420-202-RE Assignment 1-A Due Date: 31 January 2017

Objectives

1. Refresh your memory on Java programming (after a long and well-deserved holiday!)
2. Recall that we use Java classes to model *things* with common attributes and behavior.
3. Recall that the attributes of the *things* we model are implemented as *fields* within the class, with some fields representing *instance fields* and the rest *static fields*.
4. Recall that the behavior of the *things* we model are implemented as *methods* within the class, with some methods representing *instance methods* and the rest *static methods*.
5. Give you continuous practice writing and using classes.
6. Practice writing and using both normal and default constructors.
7. Become fluent with wringing and using getter and setter methods.
8. Start and get in the habit of always overriding the **toString** method in order to provide its invoking object a string representation.
9. Start and get in the habit of always overriding the **equals** method to compare for equality the object that invoked it and the object it supplies through its parameter.
10. Recall that the usage of primitive variables is fundamentally different from that of reference variables in Java.

Modeling Fruits

In this assignment you are required to implement a Java class, named **Fruit**, to model fruits in terms of five common key attributes: name, portion size, protein (in grams), calories, and dietary fiber (in grams).[1]

For example, here are instance diagrams for two sample **Fruit** objects **f1** and **f2**.

f1:Fruit	
name =	Apple
portion =	One medium apple with skin
protein =	0.47
calories =	95
fiber =	4.4

f2:Fruit	
name =	Orange
portion =	One medium orange
protein =	1.23
calories =	62
fiber =	3.1

Each line in a instance diagram shows an attribute and its value.

[1] Note that fruits and vegetables also contain minerals (such as Potassium, Phosphorus, Calcium, Magnesium, Iron, Sodium, Zinc, Manganese, Copper, Selenium, and a small amount of other minerals) and vitamins (such as Vitamin B1 (thiamine), Vitamin B2 (riboflavin), Niacin, Pantothenic Acid, Vitamin B6, Folate, Vitamin B12, Vitamin E, Vitamin K, Vitamin D, and a small amount of other vitamins). For the sake of simplicity, we are not interested in the mineral and vitamin related attributes of fruits in this assignment.

Public Interface of Class Fruit

The public interface of your **Fruit** class should include the following methods:

- **A constructor** Takes 2 parameters of type **String** that supply initial values for fruit name and portion size, and 3 parameters of type **double** that supply initial values for protein, calories, and fiber. Initializes the instance variables with the supplied values.
- **Setter and Getter Methods**
  One pair for each instance variable; that is, five setter and five getter methods.
- **An appropriate equals method** that returns a **boolean** and takes a **Fruit** as a parameter. Here we consider two **Fruit** objects to be equal if they have the same values for calories, fiber and protein. (Different applications may use different definition for equalty of two objects.)
- **A toString method** Returns a string representation of the invoking object. For example, `f2.toString()` should return the following string:

```
Fruit name   : Orange
Portion size: One medium orange
Protein      : 1.23 grams
calories     : 62
fiber        : 3.1 grams
```

  To practice formatting output, the values for calories, fiber and protein should be formatted with zero, one and two decimal places, respectively.

Test drive your Fruit class

Write a driver program to test your **Fruit** class. A driver program is a class with a method **main**. Perform the following actions in the **main** method:

1. Create a **Fruit** object named **f1** initializing it with the values shown on page 1.
2. Print **f1**.
3. Create a **Fruit** object named **f2** initializing it with the values shown on page 1.
4. Print **f2**.
5. Compare **f1** and **f2** for equality and display the result using an appropriate message.
6. Swap the calorie values of **f1** and **f2** using the getter and setter methods.
7. Print **f1**.
8. Print **f2**.

Evaluation Criteria

Correctness of execution of your program	60%
Proper use of required Java concepts	20%
Java API documentation style	10%
Comments on nontrivial steps in code, Choice of meaningful variable names, Indentation and readability of program	10%