Project Introduction and Objective

I chose to design a program that simulates a bank ATM. Users can create new bank accounts and log in to existing accounts. They are able to deposit and withdraw money as well as take out loans and pay them off. Their balance is constantly updated and shown to the user whenever a change is made. I designed this bank program to be simple to use, operated by user input. This program would work as a good basic template for a real bank's ATM. New features could easily be added to the program if desired.

High Level Design

Abstraction is utilized in this project via the Loan abstract class. The SmallLoan, MediumLoan, and LargeLoan classes all extend this abstract class so they can implement certain methods according to the subclass. Encapsulation is used throughout the program so that the Login and Account classes can call the methods they need to accordingly. Inheritance is also utilized when it comes to the Loan class. It prevents duplicate code and makes adding more Loan types in the future easier. Polymorphism exists only in the SmallLoan, MediumLoan, and LargeLoan classes when they override the createLoan method in order to print the correct message to the user.

Class Diagram

See attached file.

Sequence Diagram

See attached file.

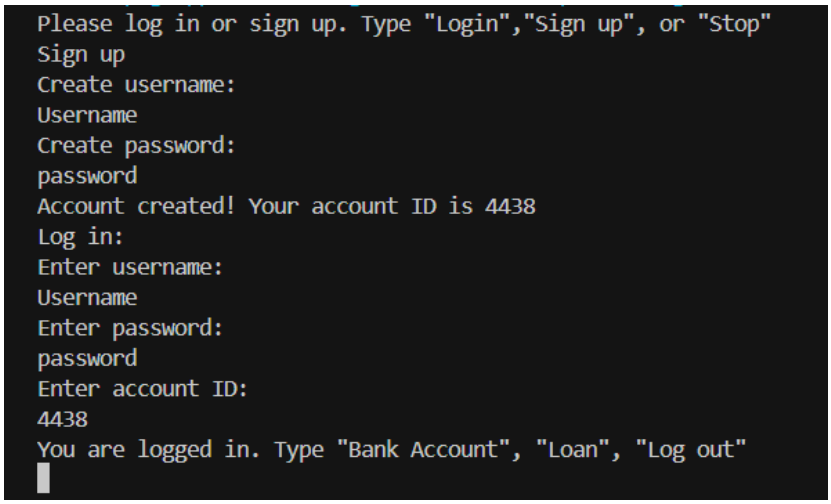
Design Patterns

This bank program utilizes the template design pattern in the form of the Loan abstract class. The subclasses all function if they follow the skeleton and implement the abstract methods. In addition, the different subclasses are able to override methods that are required to do their own unique responsibilities. The facade design pattern would also apply to this program. Although the backend is not overly complex, the frontend is

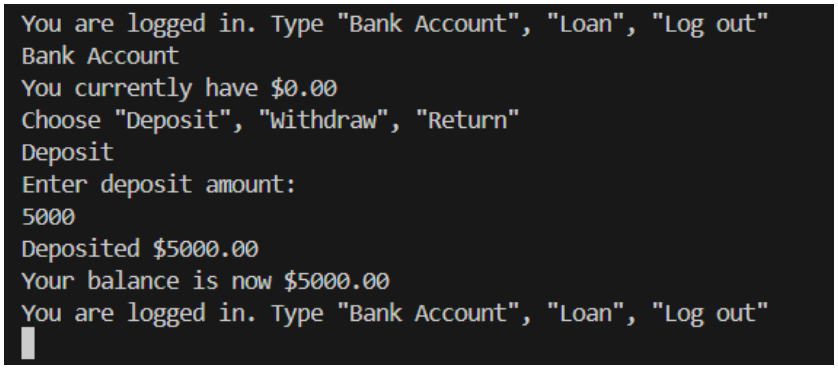
simplified for user input and does not require users to understand code to operate the program.

Key Functionalities

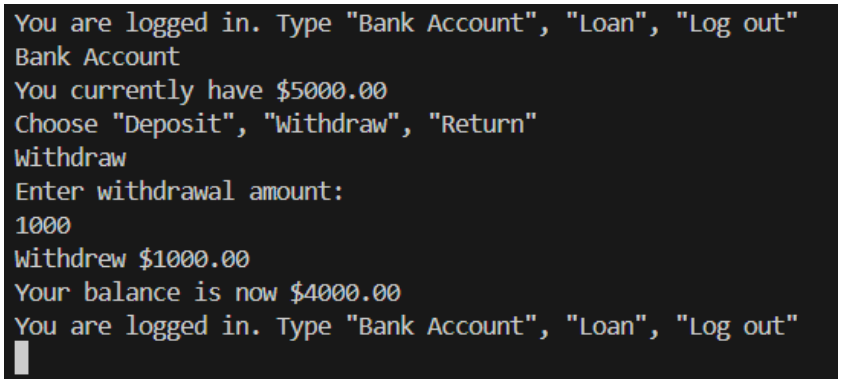
Login and Sign up features:



Deposit feature:



Withdraw feature:



Take out a loan feature:

```
You are logged in. Type "Bank Account", "Loan", "Log out"
Loan
You currently have $4000.00
Will you take a loan or pay one off? Enter "Small", "Medium", "Large", "Payoff", "Return"
Small
Small loan taken.
Deposited $1000.00
Your balance is now $5000.00
You are logged in. Type "Bank Account", "Loan", "Log out"
```

Payoff a loan feature:

```
You are logged in. Type "Bank Account", "Loan", "Log out"
Loan
You currently have $5000.00
Will you take a loan or pay one off? Enter "Small", "Medium", "Large", "Payoff", "Return"
Payoff
How much will you pay off? You have $1100.00 left.
1100
$0.0 left to pay
Withdrew $1100.00
Your balance is now $3900.00
You've paid off your loan!
You are logged in. Type "Bank Account", "Loan", "Log out"
```

Notes and Conclusion

When originally designing this project, I wanted to have a transfer function and a check redeeming function. After beginning my project, I could not think of a way to implement a transfer feature because of how my login feature worked. Now, with my current login function, a transfer function would be easy to implement. The process in which I planned on coding the project also changed, which caused me to implement certain features different than originally intended. I also changed the Loan interface into the Loan abstract class. By using an interface, I did nothing substantial for trying to reuse code. With an abstract class, I could better apply abstraction and inheritance. If I were to continue to expand on this project, I would add an account transfer feature and a check redeeming feature.