CS 250 - Fall 2021 - HW#1

Due: Sep 8, 2021, 11:59 pm

Objectives

The objectives of this assignment are:

- Familiarizing with the problem-solving steps: analysis, design, implementation, and testing.
- Practice editing, compiling, running, and revising a Java application.
- Practicing inheritance and polymorphism

Part 1. Non-programming Questions (50 points)

Q1. (18 points) Choose one correct answer for each subquestion below.

1. Which of the following statements is correct about abstract class in Java?

a. We cannot create an instance of an abstract class, but we can extend it (to define a subclass)

b. We can create an instance of an abstract class, but we cannot extend it (to define a subclass)

c. We can only define abstract methods in an abstract class, and we can extend it.

d. We can only define abstract methods in an abstract class, and we cannot extend it.

2. Which of the following statements is correct about method overloading in Java?

a. We could not define two methods with the same name in a Java class

b. Two methods are considered overloading methods if they have different parameter names

c. Two methods are considered overloading methods if they have the same name but a different number of parameters and data types

d. Two methods have different implementation

3. Which of the following statements are correct?

i) We could not define constructor methods in an abstract class

ii) We have to define a class containing abstract methods with the abstract keyword in the class header

iii) We could not define class member variables inside an abstract class

a. i) and ii) are both correct

b. Only ii) is correct

c. Only i) is correct

d. All of them are correct

4. The _____ keyword is used to refer to the superclass instance.

a. this

b. super

c. self

d. final

5. In Java, _____ is the ability to define an operation (method) that can be performed in different ways depending on the invoked objects.

a. abstract

b. static

c. polymorphism

d. final

6. In Java, a class can have a maximum of _____ superclass (parent class).

a. 0

b. 1

c. 2

d. infinite

Q2. (16 points) Drawing class hierarchy

Suppose you are asking to develop a program to manage the information of different people in a university, such as students, faculty, and staff. Faculty and staffs are employees of the university, but students are not. There are different types of students in the university: undergraduate or graduate students. The graduate students can be classified into Master or Doctoral students.

Task: Draw the class hierarchy for this application.

Note: You are required to draw the class hierarchy, not the UML diagram. So, you do not need to specify the attributes and methods for each class.

Q3. (16 pts) Class members scope

Suppose class A defines two public methods a1(), a2(), and a private method a3(). Also, class B is a subclass of class A and it defines two other public methods b1() and b2(). Suppose we define two instance variables as follows:

```
A aObj;
B bObj;
```

a. Which of the above five methods are accessible (can be called) by the aObj variable?
b. Which of the above five methods are accessible (can be called) by the bObj variable?
c. Which of the above five methods are accessible (can be called) by ((B)aObj)?

d. Suppose we change the scope of the method b1() to private. Which of the above five methods are now accessible (can be called) by the bObj variable?

Part 2. Programming Question (50 points + bonus 10 points)

2.1 Problem Descriptions

Develop a Java application to manage different types of Account in a bank. To make the scope of this program small and manageable, we only focus on two types of Accounts: Checking and Saving accounts. All accounts have some common attributes such as account number, balance, and interest rate. They also support deposit and withdraw operations. However, the Checking account and Saving account will have different interest rates: 1% for Checking Account and 5% for Saving Account.

2.2 Problem Design and Implementation Guideline

2.2.1. Account class has two instance variables: account_no and balance to store the account number and balance of such an account. It also defines the following methods:

1. **Account()**: This is a default constructor method of this Account class
2. **Account(String acc_no, double bal)**: Another constructor method that initializes the account_no and balance of this class with the input parameters
3. **String getAccountNo()**: The accessor method returns the account number of the employee
4. **void setAccountNo(String acc_no)**: The mutator method set the account number of the employee
5. **double getBalance()**: The accessor method returns the balance of the account
6. **void setBalance(double balance)**: The mutator method set the balance of the account
7. **void deposit(double amount)**: The abstract method to deposit money to this account
8. **void withdraw(double amount)**: The abstract method to withdraw money to this account
9. **void applyInterest()**: The abstract method to apply interest into the current account

2.2.2 CheckingAccount class is a subclass of the Account class. This class should define a variable named interest_rate. This class also has the following methods:

1. **CheckingAccount()**: This is a constructor method of this CheckingAccount class. You should set the interest rate with a fixed value of 1% in this method.

2. **CheckingAccount(String acc_no, double bal):** Another constructor method for CheckingAccount class that initializes the account_no and balance variables inherited from Account. Also, you should set the fixed interest rate at 1% here.
3. **void deposit(double amount)**: Implement the deposit operation for the checking account. Note, you have to make sure the amount is positive to be added to the current balance.
4. **void withdraw(double amount)**: Implement the withdrawal operation on a checking account. You have to make sure the amount is valid: positive and cannot greater than the current balance + overdraft limit. For a checking account, you can withdraw the amount more than your current balance, which is the overdraft limit. For this application, the overdraft limit is set to $100.
5. **void applyInterest():** Every month, the bank will call this method to pay interest into this account based on the applied interest rate for each type of bank account. Also, the owner needs to maintain a positive balance to receive the interest. Remember, the 1% interest rate for this checking account is the yearly interest.
6. **String toString():** You need to override this method to return a string representation of this class - return a string showing account number, current balance, and the interest rate.

2.2.3 SavingAccount class is another subclass of the Account class. This class should define a variable named interest_rate. This class also has the following methods:

1. **SavingAccount**(): This is a constructor method of this **SavingAccount** class. You should set the interest rate with a fixed value of 5% in this method.
2. **SavingAccount(String acc_no, double bal):** Another constructor method for **SavingAccount** class that initializes the account_no and balance variables inherited from Account. Also, you should set the fixed interest rate at 5% here.
3. **void deposit(double amount)**: Implement the deposit operation for this saving account. Note, you have to make sure the amount is positive to be added to the current balance.
4. **void withdraw(double amount)**: Implement the withdrawal operation on a saving account. You have to make sure the amount is valid: positive and cannot greater than the current balance. Also, the bank requires a minimum balance of $100 in a saving account, so the user cannot withdraw the money from this account which results in a balance of less than $100.
5. **void applyInterest():** Every month, the bank will call this method to pay interest into this account based on the applied interest rate for each type of bank account. Remember, the 5% interest rate for this saving account is the yearly interest.

6. **String toString():** You need to override this method to return a string representation of this class - return a string showing account number, current balance, and the interest rate.

2.2.4 Main Application class: You are required to create an application class with only one static void main method where you create different instance objects of CheckingAccount and SavingAccount to test the methods of these classes to demonstrate the inheritance and polymorphism capability. Implement using the following steps:

- Define a checking account with account number 1 and a balance of $1000
- Call deposit() method to deposit -500 to the above checking account (should show an error message)
- Call deposit() method again on the checking account 1 with an amount of 2000
- Call withdraw() method on the checking account 1 with an amount of 5000
- Call withdraw() method on the checking account 1 with an amount of 1500
- Call applyInterest() method on the checking account 1
- Call toString() on the checking account 1 to print the current state of the account
-
- Define a saving account with account number 2 and a balance of $10000
- Call deposit() method to deposit -1000 to the above saving account (should show an error message)
- Call deposit() method again on the saving account 2 with an amount of 2000
- Call withdraw() method on the saving account 2 with an amount of -2000
- Call withdraw() method on the saving account 2 with an amount of 11000
- Call withdraw() method on the saving account 2 with an amount of 5000
- Call applyInterest() method on the saving account 2
- Call toString() on the saving account 2 to print the current state of the account

Apply you finish the coding, test your program with the above inputs, and verify the final output. Make sure your program calculates correctly.

2.3 Coding Requirements

A. Source code: Write your **own** code

B. Program requirements: Your program can solve and meet all of the requirements described in 2.1 and 2.2. Do not write additional code that is not required.

C. Coding style: Your source code must be properly indented and contain adequate comments for class, methods, variables, etc.

- At the beginning of your Java class file, specify the following items:

    - Your name, the assignment number for this program, the date you create the file
    - The problem description that defines your application
    - Enumerate (list) the goals of your application
    - Enumerate (list) the inputs of your application
    - Enumerate (list) the outputs of your application
- Before each method in your class, specify the following items:
    - Specify the purpose of the corresponding method
    - Enumerate the possible inputs and outputs of the corresponding method
    - Write down the pseudocode to implement the corresponding method
- Others comments for variables, inside method implementation

2.4 Grading Rubrics

Your code in this part will be evaluated using the following grading scheme (50 points for completed functional source code + bonus 10 points for good coding style with code comments):

Points	Item
25	Correct class, variable, and method definition
10	Consistent coding-style, such as indents, curly bracket positions, etc.
15	Have all required variables and methods
10	Good and helpful comments

Assignment Submission Instructions

1. Answer Part 1 questions in a WORD, PDF, or any other document application

2. Compress the whole Eclipse source code of your project in part 2 in zip format using the Export function

Submit two items above to the HW#1 assignment folder on D2L.