CS320 Assignment 3.2: OO Design & UML – Part II: Sequence, Class & State Diagrams

Instructor: Xinghui Zhao Due: 11:59pm on 11/11/2016

1 Introduction

This assignment requires individual submissions, i.e., you must work on this assignment **independently**, no team collaboration is allowed.

The purpose of Assignment 3 is to help you practice *object-oriented design* with UML, as well as code generation/implementation based on the design. Assignment 3 consists of two parts, and this is the Part II.

2 Reverse Engineer an ATM Machine

Reverse engineering, also called back engineering, is the processes of extracting knowledge or design information from anything man-made and re-producing it based on the extracted information. When applied to software, reverse engineering means the process of analyzing a subject system to create representations of the system at a higher level of abstraction.

In this assignment, you will reverse engineer an ATM machine using UML. An example ATM machine is shown in Figure 1.

Figure 1: ATM Machine

In Part I of this assignment, you have designed the use case diagram, and activity diagram(s) for your ATM machine. In Part II, you will complete the design, and implement an ATM simulator based on your design.

To complete your UML design, follow the steps below:

1. Choose an UML tool to use.

2. Design **sequence diagrams** for the main use cases (scenarios). Here, at least you need to design two sequence diagrams, one for *deposit*, and one for *withdraw*. Based on your use case diagram, you might need more than 2.

3. The sequence diagrams you developed in the first step should indicate multiple objects/-classes are needed in order to complete the main scenarios. Using these objects/classes as a starting point, design a detailed **class diagram**. You may add classes if needed.

4. ATM is a typical event-driven system. Design a **state diagram** for the ATM.

3 Implementation

Now you have completed the design of an ATM machine. You should write code based on your design. For this step, you can choose any object-oriented programming language. To implement the ATM simulator, follow the steps below:

1. **Code Skeleton:**The implementation starts with the class diagram. Depending on the UML tool you are using, you may or may not get code generation. If your UML tool generates code for you, you may start with that. If not, you will need to write code skeleton based on your class diagram.

2. **Classes:** Implement the main operations in your classes. When you implement these operations, you may need to check your sequence diagrams to make sure that you write code based on your design.

3. **Data:** Your ATM should be initiated with a fixed amount of cash in it. You may either hardcode it, or take a parameter from the user. For the account database, you can simulate it with fake data, e.g., a txt file which contains account number and balance, etc. You will not be able to interact with a bank after all.

4. **User Interface:** For this ATM simulator, you do not need to implement a GUI. Instead, you may just take user input from command line, and display information on the screen. However, if your ATM supports *print receipt* function, try to design the output a bit differently from the *no receipt* option, i.e., display information in a nicer format with date, time, etc.

5. **Main Class:** Write a main class, which will serve as the entry point of your system. The code should be very simple: create an ATM, initialize and start it. The ATM should then operate as expected.

4 Submission

This assignment is due at 11:59pm on 11/11/2016. Late submissions are subject to a 20% penalty. **Late submissions will not be accepted after 3 days past the deadline, in this case 11:59pm on 11/14.**

You should submit the following files:

1. Sequence diagrams;
2. Class diagram(s);
3. State diagram;
4. Source code;
5. A Readme file which explains how to run your program.

5 Grading Scheme

This assignment will be graded out of 60. For your information, the grading scheme is shown in the following table.

Items	Percentage
Sequence diagrams	20%
Class diagram(s)	20%
State diagram	20%
Implementation correctness	20%
Implementation reflects design	10%
Code readability	10%

6 Go Extra Mile

Now you have successfully implemented an ATM machine. The next step would be to test the system. Try to come up with a good plan for a comprehensive testing.