MSSE670_Week2_Assignment

Thursday, September 1, 2022 3:57 PM

Week 2 Assignment

Design

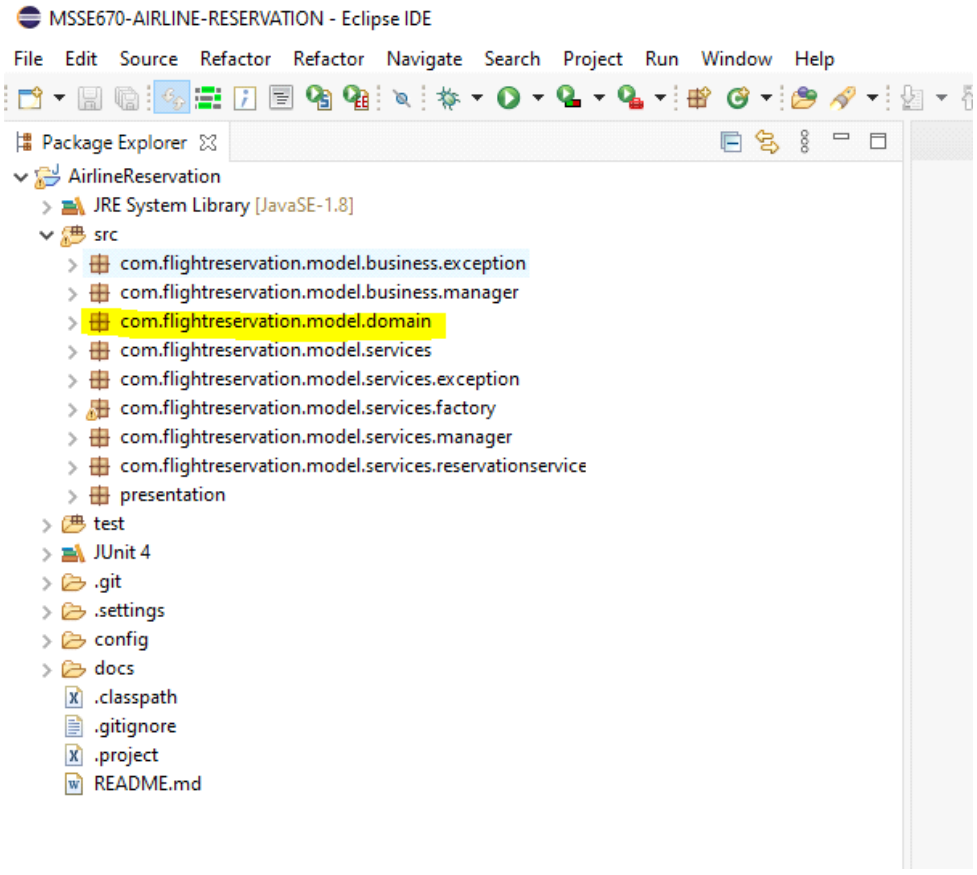
- Refer to the **Fleet Rental Sample Code** for details about how to program your application.
- This week the assignment is relatively easy. You will create your Domain layer, which is a package that contains all of you r domain classes. Refer to my lecture this week to understand how this fits into the big picture, Domain Driven Design (DDD).
- Create a UML model of three classes plus a composite class (refer to lecture this week). So you will have at least 4 classe s total. You can add classes iteratively over the course or try to create all of them this week.
- Create a diagram of the DDD Bounded Contexts for your application.
- Start a sequence diagram and add the Domain. For this, you just need to copy or copy and adapt the sequence diagram given i n Week 1. It will be the same for all projects since everyone is following the same architecture.

Code

- Create a Java Project in your IDE for your application. During the first weeks of the class, we will develop a new layer eac h week until we get to the part of the class, where we iterate and add new functionality.
- This week you will create your domain layer. You will create packages that nest the layer in the following structure:

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

Example in Eclipse



Example: Hiking Club Domain Model

- Your domain layer will contain at least three of the domain classes for the application plus a composite.

CLASS NAME	DESCRIPTION
Member	Could also be called a Customer. Could contain an address or reference an address object as a composite.
Address (Optional)	Could be included as part of the Member or treated as a composite.
Trip List	This is an "inventory-type" class that will contain a list of trip objects.
Trip	The Trip class will have attributes like Trip id, location, date, time, leader, etc.
Reservation	There might be differences in the design of the sample code and what I am showing. This could be modeled with an Itinerary aggregate that would contain a list of reservations for many things. For my sample solution, I will assume that members will make a single reservation for a hiking trip and there is no association between the trips, as there might be with a list of flight r eservations on a particular trip. The reservation is composed of a Member and a Trip. It should also contain additional metadata, such as time reservation cre ated, time updated, etc.
Composite	A composite class containing references to all the other classes. This is a container for moving information around the prog ram. Life is much easier when your methods accept and return composite objects.

What to turn in

- Add the UML model, DDD diagram, and sequence diagram to the Design Document you started in Week 1. Ensure that this documen t also has Users Stories (one liners from Week 1) and a brief project description (one paragraph)
- You will implement the classes above in your Domain layer. The classes should include:
  - Appropriate naming conventions and style (see sample code).
  - Overloaded constructors and an default constructor
  - Attributes (Instance variables)
  - Getters & Setters (Generated in the IDE).
  - A toString() Method which overrides the default toString() method. (Generated in the IDE).
  - Hash and Equals methods. (Generated in the IDE).
  - A validate method. Think about the following:
    - Which variables cannot be null.
    - Deeper validation for particular instance variables (e.g. email format must be a String followed by an '@' followed by a dot, followed by a three letter string. This sort of validation can quickly found with a pair programming tool or a web search and plugged int o your method logic.
    - Don't get stuck here this week. It's fine if you just validate for null values this week.
  - Comments
- Implement at least one test.