**CPSC 233 – Team Assignment 4 – Requirements**

As a team, complete a solution for the individual assignment that you’ll use for your team submission. Then add the following constructors to your team solution for SavingsAccount.

* default constructor
* public SavingsAccount(double annualInterestRate)
* public SavingsAccount(Customer accountHolder, double balance, double annualInterestRate)

When creating constructors in a subclass make sure to:

1. Avoid code duplication through appropriate use of this() and super().
2. Not call any of the setter methods in the parent class (BankAccount). (You may use the setter methods in other methods, but not in the constructor. Each class in the hierarchy is responsible for setting their own instance variables.)

An updated test for SavingsAccount is provided. Only submit SavingsAccount.java to the D2L dropbox.

**Rubric**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 marks | 1 mark | 0 marks |
| Functionality | All tests in SavingsAccountTA4 pass. |  | One or more tests fail. |
| Design | There is no code duplication. Constructors use ‘this’ and super where appropriate. All instance variables are private. Methods are public. Instance variables are not repeated from parent class, instead appropriate methods in parent class are called. | Some code duplication or access is public when it should be private. | Some code duplication and some incorrect access modifiers or instance variables from parent are repeated in child class. |
| Legibility | Code is easy to read: good use of white space, organization of components of a class are standardly ordered (instance variable, then constructors then methods), variables names are self-documenting. | Some improvement possible but overall code is legible. | Code is difficult to read. |
| Documentation | Class and methods are javadoc’d. (Using the correct syntax, in the correct location and documents what and how to use.) Inline documentation provided where needed if needed to explain what blocks of code accomplish. | Javadoc is not in the correct location or with incorrect content OR explains how the method/class is implemented.  OR  Inline documentation missing where needed or too verbose. | Javadoc missing or inline documentation missing when some code is complex. |