MSSE670_Week3_Assignment

Thursday, September 1, 2022 3:57 PM

Week 3 Assignment, Services Layer Part 1

Important Notes

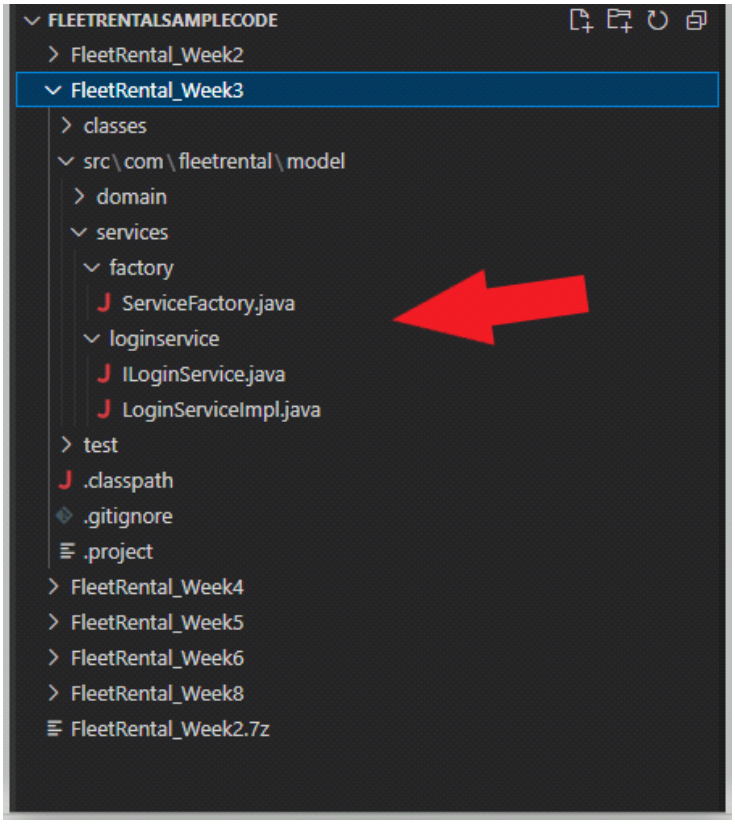
- In this course, we will not be using a real database so much of our code will function as stubs. As you start to implement the parts of the code that does something, you may focus on having your methods accept values and return values while printing information to the console showing things are working as expected.
- Do not take short cuts in the class. For example, even though we will replace code from Week 3 with code from Week 4 (the Factory), you must turn in the appropriate code for this week.

Design

Plan

- Refer to the **Fleet Rental Sample Code** for details about how to program your project.
- Add any classes that were missing from the previous week. Add the Services Layer to your Sequence Diagram.
- You can make a new project for Week3, but you may also continue developing the same project from the previous week. If you continue developing in the previous project, I recommend you use version control to be able to roll back to the previous week or previous versions of the code.
- This week you will begin creating your **Services Layer**.
- You will create packages that nest the layer in the following structure:

```
com
    <app-name>
        model
            domain
            services
```



Code

- Within your services layer, you will create the following

PACKAGE	CLASS	DESCRIPTION
factory	ServiceFactory	This is a simple service factory that is used to create the services. It will be replaced next week. You should understand how it works, but you should copy this directly.
loginservice	ILoginService LoginServiceImpl	• The login service is a simple service that is being used as an example to get you started. You may start by implementing this service as you would expect it to work. A primitive login service would query a database for a user and verify if the entered password matches the password in the database. In this service, there is only a Retrieve operation (no Create, Update, or Delete) so it's just a starting point. But for this class, you only have to print information to the console. • You should understand how we are using Interfaces and Implementations for our services. This is probably the most critical part of the class, and it builds from here. • Continue by implementing one more of the other services. A minimum the service should contain methods for all CRUD operations, but could contain more, including helper methods as needed. • So for example, the Reservation Service, would: 1. List all reservations 2. Make a new reservation 3. Update an existing reservation 4. Delete a reservation • But there could be other methods too, like: 1. Search for a reservation by customer id • When developing these services, think about handling errors. 1. Should you put logic in your code to handle common error conditions? 2. Should you let the Java try/except functionality handle errors 3. Should you do a combination of both? The answer to that is yes, but for now think about the kinds of things that could go wrong, e.g. deleting a reservation by id, but the id does not exist.

What to turn in

- You will implement your Services Layer following the details above. The should include:
  - Updated Design Document to include the Services Layer Sequence Diagram.
  - Appropriate naming conventions and style (see sample code).
  - Packages in the Services Layer, including the factory
  - Services using Interfaces and Implementations
  - **At least one fully implemented Service** in addition to the Login Service (must include 4 CRUD methods).
  - A Driver Class with a main method that runs the program. This goes above the Services Layer. I recommend putting it in the business layer for now.
  - Tests for the Services Layer.
  - A README with screen shots of a completed runs using the Driver as well as successful test runs.
  - Code uploaded to a source control repo and link submitted in slack. Code for Week 3 must be clearly labeled in a folder or branch.