Arizona State University - CSE205
Assignment #5

Due Date

Friday, February 2nd, 5:30pm

Important: This is an individual assignment. Please do not collaborate.

No late assignment will be accepted.

Make sure that you write every line of your code.

Using code written by someone else will be considered a violation of the academic integrity and will result in a report to the Dean's office.

Minimal Submitted Files

You are required, but not limited, to turn in the following source files:

Assignment5.java (Download this file and use it as your driver program for this assignment. You need to add more code to complete it.)
BankAccount.java
SavingsAccount.java
CheckingAccount.java
CreditcardAccount.java
BankAccountParser.java

Requirements to get full credits in Documentation

1. The assignment number, your name, StudentID, Lecture number/time, and a class description need to be included at the top of each class/file.
2. A description of each method is also needed.
3. Some additional comments inside of methods (especially for a "main" method) to explain code that are hard to follow should be written.

You can look at Java programs in the text book to see how comments are added to programs.

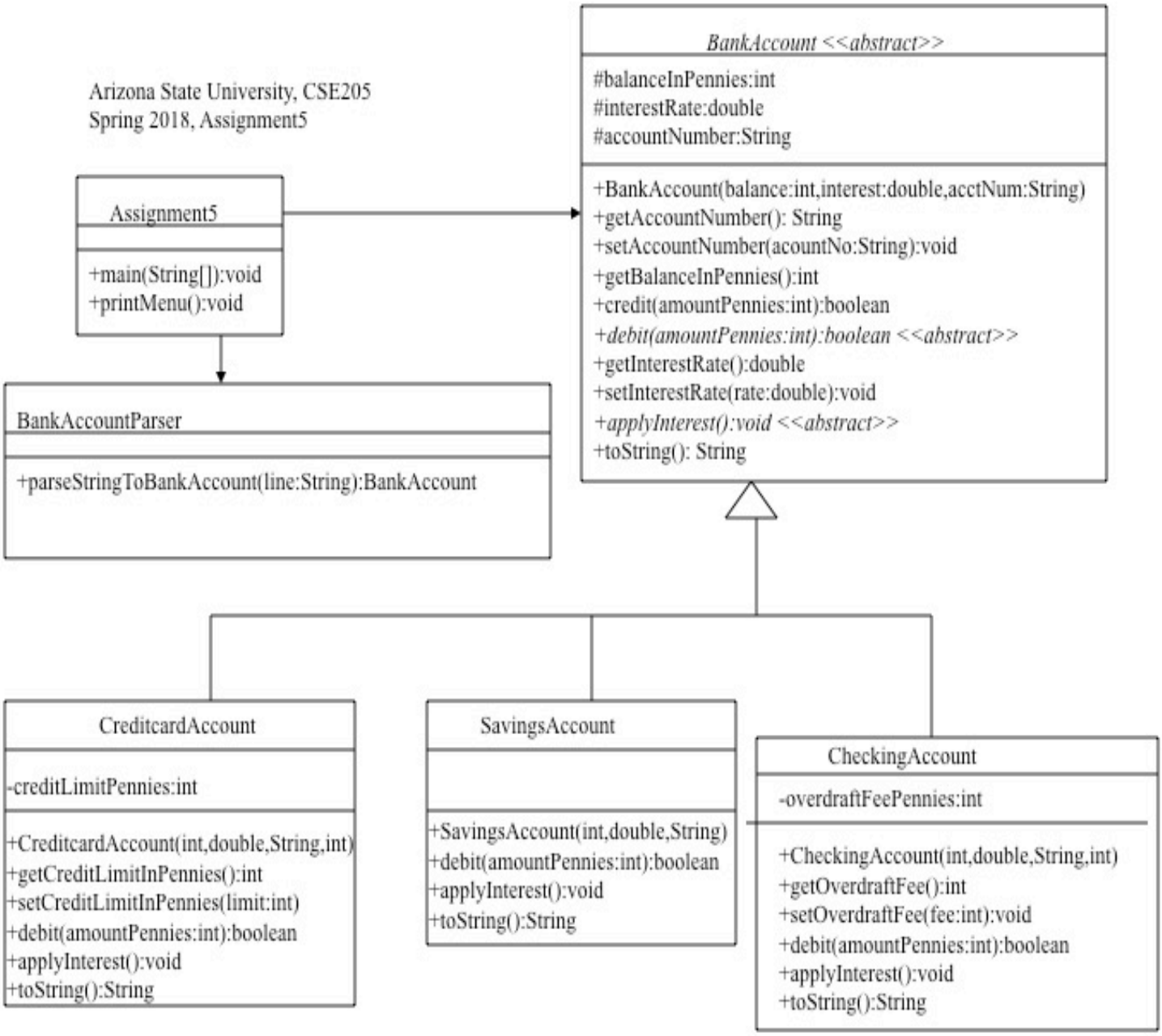
Skills to be Applied

In addition to what has been covered in previous assignments, the use of the following items, discussed in class, will probably be needed:

Inheritance
The *protected* modifier
The *super* Reference
Abstract class
NumberFormat/DecimalFormat
ArrayList

Program Description

Class Diagram:



In Assignment #5, you will need to make use of inheritance by creating a class hierarchy for bank accounts.

BankAccount is an abstract class, which represents the basic attributes of any bank account in a bank. It is used as the root of the bank account hierarchy. It has the following attributes (should be protected):

Attribute name	Attribute type	Description
balanceInPennies	int	The balance of a bank account in pennies
interestRate	double	The interest rate of a bank account
accountNumber	String	The number/ID of a bank account

The following constructor method should be provided to initialize the instance variables. Please refer to the UML class diagram for their parameter types, and return type.

public BankAccount(int balance, double interestRate, String accountNum)

The instance variables balanceInPennies, interestRate, and accountNumber are initialized to the value of the first parameter, the second parameter, and the third parameter

respectively.

The following accessor method should be provided for its instance variables::

getAccountNumber, getBalanceInPennies, getInterestRate

The following mutator method should be provided for instance variables::

setAccountNumber, setInterestRate

The BankAccount also has an abstract methods (which should be implemented by its child classes:

debit, applyInterest

The following public methods should be provided:

public boolean credit(int amountInPennies)

The parameter credit amount in pennies will be added to the balance, and return true if successful. If the credit amount is not positive, it should not add it and return false.

public String toString()

toString method returns a string of the following format:

\nAccount ID:\t:\taccount591\n
Balance\t\t:\t150.00\n
Interest rate\t\:t0.02\n

You should make use of the DecimalFormat class (in java.text package) to format the balance and interest rate. Also, the balance in pennies needs to be divided by 100 to display.

SavingsAccount class

SavingsAccount is a subclass of BankAccount class.

The following constructor method should be provided:

public SavingsAccount (int balance, double interest rate, String accountNum)

The constructor of the parent class should be called using the first, second, and third parameters to initialize its balanceInPennies, interestRate, and accountNumber.

The following methods should be implemented:

public boolean debit(int amountInPennies)

A savings account cannot have a negative balance. The debit method will return false if an attempt to overdraw the account is made. Otherwise, the debit amount will be subtracted from the balance, and return true.

public void applyInterest()

When applying interest, interest amount is calculated by multiplying the balance by the interest rate, and interest amount is added to the balance. Interest will not be applied to any account with a balance of zero.

Also, the following method should be implemented:

public String toString()

The toString() method inherited from BankAccount class should be used to create a new string, and display a savings account's information using the following format:

\nAccount type\t:\tSavings\n
Account ID\t:\taccount591\n
Balance\t\t:\t150.00\n
Interest rate\t:\t0.03\n\n

This toString method should make use of the toString method of the parent class.

CheckingAccount class

CheckingAccount is a subclass of BankAccount class. It has the following attribute in addition to the inherited ones:

Attribute name	Attribute type	Description
overdraftFeePennies	int	The overdraft fee of the checking account.

The following constructor method should be provided:

public CheckingAccount (int balance, double interestRate, String accountNum, int overdraftFeeInPennies)

The constructor of the parent class should be called using the first, second, and third parameters to initialize its balanceInPennies, interestRate, and accountNumber. Then it should also initialize its overdraftFeePennies using the value of the 4th parameter.

The following methods should be implemented:

public boolean debit(int amountInPennies)

Any CheckingAccount debit that ends with a negative balance will incur an overdraftFee (i.e. the overdraftFee amount will be subtracted from the balance in addition to the debit amount), and return false. Otherwise, the debit amount will be subtracted from the balance, and return true.

public void applyInterest()

When applying interest, interest amount is calculated by multiplying the balance by the interest rate, and interest amount is added to the balance. Interest will not be applied to a Checking account with a negative or zero balance.

Also, the following method should be implemented:

public String toString()

The toString() method inherited from BankAccount class should be used to create a new string, and display a savings account's information using the following format:

\nAccount type\t:\tChecking\n
Account ID\t:\taccount591\n
Balance\t\t:\t150.00\n
Interest rate\t:\t0.02\n
Overdraft fee\t:\t10.50\n\n

This toString method should make use of the toString method of the parent class.

CreditcardAccount class

CreditcardAccount is a subclass of BankAccount class. It has the following attribute in addition to the inherited ones:

Attribute name	Attribute type	Description
creditLimitPennies	int	The credit limit of the creditcard account.

The following constructor method should be provided:

public CreditcardAccount (int balance, double interestRate, String accountNum, int creditcardLimit)

The constructor of the parent class should be called using the first, second, and third parameters to initialize its balanceInPennies, interestRate, and accountNumber. Then it should also initialize its creditcardLimitPennies using the value of the 4th parameter.

The following methods should be implemented:

public boolean debit(int amountInPennies)

The balance of a Creditcard account cannot overrun its credit limit. The debit method will return false if an attempt to overdraw the account is made. Otherwise, the debit amount will be subtracted from the balance, and return true.

public void applyInterest()

When applying interest, interest amount is calculated by multiplying the balance by the interest rate, and interest amount is added to the balance. Interest will not be applied to a Creditcard account with a positive or zero balance.

Also, the following method should be implemented:

public String toString()

The toString() method inherited from BankAccount class should be used to create a new string, and display a savings account's information using the following format:

\nAccount type\t:\tCreditcard\n
Account ID\t:\taccount591\n
Balance\t\t:\t-150.00\n
Interest rate\t:\t0.10\n
Credit limit\t:\t2000.00\n\n

This toString method should make use of the toString method of the parent class.

BankAccountParser class

The BankAccountParser class is a utility class that will be used to create an object of a child class of BankAccount class from a parsable string. The BankAccountParser class object will never be instantiated. It must have the following method:

public static BankAccount parseStringToBankAccount(String lineToParse)

The parseStringToBankAccount method's argument will be a string in the following format:

For a savings account with its type "SA"

type/accountNumber/interestRate/balanceInPennies

For a checking account with its type "CH"

type/accountNumber/interestRate/balanceInPennies/overdraftFeeInPennies

For a creditcard account with its type "CR"

type/accountNumber/interestRate/balanceInPennies/creditLimitInPennies

A real example of this string would be:

SA/account200/0.01/502036

OR

CH/check5012/0.03/604535/2100

OR

CR/credit143/0.2/-10000/60000

This method will parse this string, pull out the information, create a new object of its corresponding child class of the BankAccount class with attributes of the object, and return it to the calling method. The type will always be present and always be either SA, CH, or CR. (It can be lower case or upper case) You may add other methods to the child classes in order to make your life easier.

Assignment5 class

In this assignment, download Assignment5.java file by clicking the link, and use it for your assignment. **You need to add code to this file.** The parts you need to add are written in the Assignment5.java file, namely for the four cases "Make Credit", "Make Debit", "Apply Monthly Interest", "List BankAcconts", and "Transfer Fund".

All input and output should be handled here. The main method should start by displaying this updated menu in this exact format:

Choice\t\tAction\n
------\t\t------\n
A\t\tAdd BankAccount\n
C\t\tMake Credit \n
D\t\tMake Debit \n
I\t\tApply Monthly Interest\n
L\t\tList BankAccounts\n
Q\t\tQuit\n
T\t\tTransfer Fund\n
?\t\tDisplay Help\n\n

Next, the following prompt should be displayed:

What action would you like to perform?\n

Read in the user input and execute the appropriate command. After the execution of each command, redisplay the prompt. Commands should be accepted in both lowercase and uppercase.

Add BankAccount

Your program should display the following prompt:

Please enter some account information to add:\n

Read in the information and parse it using the bank account parser.

Then add the new object to the bank account list.

Make Credit

Your program should read in an account number and some amount to credit, and attempt to credit the account (in the bank account list) of the given account number with the amount.

The boolean returned by credit method of such account should be assigned to the variable "operation" so that an appropriate message (credit performed or credit not performed) will be printed (please see Assignment5.java file)

Make Debit

Your program should read in an account number and some amount to debit, and attempt to debit the account (in the bank account list) of the given account number with the amount.

The boolean returned by debit method of such account should be assigned to the variable "operation" so that an appropriate message (debit performed or debit not performed) will be printed (please see Assignment5.java file)

Apply Monthly Interest

Your program should call applyInterest( ) method for each object in the bank account list so that their interest will be applied. Then it will print the message "monthly interest applied". (please see Assignment5.java file)

List BankAccounts

List all accounts in the bank account list. Make use of toString method defined in the child classes of BankAccount class.

A real example is looked like this:

Account type : Savings
Account ID : savings01
Balance : 1000.00
Interest rate : 0.02

Account type : Checking
Account ID : checking02
Balance : 2000.00
Interest rate : 0.05
Overdraft fee : 20.00

If there is no bank account in the bank account list (the list is empty), then display following:

no account\n

Transfer Fund

Your program should read in account numbers for the account to transfer from and the account to transfer to, and some amount to transfer, and attempt to transfer it. If both accounts exist in the bank account list, and the FROM account has a sufficient balance to transfer the amount, then call the credit or the debit method for the TO account and the FROM account respectively, and print the message "transfer performed", otherwise an error message will be printed.(please see Assignment5.java file)

Quit

Your program should stop executing and output nothing.

Display Help

Your program should redisplay the "choice action" menu.

Invalid Command

If an invalid command is entered, display the following line:

Unknown action\n

Test cases:

You can download each file individually:

Input

The following files are the test cases that will be used as input for your program (Right-click and use "Save As"):

Test Case #1
Test Case #2
Test Case #3
Test Case #4

Output

The following files are the expected outputs of the corresponding input files from the previous section (Right-click and use "Save As"):

Test Case #1
Test Case #2
Test Case #3
Test Case #4

Error Handling

Your program is expected to be robust to pass all test cases.