March 22th, 2024

COE528 Project

Raymond Cao Jiang - 501183087
COE 528
Section 7
Instructor: Boujemaa Guermazi

Raymond Cao Jiang
501183087
Section 7

Use Case:

In my use case diagram, I have an actor for the manager and another actor for the customer. The use cases highlighted in orange are use cases that can be accessed by both the manager and customer, login and logout. The use cases highlighted in blue can only be used by the customer which includes deposit, withdraw and purchase. The use cases highlighted in green can only be used by the manager, which are add customer and delete customer. The type of use case that I have followed from the file Week.6.1.Modeling.with.UML.pdf is the communication type. When the manager or a customer logins, they are exchanging their username and password with the use case.

Class Diagram:

In my class diagram, I have Bank as my main, highlighted in orange. I didn't include the controller classes so I had my Manager and Customer class associated with the Bank instead. The Customer and Manager classes are dependent on the user class because both of these classes contain methods that rely on the usage of the user class to perform certain operations. For example in my Customer class there are three classes (deposit, withdraw and purchase) which access methods from the User class. The Customer class is also dependent on the Level class as the customers each have a level depending on their balance. The Level class is an abstract class in which the silver, gold and platinum classes inherit its methods. There are also other methods in the controller classes but I'm not sure if I'm supposed to include them.

Class that Covers Point 2: Customer

The class that I chose to implement an abstraction function and rep invariant is the Customer class. In this class I wrote the effects and modifies for most of the methods. It also has a toString() method as well as a repOK() method which represents the abstraction and rep invariant.

Parts that form the State Design Pattern:

The parts that form the State Design Pattern is the relationship of the Customer class and Level class, where the customer class has a Level object that changes to the silver, gold, platinum state classes based on the balance. The Customer has a set purchase fee depending on the level state.

Raymond Cao Jiang
501183087
Section 7

References:

Bro Code. (2020, October 6). *Java FileWriter (write to a file) 📝*. YouTube. https://www.youtube.com/watch?v=kjzmaJPoaNc

GreekifIc. (2021, September 11). *The state pattern explained and implemented in Java | Behavioral Design Patterns | Geekific*. YouTube. https://www.youtube.com/watch?v=abX4xzaAsoc

Java Coding Community. (2020, December 23). *JavaFX login form tutorial using scene builder | Javafx and scene builder tutorial | 2020 version*. YouTube. https://www.youtube.com/watch?v=HBBtlwGpBek

Lee, A. (2018, December 13). *Files in java - read text file easily #38*. YouTube. https://www.youtube.com/watch?v=lHFlAYaNfdo

MoosesValley. (2021, August 12). *Java, NetBeans, scene builder - multiple scenes*. YouTube. https://www.youtube.com/watch?v=m5853rAfekE&list=PLfKyhM9sqOY5AvTKh___uoG531WRVigYm

Paraschiv, W. by: E. (2024, January 5). *Java - create a file*. Baeldung. https://www.baeldung.com/java-how-to-create-a-file#:~:text=If%20a%20file%20with%20a,we%27ll%20get%20a%20FileNotFoundException