Assignment #2: Implementing Classes

*Due: Friday, September 28th @ 11:55PM*

*Total Possible Points: 20*

How to Submit

* Moodle assignment (no emails or hardcopies accepted)
* Submit IDE project in ZIP or RAR format as *Assignment2.zip* or *Assignment2.rar*

Goals

* To learn how to implement Java classes
* To choose appropriate data types and meaningful names for variables
* To learn how to document your classes effectively
* To practice generating Javadoc documentation for your classes

|  |  |
| --- | --- |
| Forbidden | If you have trouble finding your Eclipse project on your computer or getting your Javadoc tool to work, take a look at these help pages:  *Opening your project folder in File Explorer/Finder (6th button on page):*  <https://jpgrady28.azurewebsites.net/Home/Docs/234>  *Generating Javadocs:*  <https://jpgrady28.azurewebsites.net/Home/Docs/235> |

## Your Task

Using the BankAccount and BankAccountTester classes from the **ch03.section\_04** code samples as inspiration, create a SavingsAccount class and a SavingsAccountTester test program class. **Your classes must meet the requirements laid out in the grading rubric**.

## Sample Output

Your test program should print this as its output:

Corey has $309.09029999999996

Sofia has $2153.78125

## Requirements/Grading Rubric

* YES = Full credit
* Partial = Half credit
* NO = No credit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Implemented**  **Successfully?** | | |  |  |
| **Requirement** | **YES** | **Partial** | **NO** | **Max Points** | **Comments** |
| ***Program Setup (3 points)*** | | | | | |
| Has a package named bank |  |  |  | **1** |  |
| Has a class named SavingsAccount |  |  |  | **1** |  |
| Has a class named SavingsAccountTester with a main() method |  |  |  | **1** |  |
| ***SavingsAccount class (9 points)*** | | | | | |
| Has three (3) private instance variables with appropriate data types:   * balance: stores the balance * interestRate: stores the annual interest rate * accountOwner: stores the name of the account's owner |  |  |  | **1** |  |
| Has two (2) constructors with correct data types added to its parameters:   * SavingsAccount(initBalance, initInterest, initHolder): initializes the balance, interest rate, and owner’s name with the given parameter values * SavingsAccount(initBalance, initHolder): initializes the balance and owner's name with the given parameter values; initializes the interest rate with a default value of 1% |  |  |  | **2** |  |
| Implements three (3) public methods with proper return types:   * getBalance(): gets the current balance * getHolder(): gets the name of the holder * addInterest(): adds interest to the balance of the account |  |  |  | **6** |  |
| ***SavingsAccountTester test program class (5 points)*** | | | | | |
| Constructs two savings accounts:   * Corey’s with $300 and the default interest rate * Sofia’s with $2000 and an interest rate of 2.5% |  |  |  | **2** |  |
| Calls the addInterest() method three (3) times on each account |  |  |  | **1** |  |
| Prints output correctly using the SavingsAccount class's methods |  |  |  | **1** |  |
| Uses clear, meaningful variable names |  |  |  | **1** |  |
| ***Documentation (3 points)*** | | | | | |
| Javadoc-style code comments for the SavingsAccount class and all public methods, including constructors |  |  |  | **1.5** |  |
| Generated complete Javadocs for the SavingsAccount class |  |  |  | **1.5** |  |

|  |  |
| --- | --- |
| Lightbulb | Again, for those who can't stand it, see Chapter 4.4 on using System.out.printf() - also called System.out.format() – and String.format().  The course's **Resources & Helpful Links** page has a few links to tutorials on string and number formatting. Browse the **Tutorials** section of the page, or do a search for "java string format" |