Alan Palayil

Anwar Adam

Rezwan Rifat

**CS 115 Fall 2019 Lab #6**

Due: **Thursday, October 17th, midnight**

Points: **20**

**Instructions:**

1. Use this document template to report your answers. Enter all lab partner names at the top of first page.

2. You don’t need to finish your lab work during the corresponding lab session.

3. ZIP your lab report and java files (if any) into a single ZIP file. Name the ZIP file as follows:

LastName\_FirstName\_CS115\_Lab6\_Report.zip

4. Submit the final document to Blackboard Assignments section before the due date. No late submissions will be accepted.

5. ALL lab partners need to submit a report, even if it is the same document.

**Objectives:**

1. (8 points) Write and test a user-defined class.

2. (12 points) Design a class for an everyday object, including required data and methods.

**Problem 1 [8 points]:**

Tasks:

Write and test a user-defined class (requiring conditions).

Write an application (client) program that uses an instance(s) of a user-defined class.

The federal income tax that a person pays is a function of the person's taxable income. The following table contains formulas for computing a single person's tax.

|  |  |  |
| --- | --- | --- |
| **Bracket** | **Taxable Income** | **Tax Paid** |
| 1 | $22,100 or less | 15% |
| 2 | More than $22,100 but $53,500 or less | $3,315 plus 28% of the taxable income over $22,100 |
| 3 | More than $53,500 but $115,000 or less | $12,107 plus 31% of the taxable income over $53,500 |
| 4 | More than $115,000 but $250,000 or less | $31,172 plus 36% of the taxable income over $115,000 |
| 5 | Over $250,000 | $79,772 plus 39.6% of the taxable income over $250,000 |

Create a FederalTax class with the following:

§ Declaration of an instance variable for the taxable income, a real number

§ Declaration of constants for all the tax bracket income levels, the tax paid base amounts for each bracket, and the percents for each bracket

§ Declaration of a constant NumberFormat to format all dollars

§ Constructors - default (zero for the taxable income) & non-default

§ Accessors - returns value of the instance variable

§ Mutators - assigns new value to the instance variable, verify that the taxable income is non-negative, or assign zero

§ a "public double taxPaid()" method that uses the above table to compute the tax

§ a "public String toString()" method that displays the value of the instance variable AND the taxPaid.

Code then test (complete and check against Expected Result below) your methods by creating application (client) class FederalTaxApp.java to test your FederalTax class.

Complete the test plan below.

Implement your pseudocode in java. Be sure your program is appropriately documented. Accept user input from the keyboard.

Compile and run your program to see if it runs (no run-time errors).

Test your program with the test plan below. If you discover mistakes in your program, correct them and execute the test plan again.

|  |  |  |  |
| --- | --- | --- | --- |
| Test plan | | | |
| Test case | Sample input data | Expected result | Verified |
| Tax Bracket 1 | 15,000 | 12,750 | Yes |
| Tax Bracket 2 | 25,000 | 20,873 | Yes |
| Tax Bracket 3 | 65,000 | 49,328 | Yes |
| Tax Bracket 4 | 150,000 | 106,228 | Yes |
| Tax Bracket 5 | 400,000 | 260,128 | Yes |

Here is some sample output for a few of the test cases above, you must test them all.

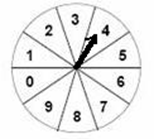
|  |
| --- |
| **Sample output:** |
| Default FederalTax Object  Taxable Income: $0.00 Tax Paid: $0.00    Enter your taxable income: 52000  Updated FederalTax Object  Taxable Income: $52,000.00 Tax Paid: $11,687.00    Enter another taxable income: 137000  Non-Default FederalTax Object  Taxable Income: $137,000.00 Tax Paid: $39,092.00    Test a negative Income  Taxable Income: $0.00 Tax Paid: $0.00 |

**Problem 2 [12 points]:**

Design a class for an everyday object, including required data and methods.

Define the private attributes (**and their data types and valid ranges**) and public methods (**and their arguments and return types**) for the following classes.

**You do not need to code, just design**.



Populate provided tables (enter as many rows as you find necessary; add more if needed) with your answers. Feel free to add extra tables, boxes, comments, etc. if needed.

1. A fair 10-section game spinner (as shown below) **[2 points]**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Private attributes (use “N/A”, “undefined”, “none”, etc. If necessary)** | | | |
| Attribute name | Data type | Valid range of values | Comments |
| set1 | int | 1-2 | This attribute includes values between tht range of 1 and 2, and has to be from 0 to 0.2 in random method |
| set2 | int | 3-4 | This attribute includes values between the range of 3-4, and has to be from 0.2 to 0.4 in random method |
| Set3 | int | 5-6 | This attribute includes values between the range of 5-6, and has to be from 0.4 to 0.6 in random method |
| set4 | int | 7-8 | This attribute includes values between the range of 7-8, and has to be from 0.6 to 0.8 in random method |
| Set5 | Int | 9-10 | This attribute includes values between the range of 9-10, and has to be from 0.8 to 1.0 in random method |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Public methods (use “N/A”, “undefined”, “none”, etc. If necessary)** | | | |
| Method name | Return data type | Arguments | Comments |
| Public Math.random() | int | The value has to be between 0 and 1 | If it is between 0 and 0.2 set1 will be chosen |
| Public Math.random() | int | The value has to be between 0 and 1 | If it is between 0.2 and 0.4 set2 will be chosen |
| Public Math.random() | int | The value has to be between 0 and 1 | If it is between 0.4 and 0.6 set3 will be chosen |
| Public Math.random() | int | The value has to between 0 and 1 | If it is between 0.6 and 0.8 set4 will be chosen |
| Public Math.random() | int | The value has to be between 0 and 1 | If it is between 0.8 and 1 set5 will be chosen |
|  |  |  |  |

2. A fraction data type **[4 points]**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Private attributes (use “N/A”, “undefined”, “none”, etc. If necessary)** | | | |
| Attribute name | Data type | Valid range of values | Comments |
| Private Fraction (set1/set2) | int | 1-4 | This attribute will set a fraction for set1 and set2 |
| Private Fraction (set2/set3) | int | 4-6 | This attribute will set a fraction for set2 and set2 |
| Private Fraction (set3/set4) | int | 6-8 | This attribute will set a fraction for set3 and set4 |
| Private Fraction (set4/set5) | Int | 8-10 | This attribute will set a fraction for set4 and set5 |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Public methods (use “N/A”, “undefined”, “none”, etc. If necessary)** | | | |
| Method name | Return data type | Arguments | Comments |
| Public Fraction (set1/set2) | int | The values can only be from set1 and set2 |  |
| Public Fraction (set2/set3) | Int | The values can only be from set2 and set3 |  |
| Public Fraction (set3/set4) | int | The values can only be from set3 and set4 |  |
| Public Fraction (set4/set5) | int | The values can only ne from set4 and set5 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

3. A soda vending machine **[6 points]**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Private attributes (use “N/A”, “undefined”, “none”, etc. If necessary)** | | | |
| Attribute name | Data type | Valid range of values | Comments |
| Private cash\_slot | int | 0 to 20 | The machine can take quarters as coins and paper bills |
| Private drink\_choice | float | A-f and 1-6 | The machine relies on two inputs one from alphabets and the other from numerical value. |
| Private change | int | 0 - 20 | The machine will give change if more amount is provided than necessary. |
| Private electricity | int | 0 to infinity | The electricity cost to run the machine is based on the total usage of the machine |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Public methods (use “N/A”, “undefined”, “none”, etc. If necessary)** | | | |
| Method name | Return data type | Arguments | Comments |
| Public cash\_slot | int | Cash\_slot >0 | The amount has to be greater than 0 |
| Public drink\_choice | int | Drink\_choice has to be an alphabelt and a numerical value |  |
| Public change | int | >= 0 | The value cannot be less than zero |
| Public electricity | Int | >= 0 | The value cannot be less than zero |
|  |  |  |  |
|  |  |  |  |