Question 1: Employee

Overview of the Task

You are asked to write a simple Java program that models an employee using a class. This exercise reinforces using instance variables and instance methods to store and display data. You'll create an **Employee** class that encapsulates employee details.

Detailed Breakdown

1. Class Definition

- **What to Do:**
  Create a public class named **Employee**.
- **Why:**
  The class acts as a blueprint for employee objects.

2. Instance Variables

- **Variables to Include:**
  - `employeeName` – to hold the employee's name.
  - `employeeId` – to hold the employee's unique identifier.
  - `employeeSalary` – to hold the employee's salary.
- **Why:**
  These variables represent the state of each **Employee** object. Each employee will have unique values for these attributes.

3. Instance Method: addEmployeeDetails

- **What to Do:**
  Define an instance method called **addEmployeeDetails** that accepts three parameters: one for the name, one for the ID, and one for the salary. Inside this method, assign the parameter values to the corresponding instance variables (`employeeName` , `employeeId` , `employeeSalary`).
- **Why:**
  This method sets or updates the state of an **Employee** object.

4. Instance Method: displayDetails

- **What to Do:**
  Create an instance method called **displayDetails** that returns a formatted string including `employeeName` , `employeeId` , and `employeeSalary` .
- **Why:**
  This method provides a way to retrieve and display the current state of the employee in a readable format.

5. Main Method

- **What to Do:**
  In the **main** method, instantiate an **Employee** object, call **addEmployeeDetails** to set the details, and then call **displayDetails** to print the employee's information.

- **Why:**
  This demonstrates the creation and use of an **Employee** object, tying together instantiation, state assignment, and state retrieval.

Question 2: Car

Overview of the Task

This task involves writing a Java program that models a car. You will focus on using instance variables and methods to manage and display the car's data.

Detailed Breakdown

1. Class Definition

- **What to Do:**
  Create a public class named **Car**.
- **Why:**
  This class serves as a blueprint for car objects.

2. Instance Variables

- **Variables to Include:**
  - `carMake` – to hold the car’s make (e.g., Toyota, Ford).
  - `carModel` – to hold the car’s model.
  - `manufacturingYear` – to hold the year the car was manufactured.
- **Why:**
  Each **Car** object uses these variables to store its unique properties.

3. Instance Method: addCarDetails

- **What to Do:**
  Define an instance method called **addCarDetails** that accepts parameters for `carMake`, `carModel`, and `manufacturingYear` and assigns these values to the respective instance variables.
- **Why:**
  This method initializes or updates the state of the **Car** object.

4. Instance Method: displayDetails

- **What to Do:**
  Implement an instance method called **displayDetails** that returns a string combining the values of `carMake`, `carModel`, and `manufacturingYear`.
- **Why:**
  This method displays the current state of the **Car** object.

5. Main Method

- **What to Do:**
  In the **main** method, create a **Car** object, call **addCarDetails** to set its properties, and then print the details by calling **displayDetails**.

- **Why:**
  This demonstrates how the **Car** object is used to set and retrieve its state.

Question 3: Book

Overview of the Task

In this exercise, you will create a Java program that models a book. The goal is to practice using instance variables and methods to manage the book's details.

Detailed Breakdown

1. Class Definition

- **What to Do:**
  Create a public class named **Book**.
- **Why:**
  The class serves as a blueprint for book objects.

2. Instance Variables

- **Variables to Include:**
  - `bookTitle` – to hold the title of the book.
  - `author` – to hold the author’s name.
  - `isbn` – to hold the book’s ISBN or unique identifier.
- **Why:**
  These variables capture the essential details of a book and form its state.

3. Instance Method: addBookDetails

- **What to Do:**
  Define an instance method called **addBookDetails** that accepts parameters for `bookTitle`, `author`, and `isbn`. Assign these parameter values to the corresponding instance variables.
- **Why:**
  This method initializes or updates the state of the **Book** object.

4. Instance Method: displayDetails

- **What to Do:**
  Implement an instance method called **displayDetails** that returns a string displaying the book’s details using `bookTitle`, `author`, and `isbn`.
- **Why:**
  This method provides a readable representation of the **Book** object's state.

5. Main Method

- **What to Do:**
  In the **main** method, instantiate a **Book** object, set its details using **addBookDetails**, and then output its details by calling **displayDetails**.

- **Why:**
  This brings together object creation, state assignment, and data retrieval for the **Book** class.

Question 4: Product

Overview of the Task

You are tasked with writing a simple Java program to model a product. This will reinforce your understanding of using instance variables and methods to manage and display an object's state.

Detailed Breakdown

1. Class Definition

- **What to Do:**
  Create a public class named **Product**.
- **Why:**
  The class defines the blueprint for **Product** objects.

2. Instance Variables

- **Variables to Include:**
  - `productName` – to hold the product’s name.
  - `productId` – to hold the product’s unique identifier.
  - `productPrice` – to hold the product’s price.
- **Why:**
  These variables represent the state of a **Product** object, storing its key attributes.

3. Instance Method: addProductDetails

- **What to Do:**
  Define an instance method called **addProductDetails** that takes parameters for `productName`, `productId`, and `productPrice` and assigns these values to the corresponding instance variables.
- **Why:**
  This method allows you to set or update the state of a **Product** object.

4. Instance Method: displayDetails

- **What to Do:**
  Implement an instance method called **displayDetails** that returns a string formatted to show the values of `productName`, `productId`, and `productPrice`.
- **Why:**
  This method retrieves and displays the **Product** object's state in a clear and readable format.

5. Main Method

- **What to Do:**
  In the **main** method, instantiate a **Product** object, use **addProductDetails** to initialize its state, and call **displayDetails** to output the product’s information.
- **Why:**
  This demonstrates the complete lifecycle of a **Product** object from creation to state retrieval.
