ATM Simulator Requirements

Summary

The ATM application simulates the functionality of an ATM, which allows the user to create accounts, select an account, withdraw and deposit money, check balance and display a list of transactions. The application allows for the creation and management of two different types of accounts, chequing and savings accounts each with specific business rules.

Project Setup.

The project consists of multiple classes as shown in the UML model of the system.

1. The AtmApplication's implements the interaction with the user, displays the menu, takes the input from the user and displays the output
2. The Bank class holds the list of accounts and provides account information to the AtmApplication
3. The Account class implements the basic features of an account: balance, withdrawals, deposits and interest calculation
4. The ChequingAccount is a specialized account that implements business rules associated with chequing accounts
5. The SavingsAccount is a specialized account that implements business rules associated with saving accounts

Detailed Design

1. **ATM class**. The class implements the interactivity between user and bank as described in the following requirements:
    a. Displays the ATM Main Menu until the user chooses to exit the application. The ATM Main Menu consists of the following options:
        1. *Create Account*: allows the user to create an account
        2. *Select Account*: allows the user to enter the account number of the account they want to work with
            i. Upon selecting the account, the application displays the Account Menu as follows:
                a. Check Balance
                b. Deposit
                c. Withdraw
                d. Display Transactions
                e. Exit Account
        3. *Exit*: allows the user to exits the application

   b. Creates a bank object which in turn shall create list of 10 default accounts with account number starting with 100 (i.e. 100, 101, … 109). Each of the default accounts shall have a default initial balance of 100. This will allow the user to exercise the account functionality without creating accounts.
   c. Ensures each menu option is handled by a separate event handler method that is named on<menu option> (e.g. onCheckBalance(), onCreateAccount(), etc.)

2. **Bank class**. The class defines a list of Account objects.
   a. Defines a constructor that creates a list that holds the account objects and creates 10 accounts with an initial balance of 100.
   b. Defines and implements the findAccount() method that finds and returns an account given the account ID. This method is used by the Atm to retrieve an account using the ID entered by the user

3. **ATM Business Logic**. Implements the following business logic:
   a. Main Menu Implementation.
      1. *Create Account*. Prompts the user to enter the account number, initial balance, annual interest rate. Adds the created account to list of accounts managed by the bank.
      2. *Select Account*. Prompts the user for an account number, retrieves the account and displays the Account menu to allow the user to work with that account.
      3. *Exit*. Exit the application
   b. Account Menu Implementation
      1. *Check Balance*. Displays the balance of the previously selected account
      2. *Deposit*. Prompts the user for an amount to deposit and perform the deposit. Record the deposit as a transaction.
      3. *Withdraw*. Prompts the user for an amount to withdraw and perform the withdrawal. Record the withdrawal as a transaction
      4. *Display Transactions*. Display the list of transactions performed on this account.

Inheritance and Polymorphism.

The Account class is specialized by two classes: *ChequingAccount* and *SavingsAccount* as shown in Appendix 1. The appropriate methods are overridden to implement the requirements below:

1. **Chequing Account**:
   a. The chequing account has an overdraft limit of 500$. That means that users are allowed to withdraw up to 500$ more than they have in the account.
   b. Annual interest rate of **maximum 1%**.
2. **Savings account**:
   a. Cannot be overdrawn

b. Annual interest rate of **minimum 3%**
c. For every dollar deposited the bank automatically deposits half a dollar (e.g. the account is setup such that an employer/parent automatically contributes as well)

The account creation menu in the ATM application is extended to allow the user to select the type of account they want to create *chequing* or *savings.* Based on the type selected by the user, the application creates either a ChequingAccount or a SavingsAccount. The rest of the application works unchanged though polymorphism.

Error Handling. The input is checked and all errors are handled both in terms of range and type of data entered using standard error checking as well as exception handling.

Data Persistence. The bank saves all the account information using a text file for each account. The name of the text file is based on the account number ensuring uniqueness. The account file contains the account properties (number, balance, interest rate and type) as well the list of transactions performed.

1. When the application starts, the bank shall check the data directory for all the account files created and created the appropriate accounts with all their data
2. When the account menu is exited after performing several transactions the account file shall be saved such that it is up to date.

Appendix 1

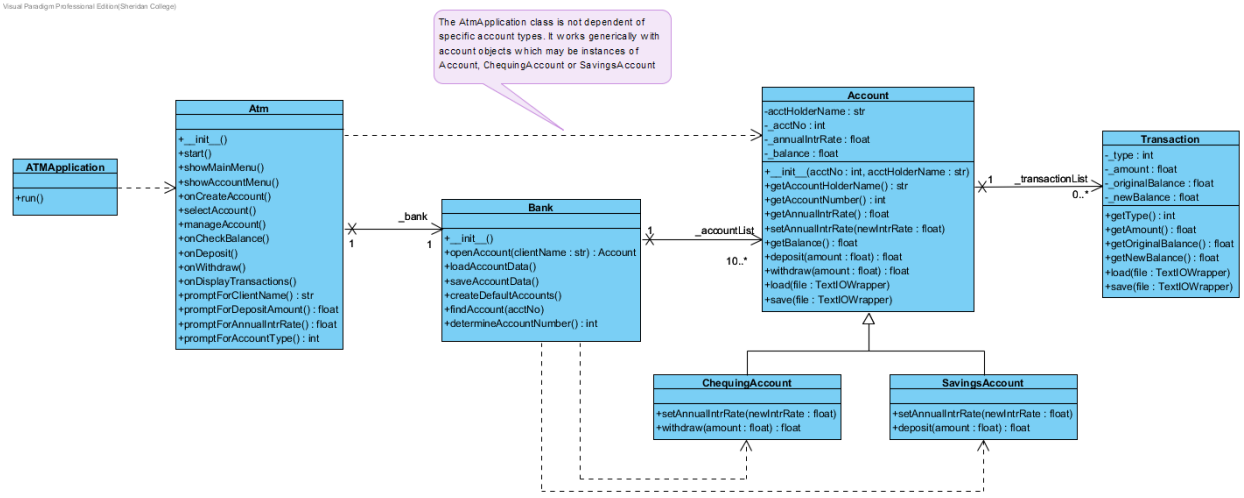

Figure 1: Main Thinking Map

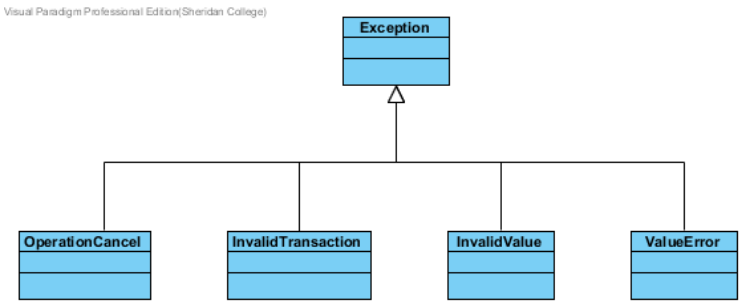

Figure 2: Exception Handling

Appendix 2: Output Example

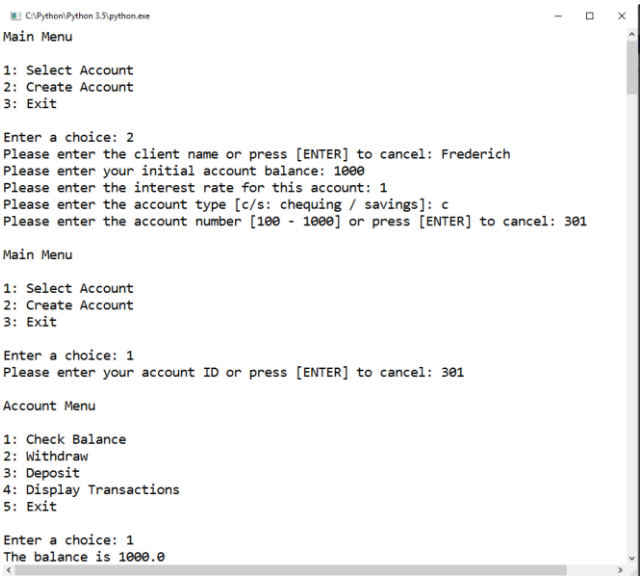

Figure 3: Creating an account (1/3)

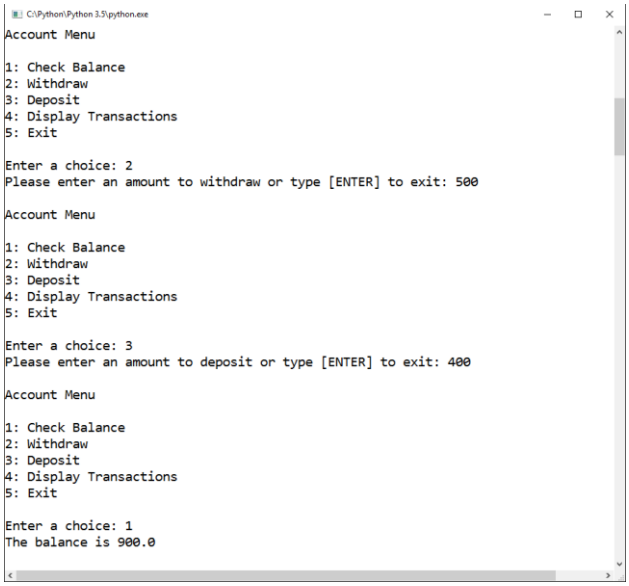

Figure 4: Banking Transactions (2/3)

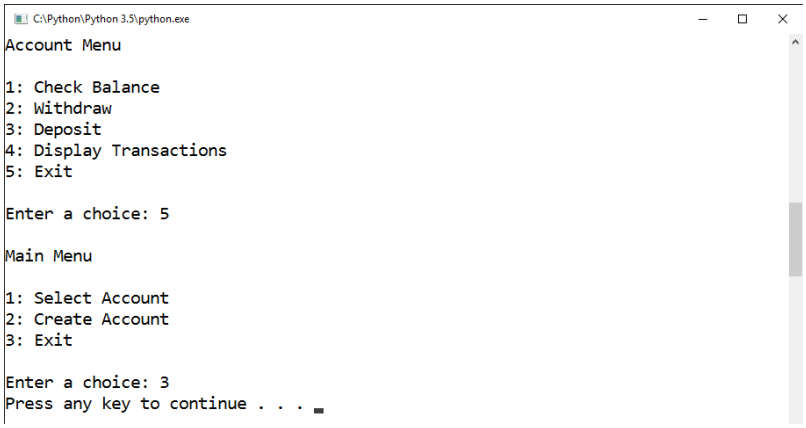

Figure 5: Exiting Account Menu and Application (3/3)