CS212: Paradigms
Dr. Gross, Spring 2016

Homework 4 - Money, Money, Money

Repository Stuff

1. In your **homework** repo, create a new package called edu.blackburn.cs.cs212sp16.lastname.bank2
2. Copy and paste the files from the **lecture** repository package edu.blackburn.cs.cs212sp16.bankpolymorphism
3. Click on those files in NetBeans and go to Team->Add
4. Go to Team->Commit… and enter "created copy of base Java classes"
5. Copy and the file bank.uml from the **lecture** repo documents package to your **homework** repo documents package, and rename it: bank-lastname.uml
6. Click on that file in NetBeans and go to Team->Add
7. Go to Team->Commit… and enter "created copy of UML file"

Part I:

1. Fix the design: it's missing some things
2. Fix the code so it works
3. In Runner fill the array allAccounts with five SavingsAccounts and five CheckingAccounts
4. Print each account (should savings and checking accounts print the same?)
5. Immediately after printing each Account, make a debit or credit on that Account
6. Print the account again, immediately below
7. Make sure your implementation still matches your design

Part II: Create a Transaction Class

Banks don't just keep track of your balance; they keep track of every transaction (debit, credit). A transaction should just be a record of the old balance, the amount of the change (in positive numbers for a deposit, or negative numbers for a withdrawal), and the final balance.

1. Add Transaction in the UML
2. Which class's objects should keep track of the Transaction objects?
3. Which methods will create them?
4. Where will you store transactions? (Hint: Account)
5. How will you store them? (Hint: an array, not a good long-term solution)
6. What methods/attributes will you need?
7. Implement Transaction
8. Implement a method to get a list of all transactions for an account

Part II: Create a Loan class

1. Add a Loan class to your UML
2. It's going to inherit from a superclass, as you can guess, but from what, and why?
3. Add any methods/attributes you need; keep track of the interest rate, but don't use it
4. Implement the class; is there a method here that doesn't make sense?
5. In Runner, create a Loan
6. Print the Loan object
7. Randomly generate a (positive) Money object and print it
8. Deposit the new Money object into the Loan
9. Print the loan
10. Repeat until the loan is paid off (you'll go over, and that's fine)
11. Get a list of all the Transaction objects for the Loan object, and print them out

Make sure all of your files are added, and then commit them, and push them. Please do NOT push code that doesn't compile.

Rubric

Standards/comments	10
Time estimate/accounts	5
Class diagram correct	10
Implementation is object-oriented (no static)	5
Subclasses designed/ implemented correctly	20
Correctly store/access the allAccount elements	20
Loan payoff and Transactions work correctly	20
All classes override toString() method	10