CPSC2108 Data Structures, Spring 2014

ASSIGNMENT 2 – DUE BEFORE MIDNIGHT, WEDNESDAY, FEB. 19

The purpose of this assignment is to refresh your understanding of object oriented program design and implementation in Java, and the use of UML diagrams to represent object oriented design.

Problem Statement

Design the classes for a banking program that simulates the operation of your local bank. It should have the following collection of classes.

Class `Account`

Data fields `customer` (type `Customer`), `balance`, `accountNumber`, `transactions` array (type `Transaction []`). Allocate an initial `Transaction` array of size 20, and provide a `reallocate` method that doubles the size of the `Transaction` array when it becomes full.
Methods: `getBalance`, `getCustomer`, `toString`, `setCustomer`

Class `SavingsAccount` extends `Account`

Methods: `deposit`, `withdraw`, `addInterest`

Class `Customer`

Data fields: `name`, `address`, `age`, `telephoneNumber`, `customerNumber`
Methods: Accessors and modifiers[1] for the data fields plus the additional abstract methods `getSavingsInterest`, `getCheckInterest`, and `getCheckCharge`.

Classes `Senior`, `Adult`, `Student`

All these classes extend `Customer`. Each has constant data fields `SAVINGS_INTEREST`, `CHECK_INTEREST`, `CHECK_CHARGE`, and `OVERDRAFT_PENALTY` that defines these values for customers of that type, and each class implements the corresponding accessors.

Class `Bank`

Data fields: accounts array `(type Account[]`. Allocate an array of size 100 and provide a reallocate method.

[1] See page 636 of textbook for descriptions of **accessor** and **modifier** methods

Methods: `AddAccount, makeDeposit`, `makeWithdrawal.`, `getAccount.`

Class `Transaction`

Data fields: `customerNumber`, `transactionType`, `amount`, `date` and `fees` (a string describing unusual fees)

Methods: `processTran`.

Before you start to code, draw a UML diagram of your classes that reflects the class hierarchy and the use of a class instance by instances of another class.

After implementing these classes, write a driver program (a separate class) for testing the above classes. It should be able to open accounts, initiate transactions and print a transactions statement similar to a regular bank account statement containing details of a customer with the list of transactions performed.

What to submit

The following items are required:

The project .jar file (make sure to include source code)

The UML diagram of your design, drawn using MS Visio or a similar tool (hand-drawn diagrams are not acceptable)

Screen shot of test execution (you can use MS Windows snipping tool for this)

Include all required items in a single zipped folder named *<your last name>_<your initial>_Assignment02.zip*. It is your responsibility to make sure you jar file can be extracted and the code runs on a different computer.

Submit it using the CougarView Assignment 2 drop box before midnight, Wednesday, February 19. DO NOT submit multiple files.

Grading Rubric

All classes implemented correctly according to specifications - 70 points

Driver program allows user to test the implementation by creating customers, opening accounts, performing transactions and printing a transactions statement – 20 points

UML diagram – 5 points

Documentation (follow javadoc commenting style used in textbook program listings) – 5 points