CS 5000: Foundations of Programming

Fall 2022 / Assignment #5 – Objects and Classes

**A note for good coding practices:** Starting with this assignment, you need to incorporate the following concepts/practices in your code as required by the problem statement. Do not expect them to be spelled out in individual problem statement or subsequent assignments. This and subsequent assignments will be assessed for utilization of these concepts in the code.

* Apply the concept of encapsulation.
* Always declare variables as private.
* Provide setters and getters methods to allow access to class variables as per the problem statement requirements and the role/purpose of each variable in the solution.
* Always declare support methods as private.
* Allow the user to enter input values to facilitate code test-ability. DO not hard code inputs unless stated in the problem statement.
* Allow the user to re-run the code in the same session using a sentinel loop.
* Always sanitize/validate input values as soon as being entered to make sure they are “good” values (using assertions or if statements).

Develop a complete Java program for each of the following problems. Please name the programs as indicated, add proper program headers, and output labels as shown below. ***Please use only concepts and programming constructs/syntax we discuss to date.***

**Program #1 (10 points):** Design and implement a Java class called AccountYourname. The class defines the following data fields and methods:

1. Private int data field named id to store the account ID (default value is 0).
2. Private double data field named balance to store the account balance (default value is 0.0).
3. Private double data field named annualInterestRate to store the interest rate (default value is 0.0). (Assume all accounts have same interest rate. Annual interest rate is percentage such as 3.2%, thus you need to divide by 100 to get double value 0.032).
4. Private Date data field named dateCreated (an object of class Date) to store the date when the account was created.
5. Non-argument constructor method that creates a default account (with default values).
6. Constructor method that creates an account with specified ID and initial balance.
7. Add Get and Set methods for variables id, balance, and annualInterestRate.
8. Add Get method for variable dateCreated.
9. Method named getMonthlyInterestRate() that returns the monthly interest rate (i.e., annualInterestRate / 12 , formatted as percentage (%)).
10. Method named getMonthlyInterest() that returns the earned monthly interest amount (i.e., balance \* monthlyInterestRate, formatted as currency ($)).
11. Method named withdraw() that withdraws a specific amount from the account.
12. Method named deposit() that deposits a specific amount to the account.

**Program #2 (10 points):**

Write a test program(main() method in the another Class) named TestAccountYourname to create an account object and test it.

Name new Account object myAccount as follows:

- Account ID is 123456; Initial balance is $10,000

- Test Withdraw $3,500 🡪 Show the balance after this

- Deposit $500 🡪 Show the balance after this

- Print out the earned monthly interest

- Print out the date the account was created

**Program #1- additional (5 points):**

Now, add method toString()to class AccountYourname to allow the user to printout a meaningful description of an account object using all of its instance variables. For example, the following statement in the test program

System.out.print(myAccount);

on object myObject would display the account information as follows:

Account ID: 123456

Account Balance: $7,000.00

Interest Rate: 2.5%

Date Opened: Mon Apr 11 14:18:16 EDT 2022

Here is an **example** of how to write a toString()method:

//-------------------------------------------------------------------

// Returns a string representation of student object using name and studentID.

//-------------------------------------------------------------------

public String toString(){

// **name** and **studentID** are instance variables in class ***Student***

return ("The student name is " + **name** + ", and the ID is " + **studentID**);

}

**Program #2-additional (5 points):**

Now, **modify the test program** above to create 1 more account object (say myAccount2) with different initial balance values and different interest rates. **Allow the user to enter all the object values**. Test all methods on myAccount2 object in a reasonable order and display meaningful information about the object after each method call.

+ Try System.out.println(myAccount2); to check the toString() method works correctly or not.

**Submission:**Before submitting your programs, make sure you review the assignment submission requirements and grading guidelines on the course webpage. The grading guidelines explain some of the common errors found in programming assignments.

1. The assignment is due no later than **11:59 PM** on the due day posted in D2L.
2. Please compile and run your java files (only the .java files) right before you upload to the assignment submission folder in D2L.
3. Please submit one document with all result screenshots, all individual .java files.