Bank Management System

06.10.2021

—

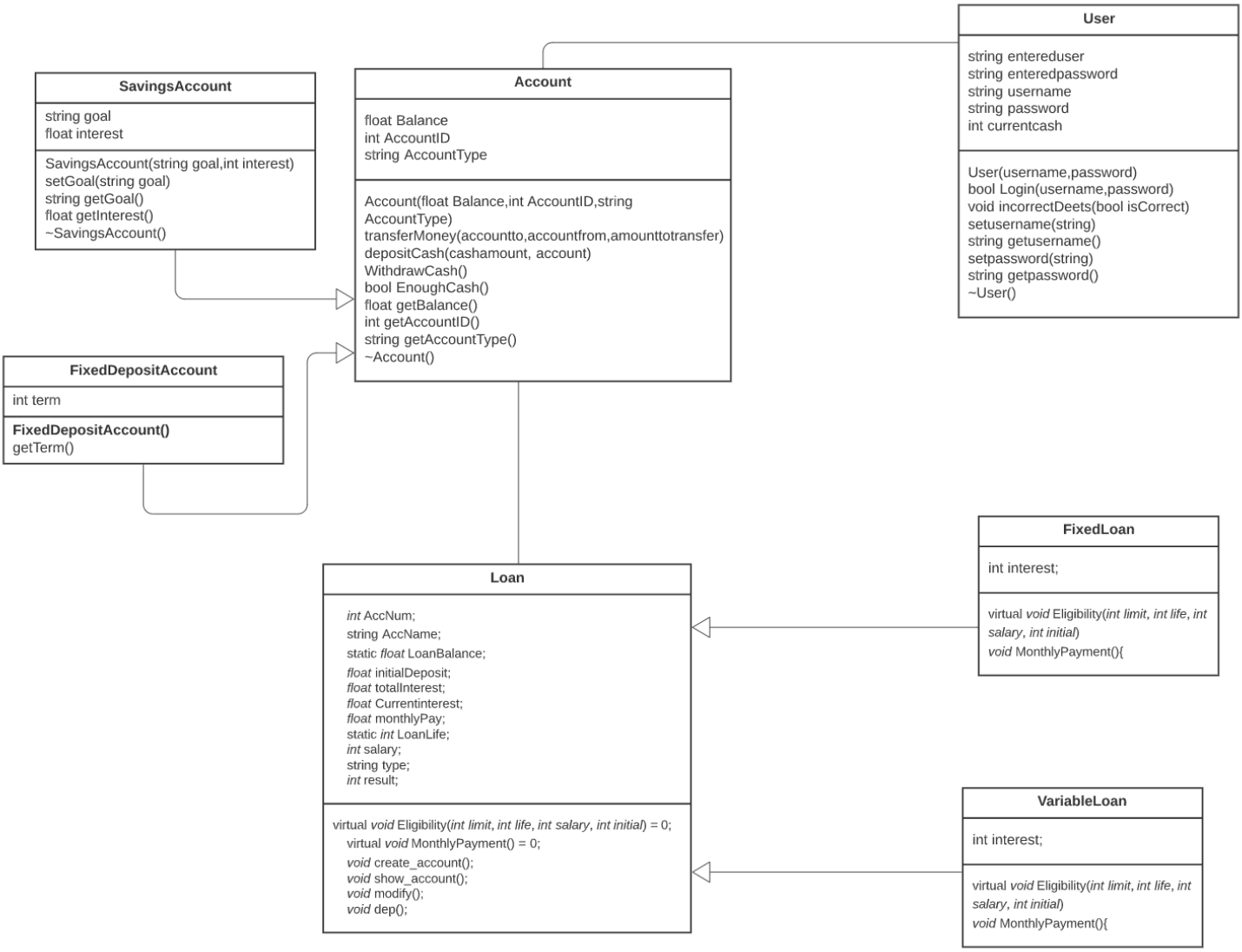
Project By Vidit Patel.

Introduction:

The project which I have created illustrates and replicates an online bank management system. Users can create and/or access an account within the bank that displays a randomly generated balance and can transfer money between the account types provided and can deposit and withdraw cash. Users are also able to apply for a fixed and variable rate home loan and get an instant decision if the loan is approved.

Class Diagram of the banking system:

The class diagram below is constructed to produce a blueprint of the banking system. The diagram highlights the explicit relationships between the classes and objects used in the system-moreover, how the classes and objects interact with each other.



Class Descriptions:

Loan:

Loan is an **abstract base class** in this system. There are two types of loans offered by the bank: Fixed rate loan and Variable rate loan.

The behaviors in the Loan class allows the user to:

1. Create a new loan
2. Show the details of existing loan
3. Modify the details of existing loan (for example, monthly payments)
4. Deposit additional funds into the loan

FixedLoan:

FixedLoan class is the **subclass** of the **Loan** class. FixedLoan inherits class members from the Loan class to make the system more efficient and allows the code to be reused because all fixed rate loans are a type of loan. The class functions in the FixedLoan allow the user to:

1. Check if they are eligible for a fixed rate loan, if so, then:
2. Calculate how much they will need to pay the bank each month-and how much total money they will pay in interest.

VariableLoan:

VariableLoan class is the **subclass** of the **Loan** class. VariableLoan inherits class members from the Loan class to make the system more efficient and allows the code to be reused because all variable rate loans are a type of loan. The class functions in the FixedLoan allow the user to:

3. Check if they are eligible for a variable rate loan, if so, then:
4. Calculate how much they will need to pay the bank each month-and how much total money they will pay in interest.

Account:

The Account class is the **superclass** for the SavingsAcccount and FixedDepositAccount because both savings accounts and a fixed deposit account are types of bank accounts. The Account class allows the user to:

1. Transfer money
2. Deposit money
3. Withdraw money
4. Check if they have enough funds to transfer

5. Check their account balance

SavingsAccount:

The savingsAccount class is a **subclass** of **Accounts** class. The savings account allows the user to:

1. Set a saving goal
2. Save for a goal
3. Create a new type of bank account - which is the saving account
4. Get interest on their account balance

FixedDepositAccount:

The FixedDepositAccount class is a **subclass** of **Accounts** class. The FixedDepositAccount account allows the user to:

1. Create a fixed deposit bank account
2. Select how long they wish to deposit the money for
3. Stops the user from transferring money OUT of this account

User:

The **User** class allows the user to:

1. Input username and password and check if the credentials are correct with the data in the system

Programming Concepts Used:

Memory allocation from stack and the heap

- **Arrays**:

The Accounts class will obtain an array containing Account objects which represent the different accounts of the user

- **Strings**:

The User class will take in strings and deduce whether the inputted string

- **Objects**:

The management system obtains User, Account, and Loan objects.

User Input/Output

- **I/O of different data types**:

The system takes in inputs in the form of integers, floats and strings, this input is then converted into the desired formatted and processed accordingly

Object-oriented programming and design

- **Inheritance**:

Inheritance has been used extensively throughout this project to ensure that it permits code reusability. There are two accounts available for the user, savings account and fixed deposit account. Both of these accounts are a type of bank account; hence, they inherit common behaviors and states from the general bank account.

Secondly, there are two loans offered by the bank, fixed rate and variable rate. Both of these are a type of loan, therefore, they share common features from the loan class.

- **Polymorphism & Abstract Classes:**

Polymorphism and abstract classes were used in the Loan section of the bank system. All types of loans have a certain criteria for eligibility, for example, minimum deposit, minimum salary, and maximum borrowing capacity. Hence, this is a behaviour which is common across all loan-thereby, it is a pure virtual function.

Testing

- **Test Inputs / Expected Outputs:**

Input and output testing will be conducted on all the files in the program that require the need for an input by the user. A corresponding output file will also be created to ensure the output for that program is what is expected.

- **Automated Testing:**

Unit testing will be conducted on all class files and each of the corresponding member functions will be tested for various inputs and outputs to rule out any discrepancies and errors within the code.

- **Regression Testing:**

Thorough regression testing will be conducted on the entire program to ensure that no defects are present and that all the information that is being inputted into and outputted by the program is consistent with what is expected through input validation.