**Java IV: Project 2 15% of the total grade**

**Due:** as indicated on Moodle

You can work in pairs or in a group of three. You must not work alone.

**Objectives:**

* JavaFX
* Threading
* Unit testing
* 3-tier design/ 3-layer design
* Database connectivity
* git
* Use mock objects

**An e-wallet**

In this project, you will develop a JavaFX e-wallet application. The following items can be put in the wallet:

1. Credit cards
2. Debit cards
3. Cash
4. Personal cards such as a health card or a driving license
5. Notes

The operations of the e-wallet are:

1. Add a card.
2. Delete a card.
3. Pay.
4. Browse through the notes.
5. Add a note.
6. Delete a note.
7. Save e-wallet to DB.
8. Load e-wallet from DB.
9. Display how much cash we have
10. Add cash

**3-layer Architecture**

In this project, your design must follow the 3-layer architecture. The 3-layer architecture consists of the following the following:

1. Presentation layer: that is your JavaFX user interface
2. Application layer: that is the business logic. Sometimes, it is referred to as the logic layer.
3. The data layer: that is the database layer. In this project we will start with a mock object to simulate the DB. In the third assignment, you will create the DB using MySQL to create an actual DB and make your project connect to it.

Graphical user interface, text, application, chat or text message

Description automatically generated

Diagram

Description automatically generated

**MVC architecture**

In the last assignment, we separated the Model (the business logic) from the View (the user interface). In this project, we must also implement the Controller. The role of the controller is to facilitate the communication between the Model and the View.

Diagram

Description automatically generated

**Application:**

Your design should be a superb Object Oriented one. Here you are a description of the problem domain. A user can only have one wallet. The wallet can have a maximum of 10 cards and 10 notes at any single point in time. Payment cards can either be debit or credit. Payment cards have the following attributes:

1. Card holder name
2. Card number
3. Expiry date in the form of mm/yy
4. A three digit security code (written at the back of the card)

Each credit card has a limit. The limit ranges from $100 to $5000. When the user reaches 50% of the card limit, the smart wallet will notify the user. To keep your credit score in a good standing, the user should not exceed 50% of the card limit. The warning generated by the wallet should not prohibit the user from exceeding the 50% threshold.

Each debit card is attached to a bank account. The limit of the debit card is the same as the funds available in the bank account. Since the web-service that communicates with the bank directly is not implemented yet, you need to create a mock object to randomly simulate the functionality of the bank account. Only two methods are needed: boolean checkFundsAvailability(int amount) and boolean withdraw(int amount). checkFundsAvailability(int amount) returns true if the account balance is greater than or equal to amount; false otherwise. boolean withdraw(int amount) will reduce the balance of the bank account by amount only if there is enough funds in the account. It returns true if the operation is successful, false otherwise.

Personal cards have the following attributes:

1. Card holder name
2. Card number
3. Some cards have expiry dates; others do not

Notes have the following fields:

1. Note creation date
2. Note body which is just a string
3. A note can be configured to issue a reminder
4. If the reminder option is set, the frequency of the reminder must also be set in minutes, hours, or days. For example, I should be able to create a note to remind me to mark the exam. I should be able to set the reminder to fire every minute! You decide the form of the reminder: a sound, a message, etc.. be creative 😊

**Specific Requirements:**

* To manage JavaFX dependencies, you need to use a Java with Maven project in NetBeans as indicated in the images below.
* Need to demonstrate the use of unit tests (as many as you see fit)
* You must use gitlab. You need to add me as a maintainer. My name is nbasha.
* Must use the MVC architecture.
* The GUI must run on its own thread. The business logic should run on a different thread. Any lengthy task should run on its own thread.
* Use the observer pattern to display a message when the cash value goes below $30.
* Save all the info of the e-wallet in a database and load it from there. In this project, you will use a mock object to simulate the DB. In the next assignment, you will create and connect to the DB.
* Use mock objects wherever you can not actually run the service. For example, when paying by a VISA credit card, you will not be able to connect to VISA and get a actual response from them! You can simulate this functionality using a mock object for each card. If trying to spend more than the card limit, then the operation should be denied. Sometimes, transaction fails because of a network error; you should be able to simulate that as well.
* A nice option of the e-wallet is to allow the user to display a favorite pic on the user interface. You need to use animation to make the pic move a bit on the screen following a path of your choice.

**Submission:**

You must submit:

* A proper **UML class diagram** in **pdf format** that shows   
  \* names and access modifiers of all attributes,  
  \* names, parameters, and access modifiers of all methods including constructors  
  \* proper relations between classes and interfaces  
  \* no need for setters and getters
* **Sample runs** of the program that show all the functionalities required in **pdf format**
* **The code** itself
* **You will also give a demo of your project in the lab. All team members must be present.**

Graphical user interface, text, application

Description automatically generated

**Grading Rubric**

|  |  |  |
| --- | --- | --- |
| UML diagram | 3 marks | Proper notation Proper class design Proper interactions |
| Threading | 2 marks |  |
| Unit tests | 2 marks |  |
| JavaFX | 2 marks |  |
| Mock object | 1 mark |  |
| Overall code quality | 2marks | Indentation/ comments / method names / variable names / class names |
| Overall functionality | 2 marks |  |
| Using Git | 1 mark |  |

**0 credit will be given if:**

1. No UML is submitted. It is a must to submit the detailed UML class diagram
2. The code does not compile

One submission per group please. In the Moodle submission comment, please indicate the names and the ID’s of your partners. -1 if this information is not given accurately.

**References:**

<https://developer.mozilla.org/en-US/docs/Glossary/MVC>

<https://www.ecanarys.com/Blogs/ArticleID/76/3-Layered-Architecture>