

David Morales

Axel Diaz

July 26th, 2020

Instructor: Dr. Daniel Mejia

Programming Assignment 4

Program Explanation:

In this assignment, we were supposed to merge our bank programs (with a partner), implement two new features, and draw UML diagrams of our program. We then demoed another team's program and ours vice versa. We wrote about their feedback on our program, and we wrote them feedback about their program.

We first tackled the problem by first looking at each other's code and making notes. We then identified areas in our code where we can merge the program more easily. Then we finally merged the code and made a functional version. We fixed any lingering bugs and commented who wrote what parts of the code. We broke the problem down by highlighting parts of code we liked to be in the merged final, and then merged code by parts.

What did we learn?

We learned a lot about working as a team in this project. We needed to coordinate our efforts to make a functional prototype. We both needed everything we learned in this class to get this project then. While our solution isn't perfect, we could have improved the organization of how information is processed (UserUtilities.java), and make it more robust and expandable.

Some other ideas were to refactor different parts of code, but a lot of consideration was done on what parts of the code was going to be replaced, rewritten, or kept. It took us two weeks to complete this project.

Solution Design:

What we did in this program was to merge the class framework (from Axel Diaz) to the user menu and information process (from David Morales). After we did that, we rewrote parts of the code that was needed to make the program work. After fixing initial problems and bugs, we then wrote the interface Printable and added password authentication. Password authentication in our program is just a simple string comparison (as the requirements stated they didn't need to be salted or hashed), as added to when the user selected their account. The interface called Printable is implemented in BankStatement.java and Customer.java. It implements methods called print(), printTransactionLog(), and printBankStatement(). Their functions are to print user info, transaction history, and bank statements.

Some data structures we used in this project were hashmaps and ArrayLists due to them being dynamically expandable and fast retrieval times. Some assumptions we made were how the interface Printable is implemented. We wanted it to print user information in classes that contained all user information like Customer.java and BankStatement.java.

Testing:

We tested our program by doing things like creating multiple accounts and just messing around with them. For example in one case we broke the code by creating an account with no checking or credit and from another account we deposited to the account with no checking and it was successful but the “money” was nowhere to be seen. We then continued to fix it and now it works fine. We used black-box testing because it was the better one in this case to use since it was more on actually trying to break the code by input and see what output we could get.

Testing Results:

Describe the results of your tests.

The results of our tests at first were kind of bad but then with many tests later and solutions we didn't get any more errors or were able to break it. So the results of our tests now are really good.

Include any console outputs showing your results

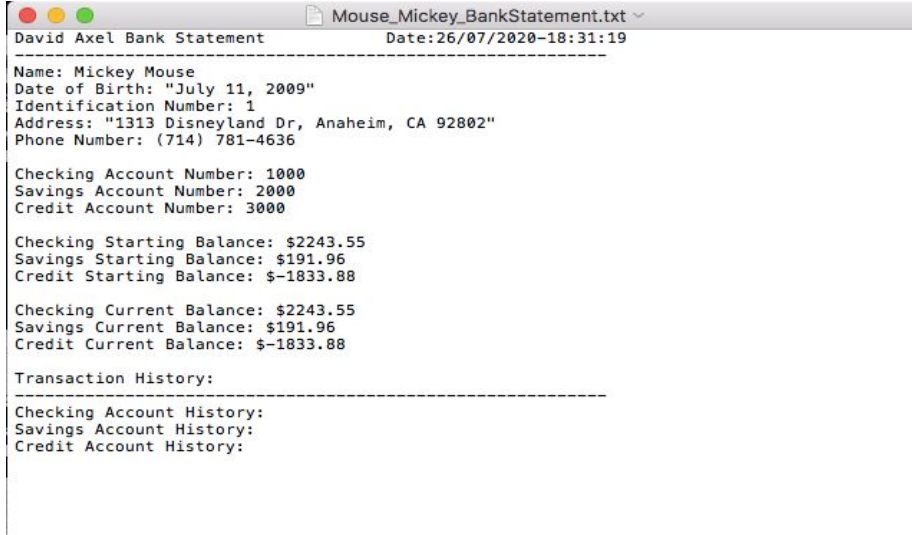
```
Welcome bank manager!
[0] Inquire user by name (with option to print bank statement)
[1] Inquire user by account type/number (with option to print bank statement)
[2] Print all users detailed information
[3] Return to user menu
[4] Exit program
0
Who's account would you like to inquire about?
Please input first name: (case sensitive)
Mickey
Please input last name: (case sensitive)
Mouse
```

```
Which account?
[0] Mickey Mouse ID: 1
0
First Name: Mickey
Last Name: Mouse
Date of Birth: "July 11, 2009"
Address: "1313 Disneyland Dr, Anaheim, CA 92802"
Phone Number: (714) 781-4636
Email:MickeyMouse@disney.com
Identification Number: 1
Checking Account Number: 1000
Checking Account Balance: $2243.55

Savings Account Number: 2000
Savings Account Balance: $191.96

Credit Account Number: 3000
Credit Account Balance: $-1833.88
```

Include any text document output as a result of your tests.



```
David Axel Bank Statement      Date:26/07/2020-18:31:19
-----
Name: Mickey Mouse
Date of Birth: "July 11, 2009"
Identification Number: 1
Address: "1313 Disneyland Dr, Anaheim, CA 92802"
Phone Number: (714) 781-4636

Checking Account Number: 1000
Savings Account Number: 2000
Credit Account Number: 3000

Checking Starting Balance: $2243.55
Savings Starting Balance: $191.96
Credit Starting Balance: $-1833.88

Checking Current Balance: $2243.55
Savings Current Balance: $191.96
Credit Current Balance: $-1833.88

Transaction History:
-----
Checking Account History:
Savings Account History:
Credit Account History:
```

Reflection:

The process of combining code was very challenging. Figuring out what parts of the code we wanted to keep, removed, and refactor was a difficult process since there were parts of each other's code we liked. Understanding each other's code was necessary on what parts of code we wanted to merge. My partner and I took considerable time reading and understanding each other's code before we gave each other proposed versions of the merged project.

Some problems we've faced where merging the class framework from one partner with the user menu and processing from one partner. Huge parts of code had to be refactored and rewritten to fit the new requirements, and to be functional. We solved them by merging the code part by part to ensure all the features and functions of the bank program work correctly.

Demo of another team:

Who demo'd to you?

The team members who demo'd were Alan and Brandon.

Did you understand their process to perform tasks?

Yes me and my partner understood their process.

Did they provide you with Javadoc?

No, they did not provide us with Javadoc, just the console.

Did you break their code? How?

Yes, we broke their code by creating a user without a name just leaving everything blank. It didn't stop the code, it would keep working which was fine but it was not supposed to work like that.

Did they meet all functionality requirements?

Yes, for the most part they did meet everything it was a little hard to break their code.

Demo for another team:

Who did you demo with?

We demo'd with Alan and Brandon.

Did you provide them with enough information in the console prompts?

Yes, we explained it well enough for them to understand and be able to test our code.

Did you provide them with Javadoc?

No, we did not provide any Javadoc just the console.

Did they break your code? What did you learn from it?

No, they were not able to break our code but it did give us some insight on maybe something to think about that we didn't think about.

Did you meet all the functionality requirements?

Yes, they were able to successfully make an actual account and do everything they were able to do with it.

Person One: Axel Diaz

I contributed to the project by being the navigator and tester. I helped this project by helping mainly my partner in constantly testing the code and just debugging some of the code that was wrong. Besides that I also did the UML scenarios and we really just went half on almost all the work. What I learned from working with a teammate was that they have a different mentality and just think overall differently from you so seeing a different point of view and the way he was doing his work was a huge help for me to learn how to approach a problem more

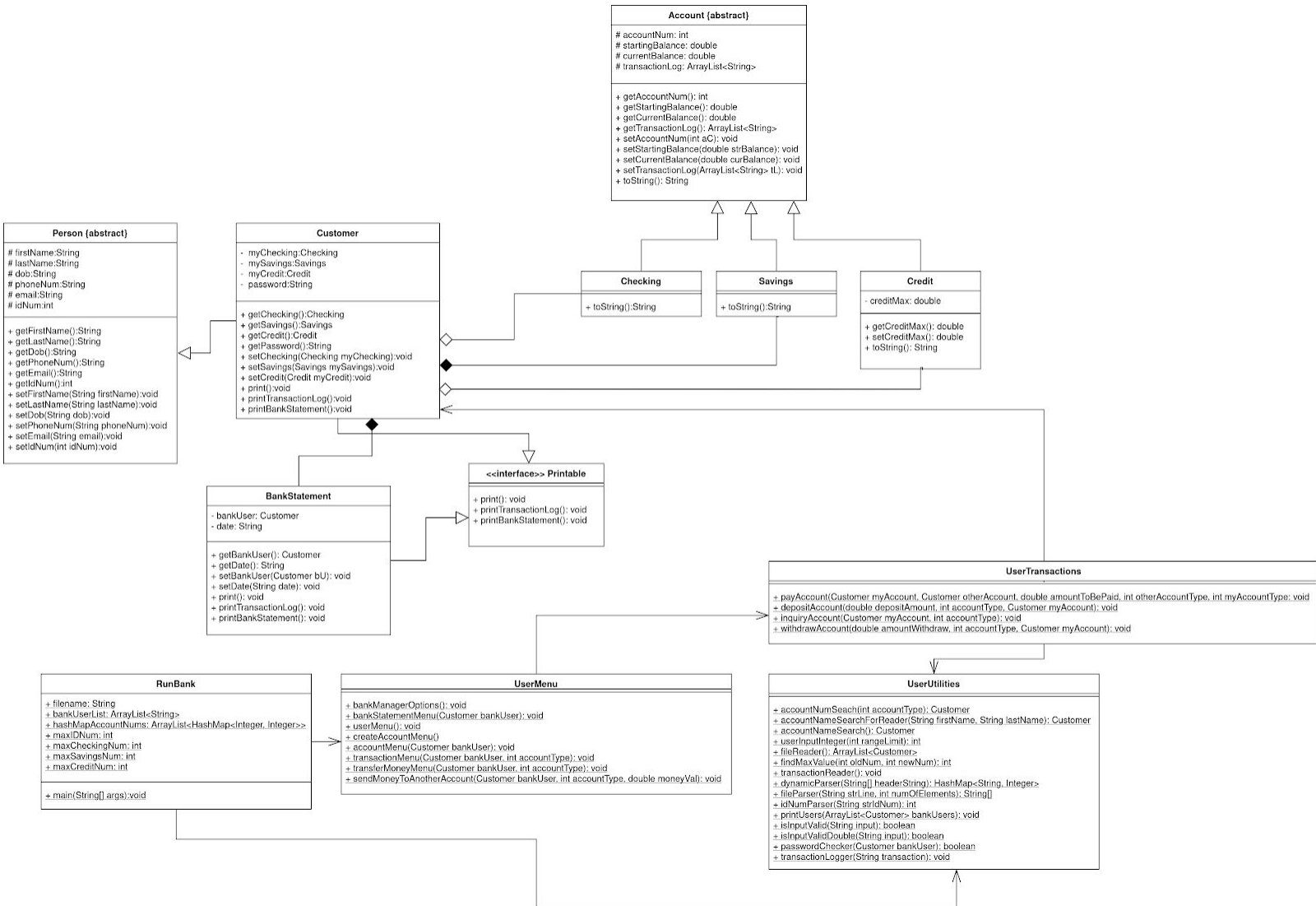
efficiently. Also having a second opinion from someone else is always good and helpful to fix your mistakes and just be better.

Person Two: David Morales

I contributed to this project by being the driver (pair programming) in the coding parts of the project. While Axel Diaz was the navigator and tester in this project. I helped this project by implementing the planned changes my partner and I agreed upon. I wrote the code untested, and I sent updated versions of the project to my partner to debug and clean up my code. I also did the UML class diagram and UML state diagram of our project.

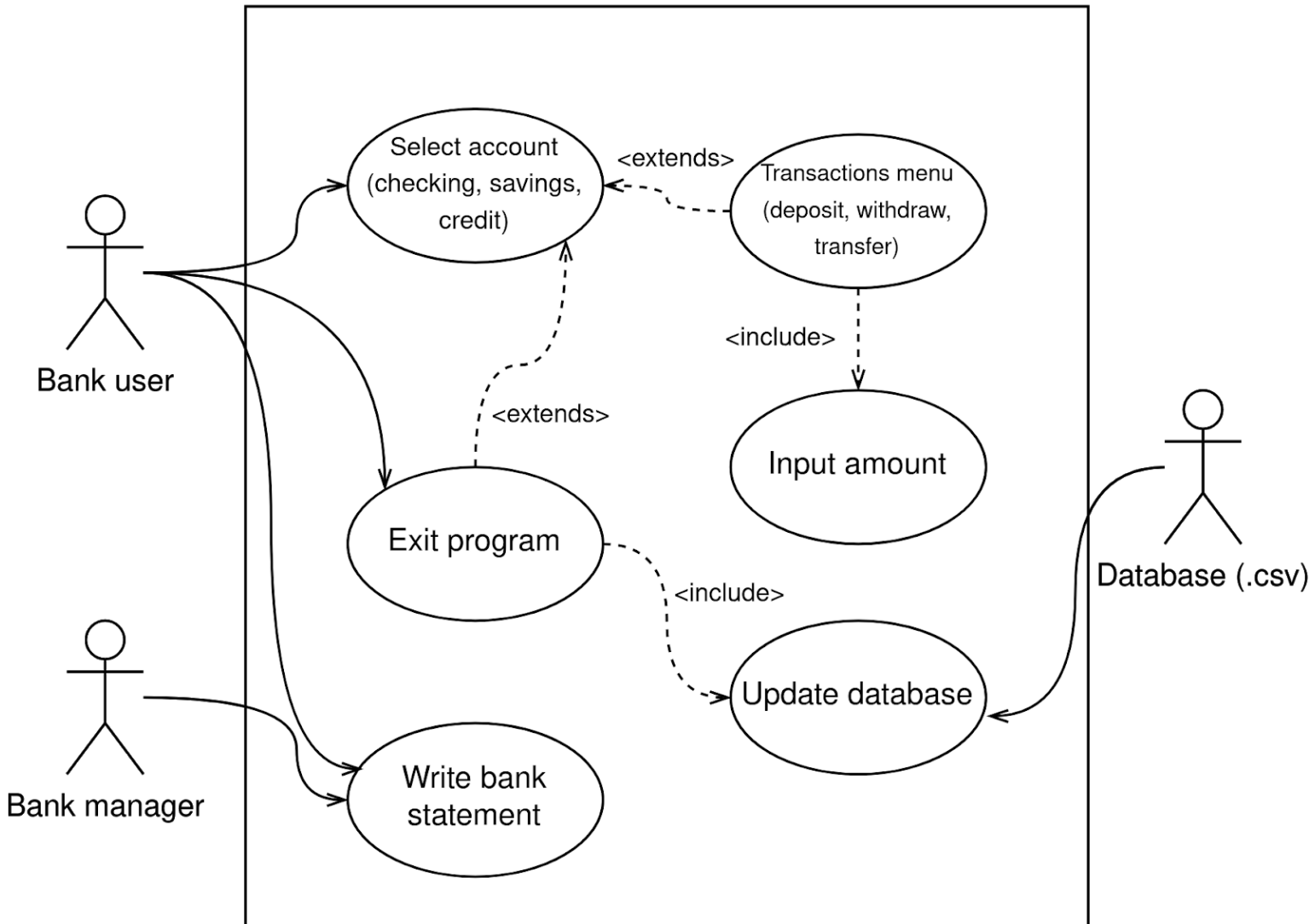
What I learned from working with a teammate is the need to coordinate and plan in order to tackle the problem more effectively. I also learned of all the small mistakes and errors I tend to make, and having a partner to check my work and give feedback is very helpful.

UML Class Diagram:

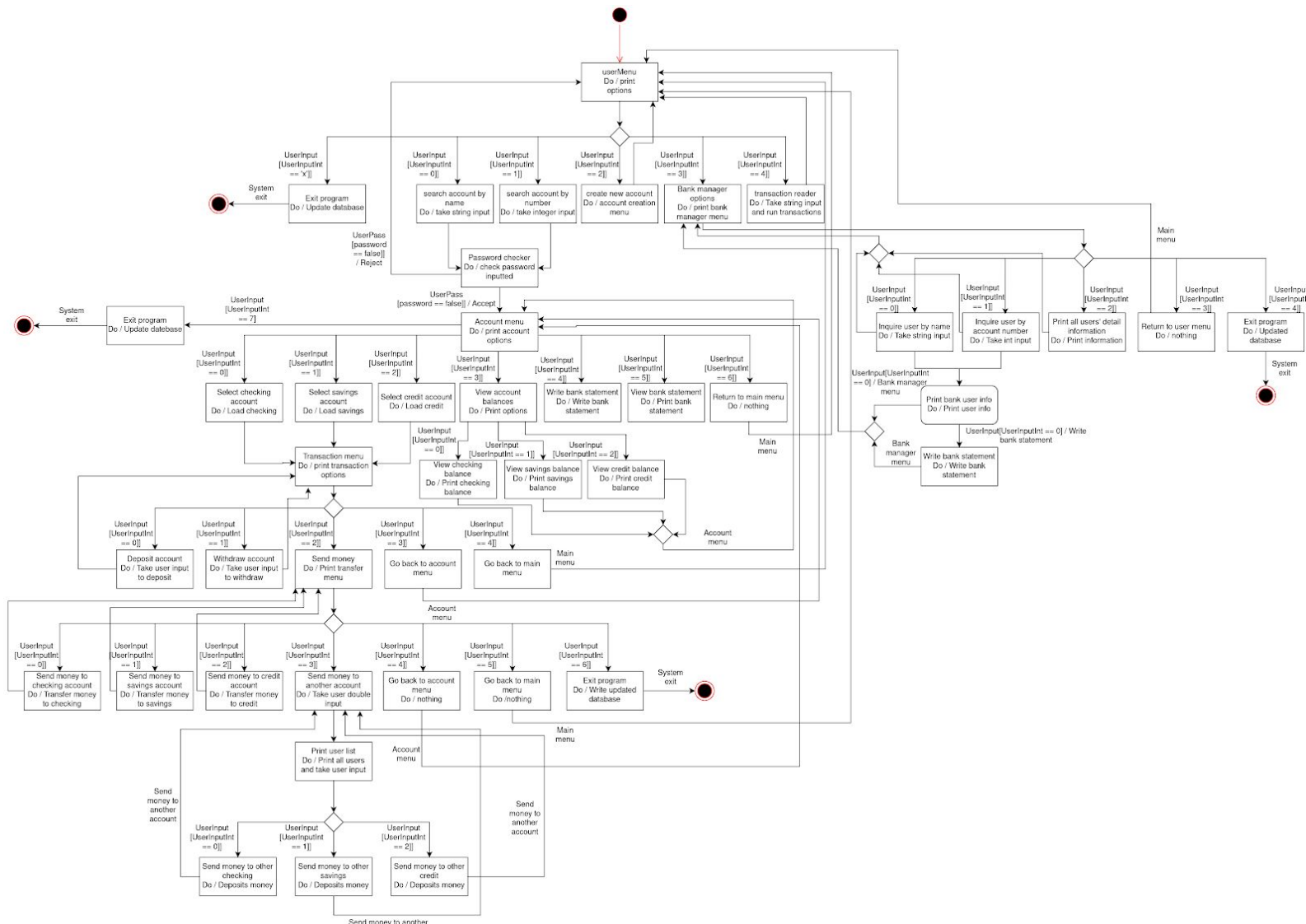


UML Use Case Diagram:

David Axel
Bank



UML State Diagram:



Use Case Scenarios:

Name	Login to account
Short Description	Login to your account using your username and password or by ID number and password.
Precondition	Have the app installed and an account already created and click on either "Search account by name" or "Search account by ID number".
Postcondition	After successful login, let the user into their account.
Error Situation	Wrong username, ID number or password.
System State in event of error	That is the wrong username\ID or password please try again.
Actor/s	User and database.
Trigger	User starts app.
Process	<ol style="list-style-type: none">1. User chooses "Search account by name".2. User enters username.3. User enters password.
Alt. Process	<ol style="list-style-type: none">1a. User chooses "Search account by ID number".2a. User enters ID number.

Name	Create Account
Short Description	Create an account.
Precondition	Have the app installed and choose "Create new account".
Postcondition	After creating an account return to main menu.
Error Situation	User leaves name or password empty.
System State in event of error	Must enter name, can't leave empty.
Actor/s	User and database.
Trigger	User clicks on "Create new account"
Process	<ol style="list-style-type: none"> 1. User enters first name. 2. User enters last name. 3. User enters date of birth. 4. User enters phone number. 5. User enters email. 6. User enters password. 7. User chooses to create a checking account. 8. User chooses to create a credit account.
Alt. Process	None

Name	Transfer money
Short Description	Transfer money to checking, savings or credit account or to someone else.
Precondition	Must be logged in and have funds to transfer.
Postcondition	Transfer successful and update balances.
Error Situation	Not enough funds.
System State in event of error	Cannot make transfer, not enough funds.
Actor/s	User and database.
Trigger	User first chooses which type of account they want access to then clicks on "Send money".
Process	<ol style="list-style-type: none"> 1. User chooses to send money to another type of account like checking, savings or credit. 2. User then decides how much money they want to transfer.
Alt. Process	<ol style="list-style-type: none"> 1a. User chooses to send money to someone else's account. 3. User chooses the person to send money to. 4. User then chooses which type of account to send it to.

I confirm that the work of this assignment is completely my own. By turning in this assignment, I declare that I did not receive unauthorized assistance. Moreover, all deliverables including, but not limited to the source code, lab report and output files were written and produced by me alone.