csTask24 Assignment

The activities in this assignment require you to modify Java programs using BlueJ. The BlueJ project csTask24 already contains partial classes for you to work on.

All programs (class files) must be in your project, and the project must be copied to your shared Dropbox.

When questions are posed, write your responses as full sentences in the Readme.txt file in your project.

There are two parts to this assignment:

I. A Bank Account Class
II. Tracking Grades (*This part is for Extra Credit.*)

I. A Bank Account Class

1. The Account class contains a partial definition for a class representing a bank account. Study it to see what instance variables and methods it contains. Then complete the Account class as described below. In order to test your Account class before you write the driver class ManageAccounts, you will use BlueJ's ability to instantiate objects on the workbench. Even though we are dealing with dollar amounts in the project, don't worry about currency formatting.

> a. Fill in the code for method *toString*, which should return a string containing the name, account number, and balance for the account. Use BlueJ to instantiate an Account and run your new method with this object to see if it works. Note that the Account constructor is expecting a String for the name, so put " " around the name when you type it in the box.
>
> b. Fill in the code for method *chargeFee*, which should deduct a $25 service fee from the account. Use BlueJ to instantiate an Account and run your new method with this object to see if it works.
>
> c. Modify *chargeFee* so that instead of returning void, it returns the new balance. Note that you will have to make changes in two places. Use BlueJ to instantiate an Account and run your new method with this object to see if it works.
>
> d. Fill in the code for method *changeName* which takes a string as a parameter and changes the name on the account to be that string. Use BlueJ to instantiate an Account and run your new method with this object to see if it works.

2. The ManageAccounts class contains a shell of a "driver" program that uses the Account class above. Complete this class as indicated by the comments supplied in the class.

3. Modify ManageAccounts so that it prints the balance after the calls to chargeFees. Instead of using the getBalance method like you did after the deposit and withdrawal, use the balance that is returned from the chargeFees method. You can either store it in a variable and then print the value of the variable, or embed the method call in a println statement.

II. Tracking Grades

A teacher wants a program to keep track of grades for students and decides to create a student class for his program as follows:

_ Each student will be described by three pieces of data: his/her name, his/her score on test #1, and his/her score on test #2.
_ There will be one constructor, which will have one argument—the name of the student.
_ There will be three methods: *getName*, which will return the student's name; *inputGrades*, which will prompt for and read in the student's test grades; and *getAverage*, which will compute and return the student's average.

1. The Student class contains an incomplete definition for the Student class. Complete the class definition as follows:

a. Declare the instance data (name, score for test1, and score for test2).
b. Create a Scanner object for reading in the scores. Think about scope: Where are you going to use this variable? Where should it be declared?
c. Add the missing method headers.
d. Add the missing method bodies.

Before you continue, use BlueJ to instantiate a Student and run your methods with this object to see if they work. Remember that the Student constructor is expecting a String for the name, so put " " around the name when you type it in the box.

2. The Grades class contains a shell program that declares two Student objects. Use the *inputGrades* method to read in each student's test scores, then use the *getAverage* method to find their average. Print the average with the student's name, e.g., "The average for Joe is 87." You can use the *getName* method to print the student's name.

3. Add statements to your Grades program that print the values of your Student variables directly, e.g.:

```
System.out.println("Student 1: " + student1);
```

This should compile, but notice what it does when you run it—nothing very useful! When an object is printed, Java looks for a *toString* method for that object. This method must have no parameters and must return a String. If such a method has been defined for this object, it is called and the string it returns is printed. Otherwise the default *toString* method, which is inherited from the Object class, is called; it simply returns a unique hexadecimal identifier for the object such as the ones you saw above.

Add a toString method to your Student class that returns a string containing the student's name and test scores, e.g.:

```
Name: Joe Test1: 85 Test2: 91
```

Note that the toString method does not call System.out.println—it just returns a string. Recompile your Student class and the Grades program (you shouldn't have to change the Grades program—you don't have to call toString explicitly). Now see what happens when you print a student object—much nicer!