Practical Assignment Questions for Interfaces in Java

Assignment 1: Basic Interface Implementation

1. Create an Interface :

   - Define an interface named `Vehicle` with methods `start()`, `stop()`, and `getFuelLevel()`.

2. Implement the Interface :

   - Create a class `Car` that implements the `Vehicle` interface. Provide implementations for the `start()`, `stop()`, and `getFuelLevel()` methods.

3. Test the Implementation :

   - In your main method, create an instance of `Car` and call its methods. Verify the outputs.

Assignment 2: Default and Static Methods in Interfaces

1. Enhance the Interface :

   - Add a default method `getVehicleType()` to the `Vehicle` interface that returns a string "Unknown Vehicle".

   - Add a static method `serviceRequired()` to the `Vehicle` interface that returns a boolean indicating if the vehicle needs servicing.

2. Override the Default Method :

   - In the `Car` class, override the `getVehicleType()` method to return "Car".

3. Test the Methods :

   - In your main method, create an instance of `Car` and call the `getVehicleType()` and `serviceRequired()` methods. Verify the outputs.

Assignment 3: Private Methods in Interfaces

1. Add Private Methods :

   - Add a private method `log(String message)` to the `Vehicle` interface that prints a log message.

You Tube Playlist Link:

https://www.youtube.com/playlist?list=PLzrb6iZd6X9IOZHGVPWJqx0M5eEBvxcOV

- Create a default method `startWithLog()` in the `Vehicle` interface that calls `log("Vehicle started")` before calling the `start()` method.

2. Test the Private Method Usage :

- In your main method, create an instance of `Car` and call the `startWithLog()` method. Verify the output includes the log message and the start message.

Assignment 4: Interface Inheritance

1. Create a Sub-Interface :

- Define a sub-interface named `ElectricVehicle` that extends `Vehicle` and adds a method `chargeBattery()`.

2. Implement the Sub-Interface :

- Create a class `ElectricCar` that implements the `ElectricVehicle` interface. Provide implementations for all methods, including `chargeBattery()`.

3. Test the Inheritance :

- In your main method, create an instance of `ElectricCar` and call all its methods. Verify the outputs.

Assignment 5: Practical Application

1. Create a Complex Interface Structure :

- Define an interface `Gadget` with methods `powerOn()` and `powerOff()`.

- Define another interface `SmartDevice` that extends `Gadget` and adds methods `connectToWiFi()` and `disconnectFromWiFi()`.

2. Implement the Interfaces :

- Create a class `Smartphone` that implements the `SmartDevice` interface. Provide implementations for all methods.

3. Test the Complex Interface :

- In your main method, create an instance of `Smartphone` and call its methods. Verify the outputs.

You Tube Playlist Link:

https://www.youtube.com/playlist?list=PLzrb6iZd6X9IOZHGVPWJqx0M5eEBvxcOV

Assignment 6: Real-World Simulation

1. Define Interfaces for a Library System :

   - Create an interface `LibraryItem` with methods `checkOut()`, `returnItem()`, and `getDueDate()`.

   - Create a sub-interface `Book` that extends `LibraryItem` and adds a method `getAuthor()`.

   - Create another sub-interface `DVD` that extends `LibraryItem` and adds a method `getDirector()`.

2. Implement the Interfaces :

   - Create classes `LibraryBook` and `LibraryDVD` that implement `Book` and `DVD` respectively. Provide implementations for all methods.

3. Test the Library System :

   - In your main method, create instances of `LibraryBook` and `LibraryDVD`. Call their methods and verify the outputs.