CSCI185-M04

ATM Project

**Table of Contents:**

* Page 3: Analysis
* Pages 4-5: Design and CRC Cards
* Page 6: UML Diagram
* Pages 7- 12: Method Documentation
* Pages 13 - 17: Implementation /Code
* Page 18: Output

**Analysis:**

This project is to create a simulation of an Automatic Teller Machine. This ATM is used by a bank and functions almost exactly how a real life ATM is supposed to work. The customer accesses their bank account using a customer identification number and a 4-digit personal identification number. The customer has two accounts within the one bank account, either a checkings or savings account. Once an account is chosen, the user can either choose to deposit or withdraw from the account. The customer will continue to do this until they choose to quit.

**Design:**

1. Classes:

*Account*

*BankAccount*

*SavingsAccount*

*CheckingsAccount*

*ATM*

*ATMDriver*

1. CRC Cards:

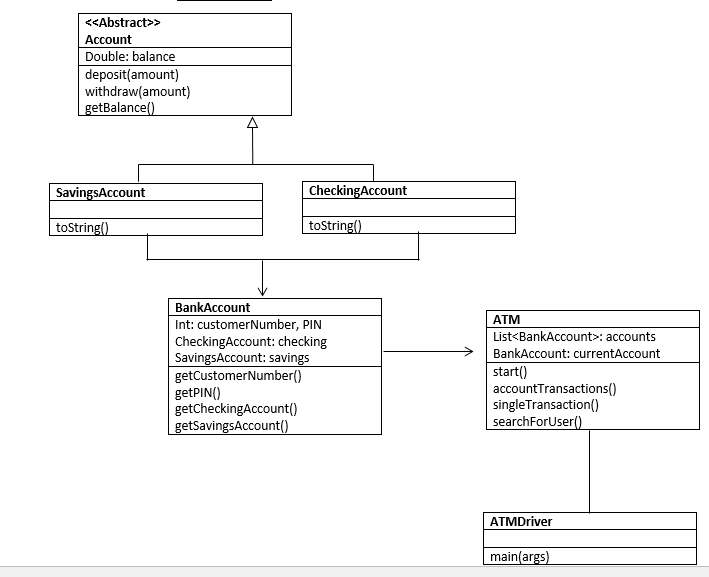
|  |
| --- |
| **Account** |
| Deposit amount  Withdraw amount |

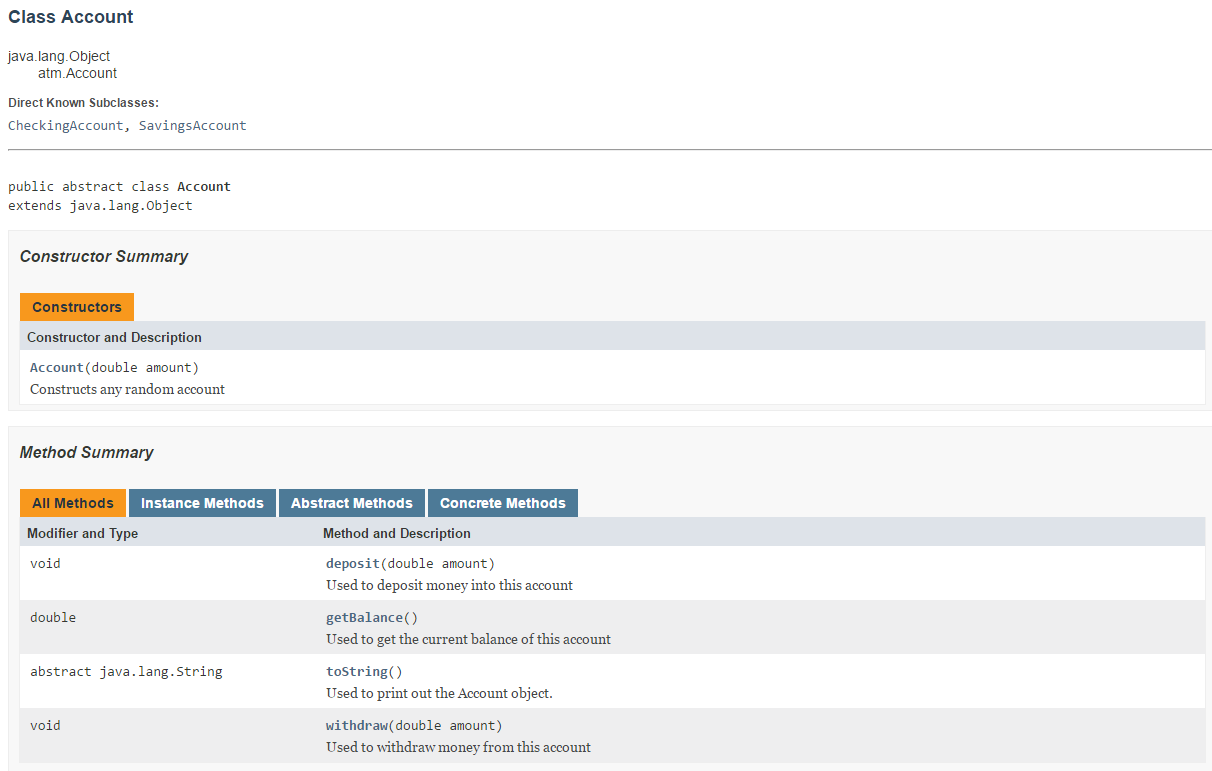
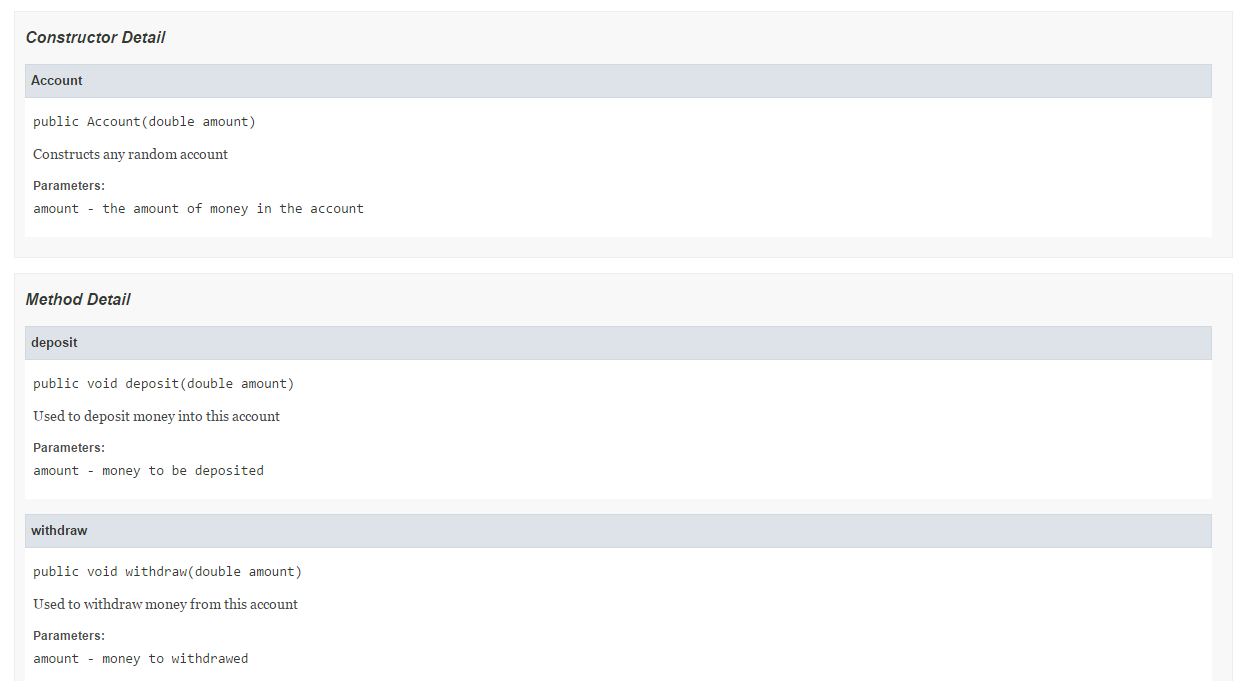
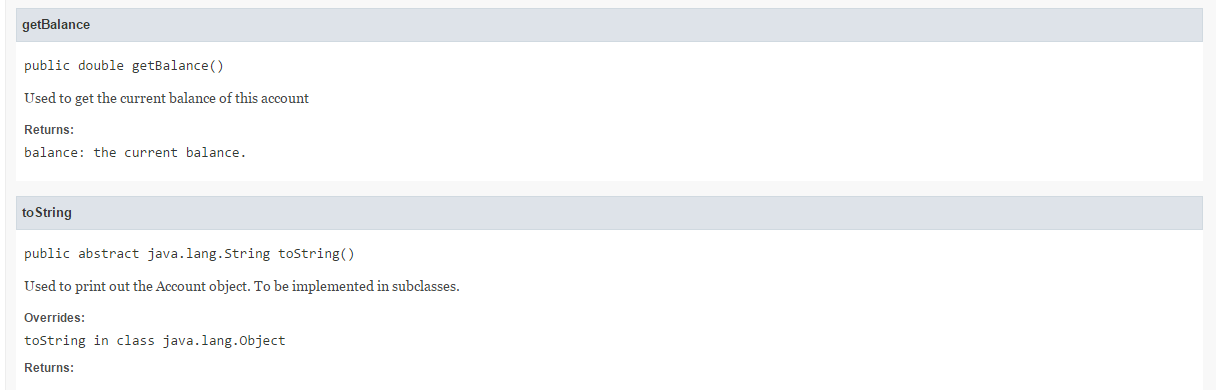
|  |
| --- |
| **SavingsAccount** |
| Deposit and withdraw amounts Account |
| **CheckingAccount** |
| Deposit and withdraw amounts Account |

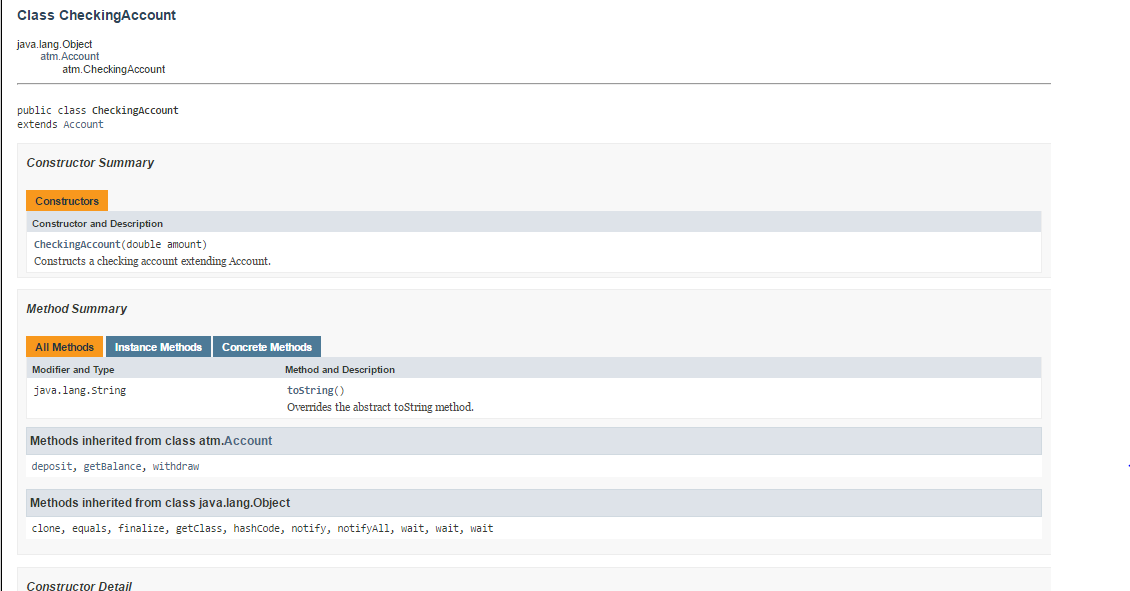
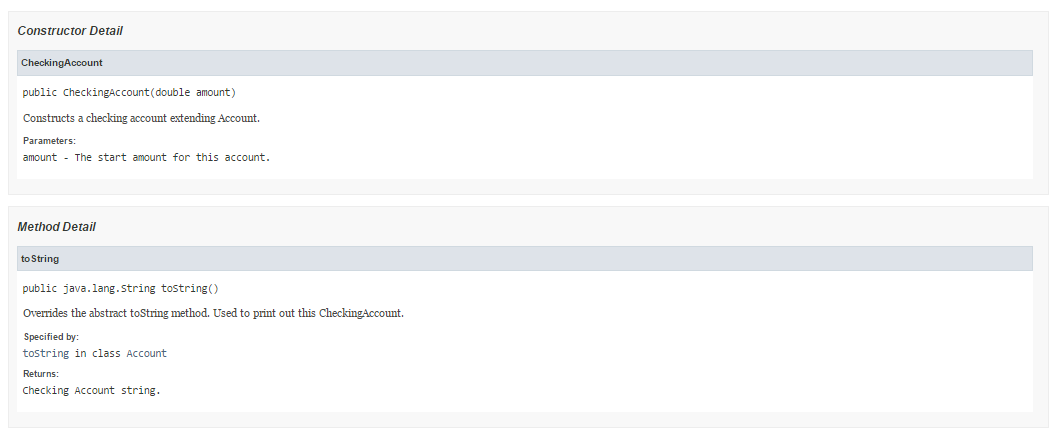
|  |
| --- |
| **BankAccount** |
| Access customer number and PIN  Access the checking account CheckingAccount  Access the savings account SavingsAccount |

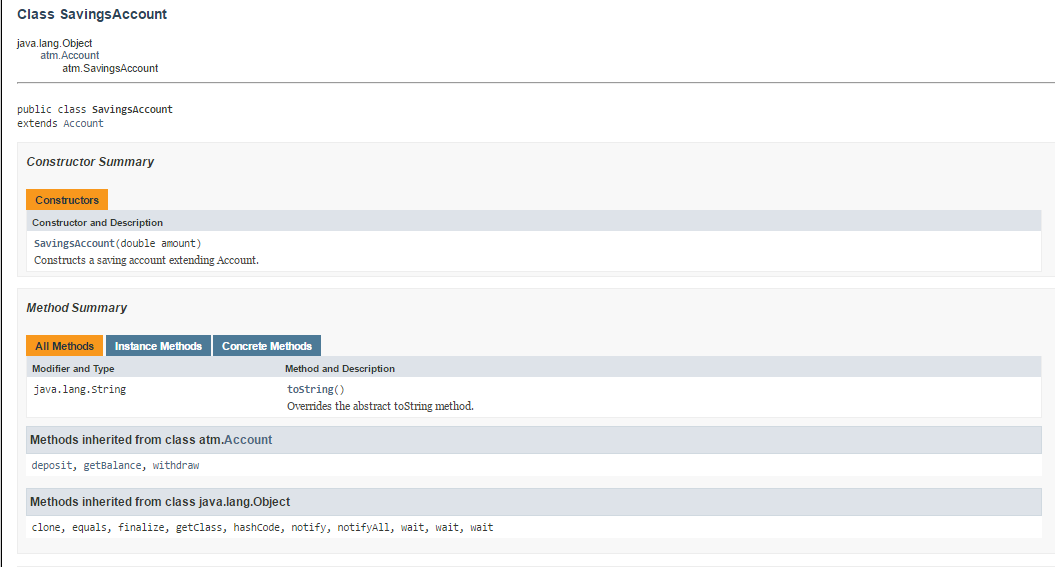
|  |
| --- |
| **ATM** |
| Access account using PIN and customer number BankAccount  Access either checking or savings account CheckingAccount  with withdrawing or depositing SavingsAccount |

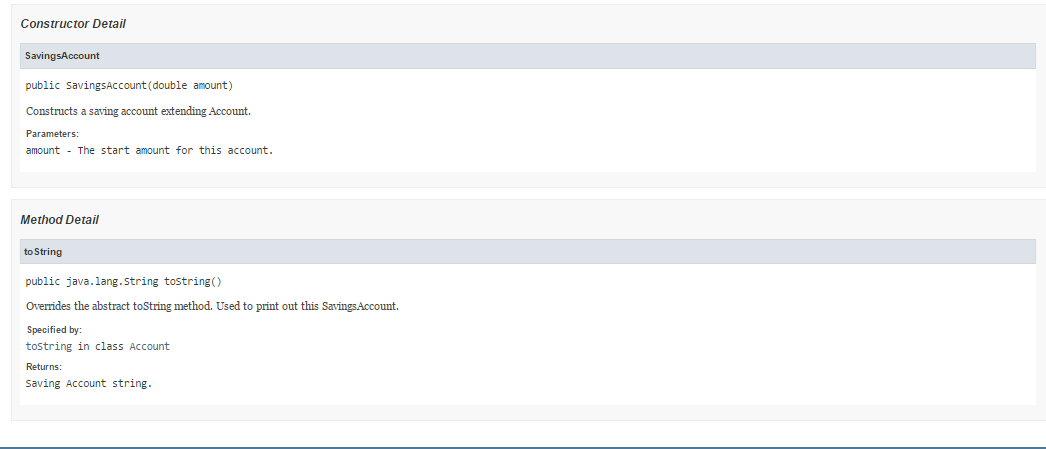
|  |
| --- |
| **ATMDriver** |
| Access the ATM ATM |

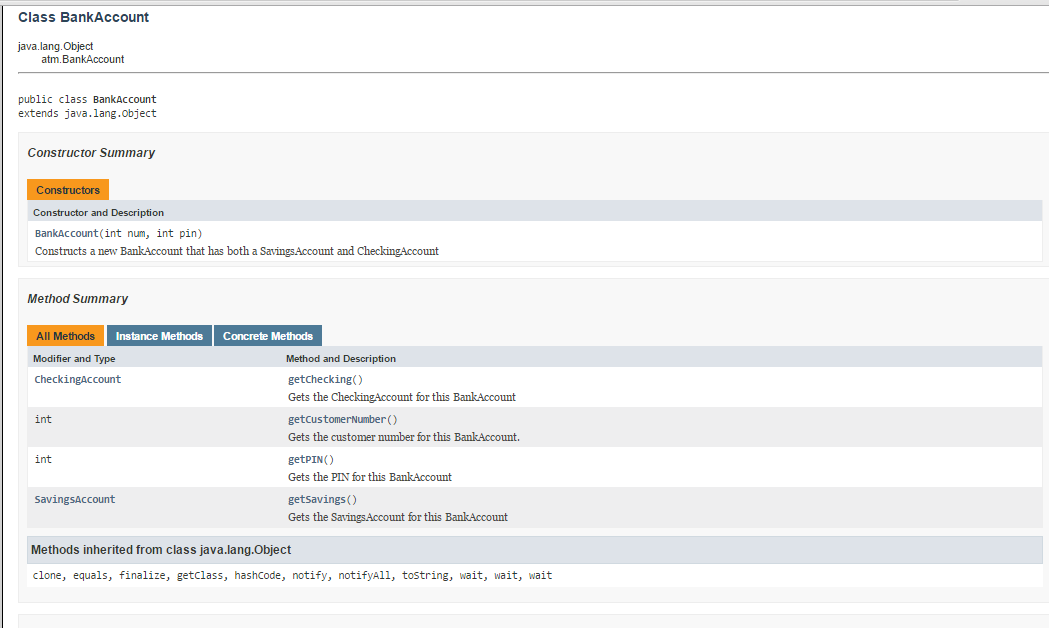
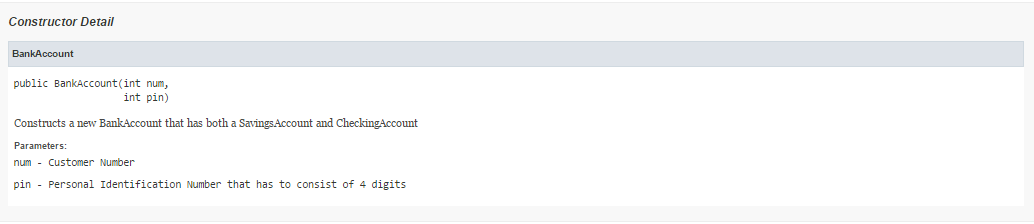
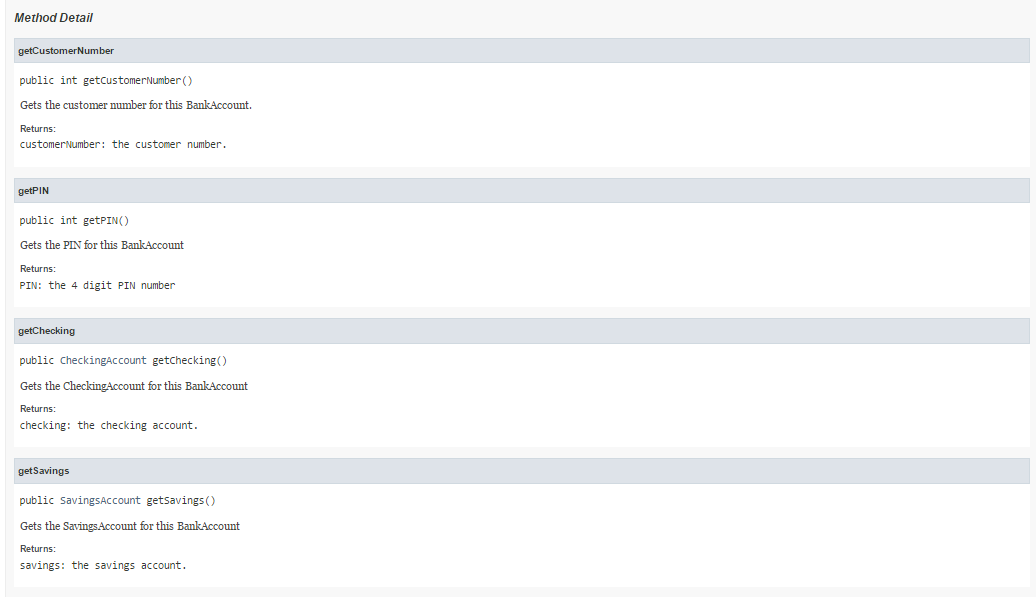
1. UML Diagram:
2. Class Documentation:

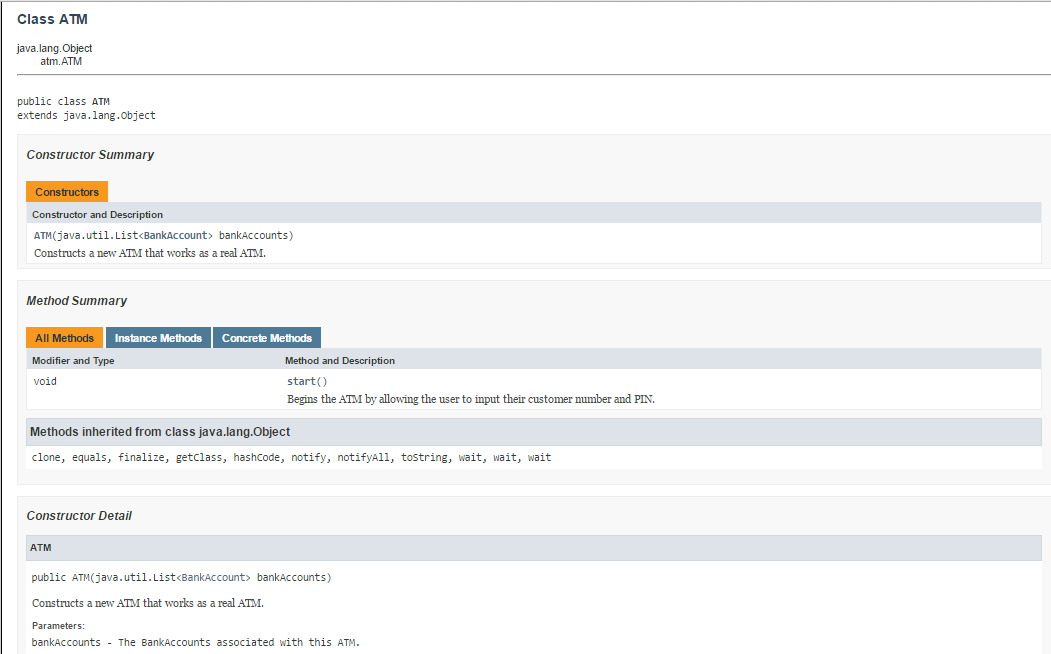
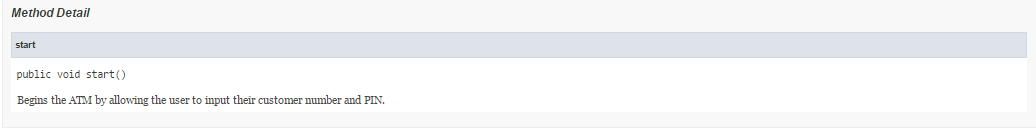


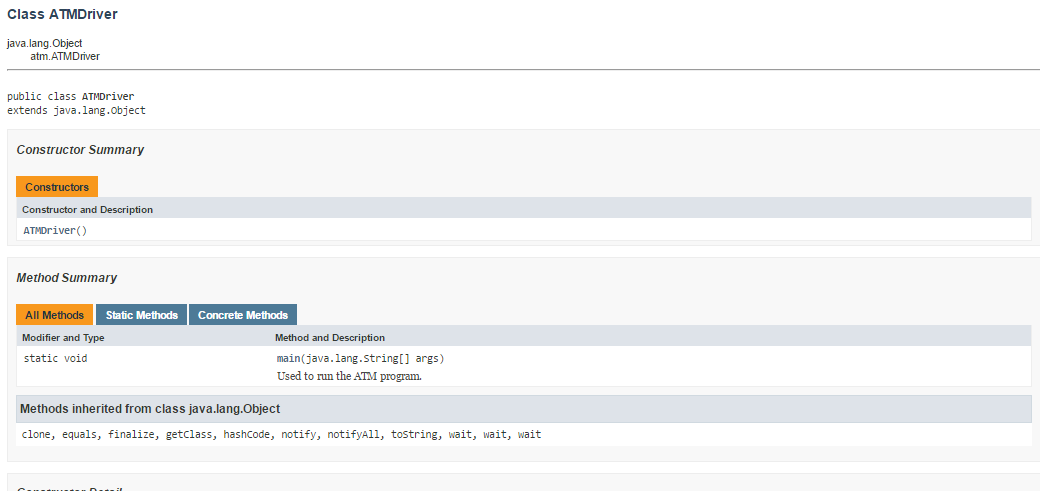
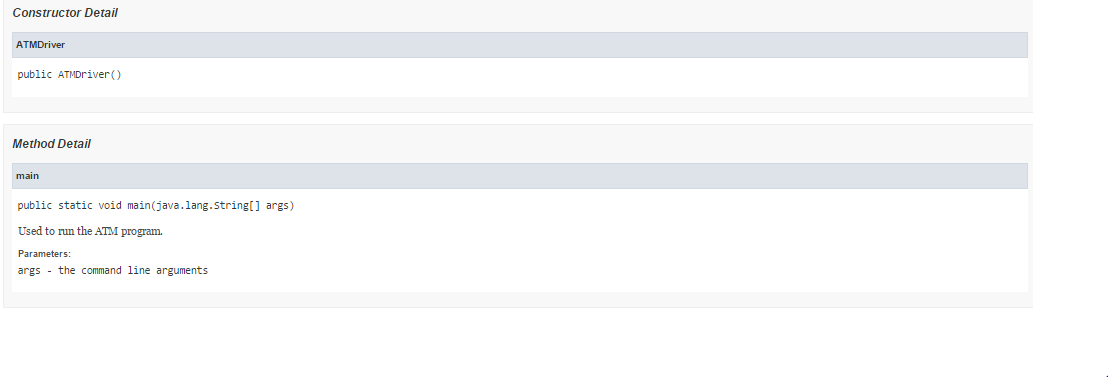






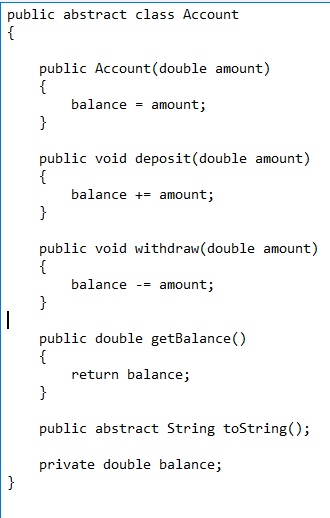




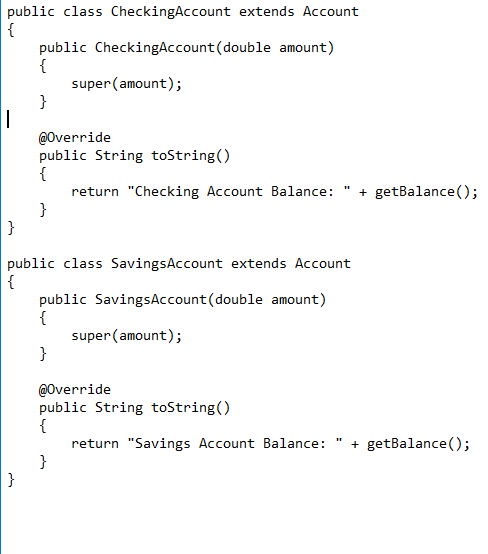


1. Implementation/Code

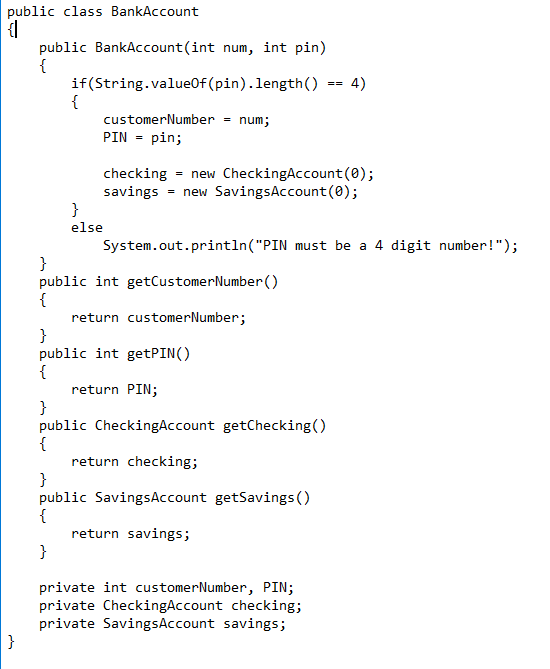
The abstract class Account is the core for this project. The class represents any account, whether it is a checking, savings, or piggy bank account. The constructor accepts a double that represents the initial amount for this specific account. The class also has an accessor that accesses the balance for the account. The toString method is abstract which means all subclasses for Account must have and implement this method. The 2 methods, deposit and withdraw, do as their names say they do. The deposit method simulates depositing money for the account by accepting a parameter that is the amount to be deposited and adding it to the balance. The withdraw method simulates withdrawing money for the account by accepting a parameter that is the amount to be withdrawed and subtracting it from the balance.

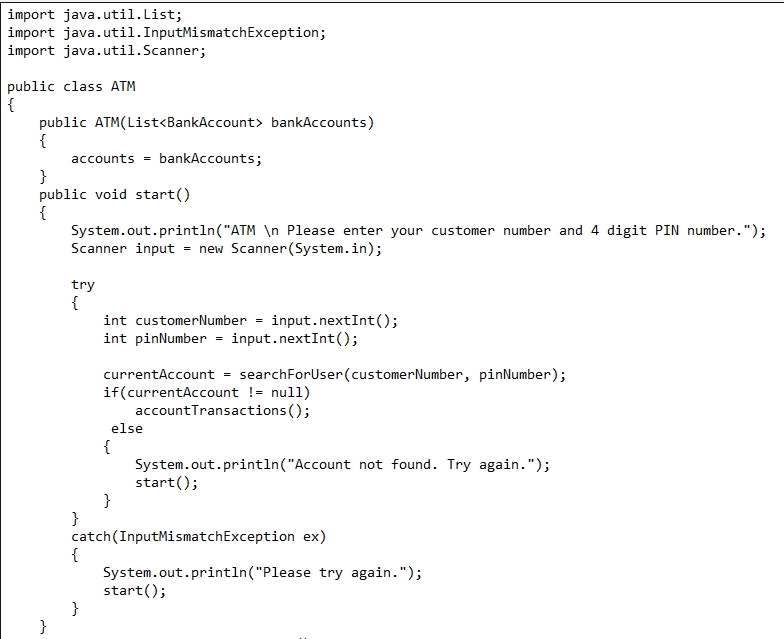
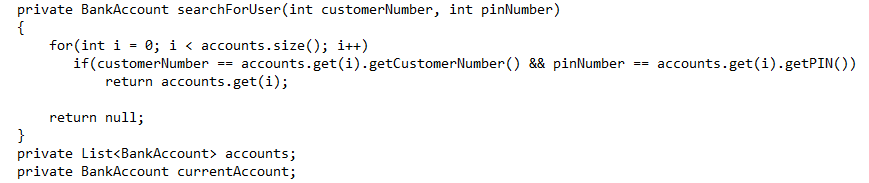


The CheckingsAccount and SavingsAccount classes are almost exactly alike. They both extend the Account class. Their constructors accept an initial amount of money as their parameters. The constructors have a call to super with the parameter amount as the parameter for the call to super. The toString method is almost exactly alike except one says that the object is a Checking Account while the other says it is a SavingsAccount. Both display the balance for their respective accounts.

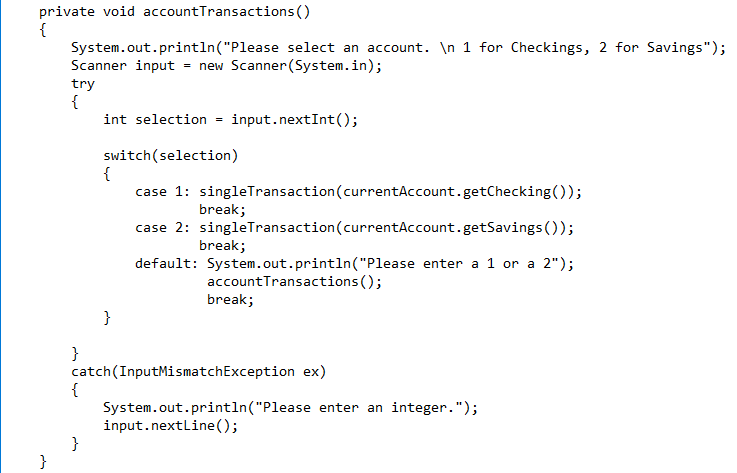


The BankAccount class represents a real life bank account. The constructor accepts a customer number and a PIN as it’s parameters. Note that the constructor checks if the PIN is 4 digits long. The instance variables for this class consist of the costumer number, PIN, as well as a checking and savings account. The class has accessors to access the customer number, PIN, CheckingAccount and SavingsAccount.

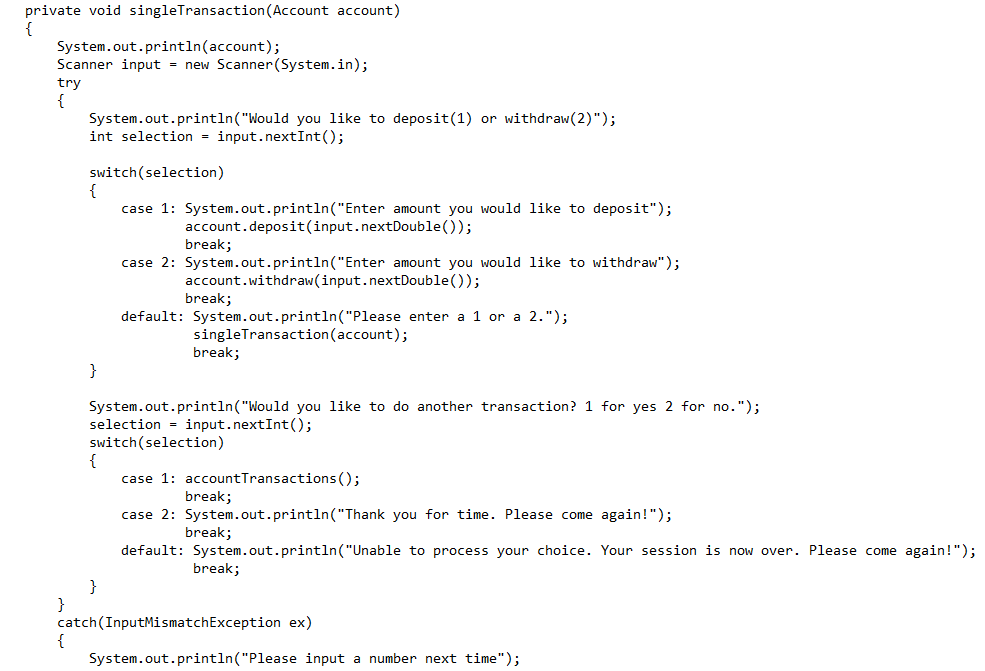


The ATM Class is constructed using the ATM constructor which accepts a List of BankAccounts as it’s parameter. The start() method, as explained in the documentation, starts the ATM program. The method starts by asking the customer to input their customer number and PIN. The method then searches for the user by calling the searchForUser method. If a user is found, the method begins the transaction for that account. If an account is not found, the ATM asks the user to try again. The 2 instance variables used represent the accounts associated with this ATM and the current account that transactions are being made from, respectively.

The accountTransactions() method begins the transaction for the current BankAccount. It begins by asking the user to select which account to run transactions off of. If the user makes any other selection, the ATM will ask the user to try again. Once an account is selected, the singleTransaction method will be called.



The singleTransaction method accepts an Account as it’s parameter. The method runs transactions off of this Account. The method first asks if they would like to withdraw or deposit money into their account, any other choice would make the ATM ask again. Whether a deposit or withdrawal is chosen, the ATM will ask the user to input the amount of money they would like to do their transaction with. Once the transaction is finished, the ATM will ask if the user wants to do another transaction. If yes is chosen, then the method will be called again, if no then the ATM will power off.





1. Output:

