**CSC 110AB**

**Program5**

For this program, you will be the creator of a class (in Part1) and later the user of the class (in Part2). You will have to be familiar with classes, methods, passing things in and out of methods, constructors, static data/methods, exception handling, interfaces, and “aggregate” classes.

**The problem:**

You are to write a class called BankAccount, which will simulate a user’s bank account. You can use my Program5Driver.java as the “driver” that will create instances and tell them to do things (through their methods).

Then you will create your own driver called Program5.java to create other instances and tell them to do things.

**The BankAccount class: this class should implement AccountInterface and have the following:**

* **Data:** it should have the following data. I have suggested variable names to help in understanding what they do and to make the explanations clearer. You can name them anything you want (as long as the name is meaningful).
* A double called balance, which will hold the balance for that account.
* A String called name, which will hold the name for that account.
* A java.util.Date called creationDate, which will contain a reference to a java.util.Date that holds the creation date. You can declare it as a java.util.Date or import java.util.\*. But…if you import, be sure that you do not have another Date class in the same directory or the compile will use the one in the directory.
* A boolean called frozen, which keeps track of whether or not the account is “frozen” (cannot be accessed).
* A double called limit, which will hold the maximum that can be withdrawn at one time from that account.
* A double called MAXLIMIT, which will be a constant (“final”) set to 500. This will be the maximum that the withdrawal limit can be set to.
* An int called accountNumber, which will hold the account number for that account.
* A static int called howMany. The BankAccount class will use this to keep track of how many individual accounts have been created.
* **Constructors:** it should have the following 2 constructors.
* A default constructor (which receives no arguments). It should set the account’s name field to “classified” and the other fields should be set to the same initial values as the other constructor (below). If you want, you can make this work by calling the other constructor with the statement:

this(“classified”);

* A parameterized constructor which receives 1 argument – the name (a String) of the person who has the account. It should do the following:
  + Set the account’s name to the name that is passed in
  + Create the creationDate using the default constructor of the java.util.Date class. It will initialize the creationDate to the current time (from the computer clock).
  + Set frozen to false;
  + Set the limit to 300 (always, does not depend on what is passed in).
  + Add 1 to howMany (so the class knows how many accounts are now created)
  + Then set the accountNumber to howMany (if there are now 3 accounts, then this will be #3)
  + Set the balance to 0.
* **Methods:** Since BankAccount implements AccountInterface, it will automatically have to have the methods defined in the interface. It should also have the static method shown below. Therefore the following methods should be implemented.
  + - a static method called getNumAccounts(), which receives nothing and returns an int showing how many accounts have been created.
    - Also implement all of the other methods listed in the AccountInterface. Please see the comments in AccountInterface.java for their descriptions.

**Part 1:**

Write your BankAccount class and compile it (with the java.util.Date class as an “aggregate” class). To test it, you can download Program5Driver.java from Blackboard and use it as a “driver” to create accounts and tell them to do things. The expected results from running Program5Driver.java are also on Canvas as expected.txt.

**Part 2:**

Once your BankAccount class is working, create another “driver” program of your own called Program5.java. Have it implement the following functionality inside main. Notice there are 4 sections (create/use 3 accounts and then ask the BankAccount how many accounts have been created). Put each of the 4 sections into a separate try-catch block; for each one make sure that every possible type of exception can be caught (either by having multiple catch blocks or by setting it up to catch every possible type of exception). You can just print the exception if it is caught.

* Create a new account (use any variable name you want) – pass in “Brad Pitt”.
* Tell it to deposit 439.50
* Tell it to deposit 10
* Tell it to withdraw 60
* Tell it to get the balance and print “The balance is: “ and then the balance that was returned.
* Create a new account (use any variable name you want) – pass in YOUR name
* Tell it to deposit 45.00
* Tell it to withdraw 20
* Print the account information (its toString()).
* Create a new account (use any variable name you want) – pass in “Scam Artist”
* Tell it to deposit 10
* Tell it to withdraw 260
* Ask the BankAccount class how many accounts have been created and print the results (refer to the tester if needed).

**Extra Credit (optional, 6 pts):**

The variables that will hold money should be doubles (because of decimals). However, they will not automatically print out like money. After your program is working, make the following changes to have them print out like money.

1. Put the following import statement in your program (both BankAccount and Program5)

import java.text.NumberFormat;

1. Somewhere in the class (before you print things as money), put the following line:

NumberFormat money = NumberFormat.getCurrencyInstance();

1. You have created an instance of the NumberFormat class and called it “money.” Now, every time you want to print out a money variable, you can tell money (your variable) to format it so it looks like currency. Instead of using the variable in the print statement, use money.format(<the variable>). For example, if you have a variable called theDeposit, you can change it from this:

System.out.println(“\tTotal income:\t\t” + theDeposit);

to this:

System.out.println( "\tTotal income:\t\t" + money.format(theDeposit));

Please note that negative numbers will come out with ( )’s instead of a minus sign. This is the way accountants do it and it is OK to leave it like this…

**Extra Credit (optional, 6 pts)**

Create a new class called IllegalWithdrawalException. Make it a subclass of RuntimeException. It will not need any data or methods (all will be inherited). It should have one constructor which accepts a single String as an argument. Have that constructor do the same thing as its parent’s constructor would do with the same String.

Once your IllegalWithdrawalException compiles, change the BankAccount code so that it throws a new IllegalWithdrawalException instead of an IllegalArgumentException for any of the bad conditions in the withdraw method.

Finally, go back to your Part2 and change the try-catch blocks everywhere so that they will also catch an IllegalWithdrawalException. You can do this either by adding another catch block to the existing code everywhere or by changing the catch blocks so that the same catch block will the IllegalArgumentExceptions, IllegalWithdrawalExceptions, and IllegalStateExceptions.

**Comments and indenting:**

At the top of the program (class) should be a short comment which describes its purpose. 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. Indent inside while loops and for loops. Also indent inside if statements and the else part. Put a blank line and then a comment as needed to describe “sections” of code.

Finally, make sure that your variable names are meaningful.

**Handing in the program:**

Please submit your BankAccount.java, your Program5.java, and your IllegalWithdrawalException.java (if you did that extra credit) via Canvas.