CISC 327 Assignment 2

Breaking Bank

Scott Wallace 10051890

Brad Guner 10059112

**Design Document**

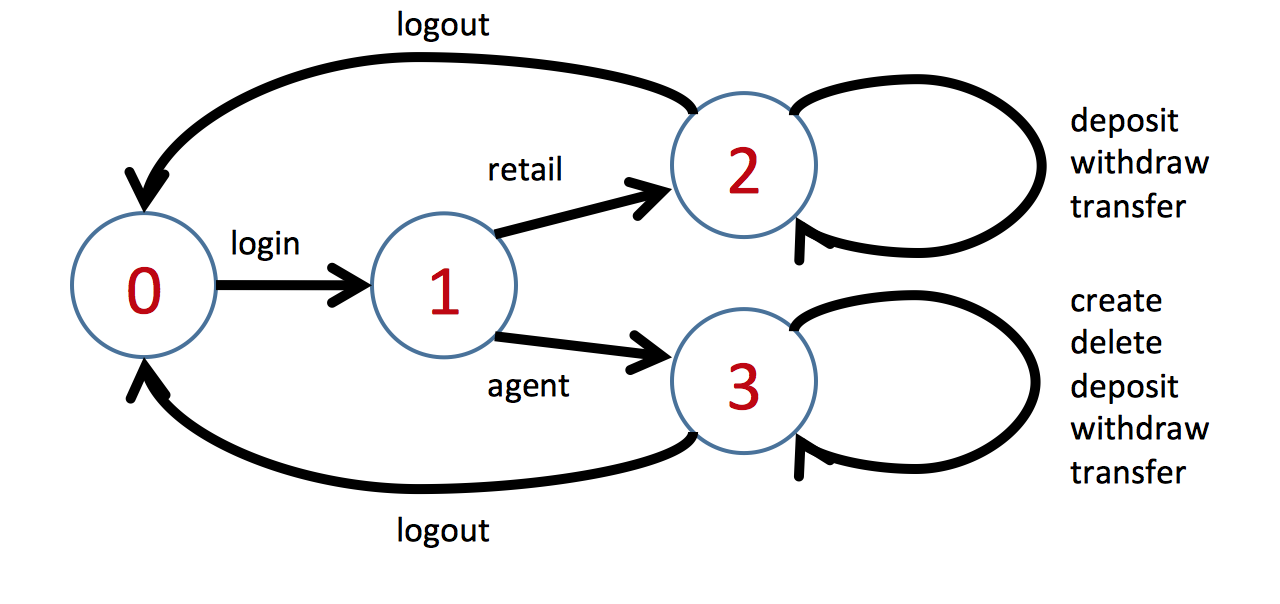
For our architecture we have chosen to go with a Main program, which contains one function, and with 2 classes that handle agent and retail. When our program first runs the main program opens our current accounts file (to be accessed during transactions), and then executes our openBankingSystem function, this function is designed to handle the system login, which type of day to run, and restarting the system after logout. However the function itself does not take the logout command as input, rather receives a logout signal from either a retail or agent day, and then proceeds to restart the system at the pre-login stage.

Our Agent and Retail classes are quite similar in construction and how they operate. The openBankingSystem function instantiates one of these classes and then runs either the runAgentDay or runRetailDay method within the class based on which type was instantiated. These methods are meant to be running to receive any sort of transaction that might occur during a banking day. This is also where the logout command will be received and then returns a logout signal back to openBankingSystem for it to reboot the system.

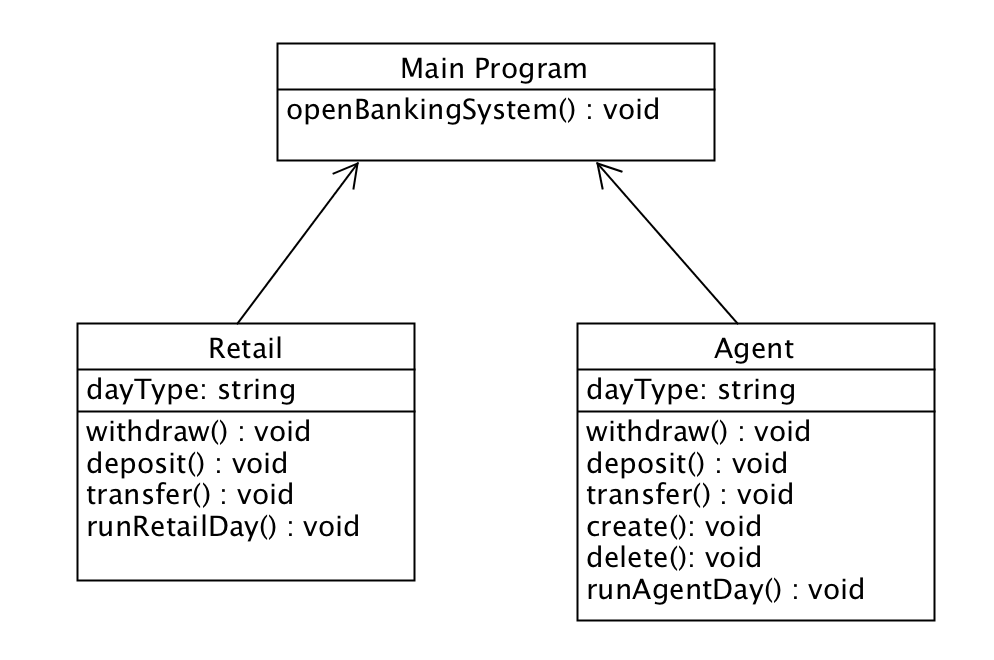
This is architecture is modeled after the 0 to 3 stage architecture (See Figure 1) modeled during class. We decided to implement a similar design because it would make the simplest solution. The 0 and 1 phase of the design are handled by the openBankingSystem function in our main program. Our Retail and Agent classes represent the 2 and 3 phases respectively. The logout functionality is handled in a similar way with either stage 2 or 3 receiving the logout command and starting over in stage 0.

Figure 2 is a simple UML diagram of the programs structure. We’ve taken some liberties in UML standards to illustrate the structure. Main program isn’t actually a class, because of the way python operates. We have chosen to represent it this way to show that openBankingSystem is apart of the main program. Agent and Retail are classes as shown with all the methods they have. The arrows pointing from these classes to main program are not hierarchy but rather symbolize the relationship between main, agent and retail. Since openBankingSystem instantiates one of these classes, we felt it appropriate to represent it in this way.

**Figure 1**

****

**Figure 2**

****

|  |  |  |
| --- | --- | --- |
| **Class** | **Method** | **Function** |
| Main Program | openBankingSystem | Handles login, agent/retail instances (Stage 0 and 1, from Figure 1) |
| Retail | withdraw |  |
| Retail | deposit |  |
| Retail | transfer |  |
| Retail | runRetailDay |  |
| Agent | withdraw |  |
| Agent | deposit |  |
| Agent | transfer |  |
| Agent | create |  |
| Agent | delete |  |
| Agent | runAgentDay |  |