CSC205
Program_2_Inheritance and Polymorphism

You may work in **pairs** (that is, as a **group of two**) with a **partner** on this Program if you **wish,** or you may work **alone**. If you work with a partner, only submit **one** lab project with **both** of your **names** in the **source code file**(s) for grading; you will each earn the **same** number of points.

Program Objectives

After completing this Program, the student should be able to,

- Write an abstract base class in Java according to Interface specifications given in UML.
- Implement base class Interfaces in java according to specifications given in UML.

Program Requirements

For this Program you are given a Java source code file (CSC205_Test_Program2.java) with a main method and supporting methods (MAKE NO CHANGES TO THIS FILE). You must write the following Java files to complete the program:

BankAccount.java CheckingAccount.java
SavingsAccount.java CreditcardAccount.java

The specifications for these files are given below (including the UML diagram on the following page). Special requirements:

All Bank Accounts

1. All accounts have balance, credit and debit amounts, and fees stored and passed as a number of pennies (**int**).
2. All debit amounts will be subtracted from the balance, and all credit amounts will be added to the balance.
3. All bank accounts have a non-negative interest rate (0.02 would be a 2% interest rate).
4. When applying interest, interest amount is calculated by multiplying the balance by the interest rate.
5. When applying interest, interest amount is **added** to the balance.
6. Interest will not be applied to any Savings or Checking account with a balance of zero or less.
7. Debit methods will return **`false`** if the transaction cannot be made because of insufficient balance or insufficient credit limit. Otherwise they will return **`true`**. A Credit will always return **`true`**.

Savings accounts

1. A savings account cannot have a negative balance.
    a. The debit method will return **`false`** if an attempt to overdraw the account is made.
2. The `getAccoutInfoAsString` method will return a string formatted exactly like this:

```
Account type  : Savings
Account #     : 101101
Balance       : $123.45
Interest rate : 1.00%
```

Checking accounts

1. Any CheckingAccount debit that ends with a negative balance will incur an overdraftFee (i.e. the overdraftFee amount will be subtracted from the balance). A checkingAccount debit will always succeed.
2. Interest will not be applied to a Checking account with a negative balance.
3. The `getAccoutInfoAsString` method will return a string formatted exactly like this:

```
Account type  : Checking
Account #     : 202202
Balance       : $2000.00
Interest rate : 2.00%
Overdraft fee : $20.00
```

Creditcard accounts

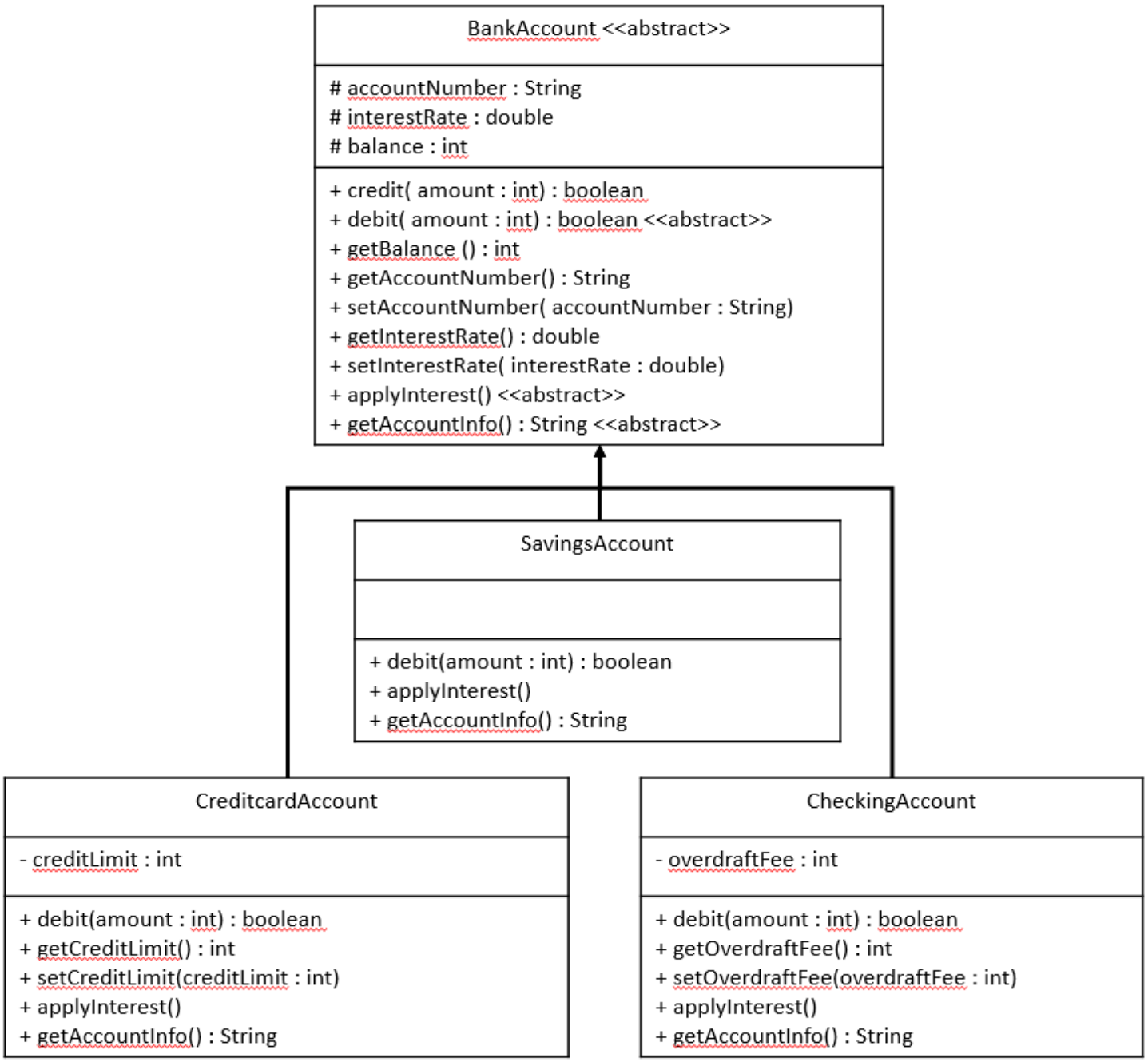
1. The balance of a Creditcard account cannot overrun its credit limit
    a. The debit method will return **false** if an attempt to overdraw the account is made.
2. Interest will not be applied to a Creditcard account with a positive balance.
3. The `getAccoutInfoAsString` method will return a string formatted exactly like this:

```
Account type  : Creditcard
Account #     : 303303
Balance       : $-275.50
Interest rate : 3.00%
Credit limit  : $10000.00
```

For all account objects:

- **accountNumber** will be initialized to **"0000-0000-0000-0000"**
- **all numeric fields** will be initialized to **0**

UML Class Diagram



Comments and formatting: Please put in an opening comment that briefly describes the purpose of your program. This should be from the perspective of a programmer instead of a student, so it should tell what the program does. It should also have your name and class on a separate line. In the code itself, indent inside the class and then again inside main. Also, please be sure that your indenting is correct, your variable names are meaningful, and there is "white space" (blank lines) to make each part of your program easily readable. This is all for "Maintainability" – and deductions for lack of maintainability will be up to 10% of your program.

Maintainability: The program should be maintainable. It should have an opening comment to explain its purpose, comments in the code to explain it, correct indenting, good variable names, and white space to help make it readable.

Please submit: all java files on Canvas under the Assignments section Program 2: Inheritance and Polymorphism. You may submit as many times as you want prior to the due date, in case you later find and fix an error, but only the last one is graded. Make sure it passes all different Test Cases.

Program_2_Inheritance and Polymorphism includes all the following java files:

BankAccount.java CheckingAccount.java
SavingsAccount.java CreditcardAccount.java

Grading Rubric

Criteria	Points
All required files are submitted	10
Each file includes a comment header with the following information: • CSC 205: <Class #> / <meeting days and times> • Program: <Program #> • Author(s): <name> & <studentID> • Description: <of the file contents>	
Code is neat and well organized	10
• Good naming conventions for all identifiers • Good use of whitespace • Methods are small and flat	
Code compiles with no syntax errors	10
• No credit will be awarded for Code passes structure tests and Code passes logic tests (Test Cases) if your code does not compile	
Code passes structure tests and Code passes logic tests (Test Cases)	70
• Credit is awarded based on percentage of Tests Cases passed	
TOTAL	**100**