDetails() {
        System.out.println("Name: " + this.name);
        System.out.println("Age: " + this.age);
    }
}

// Employee class inheriting from Person
class Employee extends Person {
    // Additional private property in the derived class
    private int employeeId;

    // Public methods to set values for the additional property using mutators (setters)
    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }

    // Public method to get the value of the additional property using accessor (getter)
    public int getEmployeeId() {
        return this.employeeId;
    }

    // Method to display employee details, including details from the base class
    public void displayEmployeeDetails() {
        displayDetails(); // Invoking the displayDetails method from the base class
        System.out.println("Employee ID: " + this.employeeId);
    }
}

// Main class for demonstration
public class InheritanceDemo {
    public static void main(String[] args) {
        // Creating an object of the derived class (Employee)
        Employee employee = new Employee();

        // Setting values for the properties of the person (base class) using mutators
        employee.setName("John Doe");
        employee.setAge(30);

        // Setting values for the additional property of the employee (derived class) using mutators
        employee.setEmployeeId(101);

        // Displaying details of the employee using accessors from both base and derived classes
        System.out.println("Employee Details:");
        employee.displayEmployeeDetails();
    }
}
```

In this program, the Person class is the base class with private properties for name and age. The Employee class is the derived class that inherits from Person and adds an additional private property for employeeId. The program demonstrates inheritance by creating an object of the derived class, setting values for properties, and displaying details using methods from both the base and derived classes.

Title: Create and demonstrate programs using method overloading and method overriding.

Question 1: Method Overloading

Design a Java program to illustrate method overloading. Create a class named "Calculator" with overloaded methods for addition. The class should have:

1. A method named add that takes two integers and prints their sum.
2. Another method named add that takes three doubles and prints their sum.
3. A third method named add that takes two strings and concatenates them.

Create an object of the "Calculator" class, invoke each add method, and observe the output.

Java Program for Question 1:

```
// Calculator class definition
class Calculator {
   // Method overloading for addition with two integers
   public void add(int num1, int num2) {
       int sum = num1 + num2;
       System.out.println("Sum of two integers: " + sum);
   }

   // Method overloading for addition with three doubles
   public void add(double num1, double num2, double num3) {
       double sum = num1 + num2 + num3;
       System.out.println("Sum of three doubles: " + sum);
   }

   // Method overloading for string concatenation
   public void add(String str1, String str2) {
       String result = str1 + str2;
       System.out.println("Concatenation of two strings: " + result);
   }
}

// Main class for demonstration
public class MethodOverloadingDemo {
   public static void main(String[] args) {
       // Creating an object of the Calculator class
       Calculator calculator = new Calculator();

       // Invoking each overloaded add method
       calculator.add(5, 7);
       calculator.add(3.5, 2.0, 1.5);
       calculator.add("Hello", "World");
   }
}
```

Question 2: Method Overriding

Design a Java program to illustrate method overriding. Create a base class named "Vehicle" with a method named startEngine(). Derive two classes, "Car" and "Motorcycle," from the base class.

1. In the "Vehicle" class, define a method startEngine() that prints "Starting the vehicle engine."
2. In the "Car" class, override the startEngine() method to print "Starting the car engine."
3. In the "Motorcycle" class, override the startEngine() method to print "Starting the motorcycle engine."

Create objects of both the "Car" and "Motorcycle" classes, invoke the startEngine() method for each, and observe the output.

Java Program for Question 2:

```
// Vehicle class definition
class Vehicle {
    // Method to start the engine for a generic vehicle
    public void startEngine() {
        System.out.println("Starting the vehicle engine.");
    }
}

// Car class inheriting from Vehicle
class Car extends Vehicle {
    // Overriding the startEngine() method for a car
    @Override
    public void startEngine() {
        System.out.println("Starting the car engine.");
    }
}

// Motorcycle class inheriting from Vehicle
class Motorcycle extends Vehicle {
    // Overriding the startEngine() method for a motorcycle
    @Override
    public void startEngine() {
        System.out.println("Starting the motorcycle engine.");
    }
}

// Main class for demonstration
public class MethodOverridingDemo {
    public static void main(String[] args) {
        // Creating objects of Car and Motorcycle classes
        Car myCar = new Car();
        Motorcycle myMotorcycle = new Motorcycle();

        // Invoking the startEngine() method for each vehicle
        myCar.startEngine();
        myMotorcycle.startEngine();
    }
}
```

These two programs demonstrate method overloading in the first question and method overriding in the second question in Java.