Michael Wise
Professor Arias
CMPT 220 Software Development I
24 March 2020

Project Milestone

Abstract

For this project, I have created a program that simulates a functioning ATM machine. Some of the classes I have implemented include ATM, Transaction, Account, Security, and User. There are some class hierarchies - for example, the Account class is a superclass of the Savings and Checkings class. The next step in my process is implementing the GUI to my ATM. After doing some research, it seems like JavaFX is the best for accomplishing this. I have started to watch tutorials and read parts of our textbook that outline the general process. It might take a lot of trial and error, but I hope to be able to implement it into my existing classes to make a functioning ATM. Once I get a basic working GUI, I plan to add new features that traditional ATMs don't necessarily have to make the transaction process more efficient. I'm still looking into maybe implementing a database, but I'll decide what I want to do once I implement the GUI.

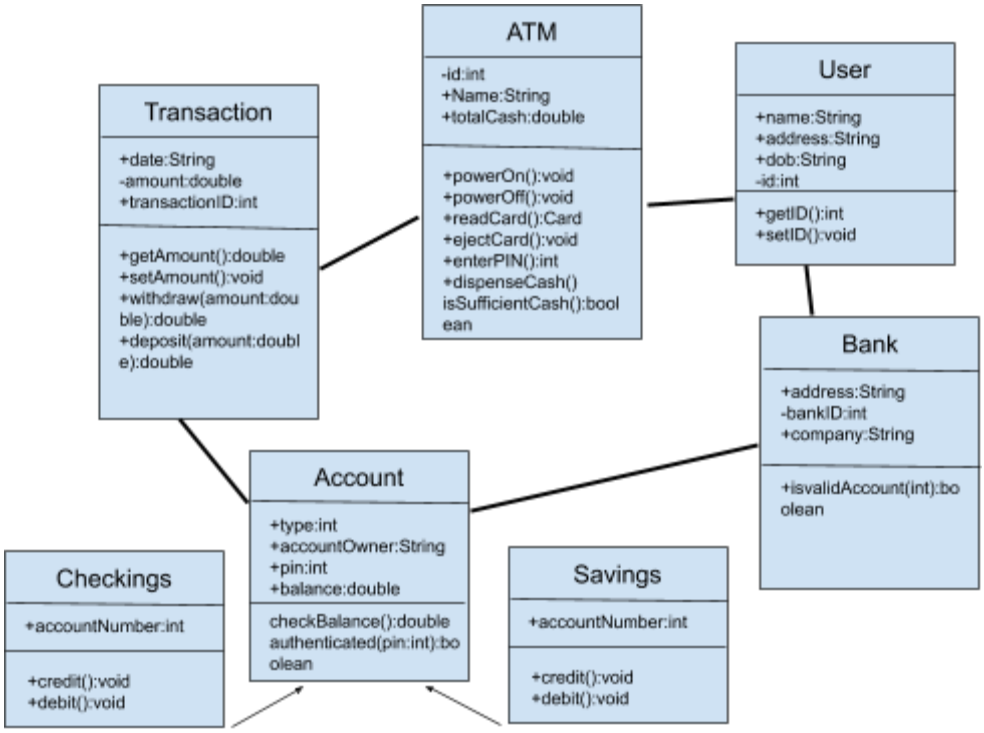
Introduction

The main reason I wanted to do this was to take a simple programming task and really improve my skills by making it as complex and efficient as possible. As a data science major, I really have no plans using Java in any future jobs, but I think it is crucially important to understand the software development process regardless as well as having a wide spectrum of programming knowledge in various languages. I wanted to learn how to use GUIs in Java along with learning how to effectively integrate classes into a big program.

Detailed System Description

The ATM will allow users to perform standard actions you would expect to get at an ATM. The main class interacts with Transaction to handle withdrawals, deposits, cash amounts, and interacting with your actual bank account. The Savings and Checkings class extend the Account class, dealing with the user's info which is provided from the bank. Connected with that

is the Bank class which also interacts with the User class that has all the essential information of the customer. Not all the current classes/methods are included in this specific UML diagram.



Requirements

The system is addressing the problem of being able to transfer money from cash to electronic currency efficiently and securely. ATMs are designed for the purpose of allowing users to withdraw and deposit money in convenient locations without having to go directly through a bank or other service. They save a lot of hassle when you are in desperate need for physical cash (assuming that the location has an ATM, which is a totally separate problem).

Literature Survey

There are hundreds of ATM manufacturers around the world, all having different features. In an era where digitalization is getting bigger and bigger, there have been features that allow users to not require cards or cash, such as Apple Pay. People can access/interact with their balances without the need of a physical credit/debit card or cash. We are already advancing towards a society where even ATM machines are becoming obsolete with software like Venmo and Paypal.

User Manual

The ATM will feature a GUI that will make transactions seem effortless. Once the ATM gets access to your account number by entering your PIN, the menu will give options to view balance, deposit, and withdraw funds. It will be touch-based, so all you have to do is click on the option you want (rather than some machines that have numbered analog buttons you press to choose an option). It will also have a keypad that will allow users to enter specific amounts.

Conclusion

The general goal of this system is to make the transaction of currency easy and accessible to anyone regardless of technological competency. There should be multiple ways to access an account balance and be able to effectively authenticate users by interacting with the bank's information database. The software encapsulates the functionality of the hardware parts of the physical ATM.

References/Bibliography

JavaFX Tutorial

- https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm