CSC262 – Week2 Assignment

Due on: Tuesday April 23, 2019

Chapter 6: (20 points)

Q1: Answer exercises 6.17 and 6.22 of the text book:

6.17 *(Even or Odd)* Write a method `isEven` that uses the remainder operator (%) to determine whether an integer is even. The method should take an integer argument and return `true` if the integer is even and `false` otherwise. Incorporate this method into an application that inputs a sequence of integers (one at a time) and determines whether each is even or odd.

6.22 *(Temperature Conversions)* Implement the following integer methods:

a) Method `celsius` returns the Celsius equivalent of a Fahrenheit temperature, using the calculation

```
celsius = 5.0 / 9.0 * (fahrenheit - 32);
```

b) Method `fahrenheit` returns the Fahrenheit equivalent of a Celsius temperature, using the calculation

```
fahrenheit = 9.0 / 5.0 * celsius + 32;
```

c) Use the methods from parts (a) and (b) to write an application that enables the user either to enter a Fahrenheit temperature and display the Celsius equivalent or to enter a Celsius temperature and display the Fahrenheit equivalent.

Chapter 7: (30 points)

Q2: Answer exercises 7.11 and 7.17 of the text book:

7.11 Write statements that perform the following one-dimensional-array operations:

a) Set the 10 elements of integer array `counts` to zero.

b) Add one to each of the 15 elements of integer array `bonus`.

c) Display the five values of integer array `bestScores` in column format.

7.17 *(Dice Rolling)* Write an application to simulate the rolling of two dice. The application should use an object of class `Random` once to roll the first die and again to roll the second die. The sum of the two values should then be calculated. Each die can show an integer value from 1 to 6, so the sum of the values will vary from 2 to 12, with 7 being the most frequent sum, and 2 and 12 the least frequent. Figure 7.28 shows the 36 possible combinations of the two dice. Your application should roll the dice 36,000,000 times. Use a one-dimensional array to tally the number of times each possible sum appears. Display the results in tabular format.

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Fig. 7.28 | The 36 possible sums of two dice.

Q3: Write an application that performs the following:

1- Creates an ArrayList, called **al** that can store integers.
2- Fills the **al** ArrayList with 10 random integer numbers.
3- Prints the content of **al**.
4- Prints the number of elements in **al.**
5- Removes the first element of **al**.
6- Prints the number of elements in **al.**
7- Prints the content of **al**.

Chapter 8: (30 points)

Q4: Answer exercise 8.4 of the text book.

8.4 *(Rectangle Class)* Create a class `Rectangle` with attributes `length` and `width`, each of which defaults to 1. Provide methods that calculate the rectangle's perimeter and area. It has *set* and *get* methods for both `length` and `width`. The *set* methods should verify that `length` and `width` are each floating-point numbers larger than 0.0 and less than 20.0. Write a program to test class `Rectangle`.

Q5: A bicycle has two wheels, several gears and a body. For this question, create and use a bicycle class as follows:

1- Create a class called **Wheel.** This class has two private instance variables called **width** and **radius,** both of type **double**.

For the **Wheel** class do the following:

a) Write two constructors for the **Wheel** class as follows:

- A default constructor that sets the class instance variables to zero
- A constructor that receives two double values as parameters and uses them to set the values of the instance variables.

b) Write the set and get methods for the **Wheel** class to set and return the values of the instance variables, respectively.

2- Create a class called **Bicycle** with the following private instance variables:

1: wheelOne of type Wheel,
2: wheelTwo of type Wheel,
3: bodyColor of type String,
4: numberOfGears of type integer

For the **Bicycle** class do the following:

a) Write a constructor that receives values for the instance variables as parameters and sets them. **Note:** ***Do not define a default constructor for the Bicycle class.***

b) Write the set and get methods for the **Bicycle** class to set and return the values of its instance variables. Please note that:

- The set methods that assign values to the wheelOne and wheelTwo, should receive **Wheel** objects as parameters.
- The wheelOne and wheelTwo get methods should return **Wheel** objects.

3- Test the above classes by writing a class called **BicycleTest.java** that does the following:

a) Creates an instance of **Bicycle** as follows:

- wheelOne: width = 1.0, radius = 27.0 (This **Wheel** object needs to be created)
- wheelTwo: width = 1.5, radius = 28.0 (This **Wheel** object needs to be created)
- bodyColor: "Blue"
- numberOfGears: 18

b) Prints the values of the **Bicycle** object and its **Wheel** objects instance variables.

c) Changes the numberOfGears to 21.

d) Prints the values of the **Bicycle** object and its **Wheel** objects instance variables again.

e) Compares the widths and radiuses of wheelOne and wheelTwo. If both are equal, the word "Equal" is printed otherwise the word "Not Equal" is printed.