CISC 181 Spring 2014
Practice Set 4
Assigned: March 23
Due: April 6 at 11:55PM on Sakai (technically it should be due March 30, but Spring Break is that week so I am automatically extending it to the end of Spring Break. However, you should probably do the assignment this week before Spring Break so that you don't forget/run out of time).

Practice sets are to be completed individually. You are free to consult other students to help complete the practice sets (see syllabus for collaboration policy). However, keep in mind that each practice set is designed to cover basic material on which you will be quizzed and tested.

This practice set is intended to cover the following major topics:

- Creating object hierarchies
- Overriding methods for polymorphism
- Abstracting patterns of behavior using abstract classes and interfaces
- Implementing pre-defined patterns common for Objects (equals, toString, compareTo)
- Working with arrays and loops over arrays of primitive types
- Identifying test cases and writing your own tests

PART A [50 POINTS]: A CLASS HIERARCHY WITH ABSTRACTION AND INTERFACES

You will be writing a total of 8 classes/interfaces for this part (excluding the Restaurant and PS4PartATests class). Your code needs to pass the included tests in PS4PartATests without modifying PS4PartATests or Restaurant. Create appropriate data definitions (including constructor, properties, accessors -- no mutators) for each of these new types.

1) **[10 points]** All Fruit have properties for color and weight (in grams) and Orange is a Fruit. All Fruit have a way of computing how many calories they contain based on their weight, `double getCalories()`, but implementation must be left to the subclass.
2) **[4 points]** A Fruit is equal to another Fruit if they have the equal weight and color.
3) **[4 points]** Bananas are also Fruit but they should have a different color than Oranges.
4) **[4 points]** A NavalOrange is an Orange, but it is sweeter than a regular Orange. In fact, it has exactly 10% more calories than an Orange of the same weight.
5) **[6 points]** Bananas and Oranges are considered Edible. All Edible objects have a way of computing how many calories they contain.
6) **[4 points]** A Snowberry is a Fruit but is not Edible. It is a White, toxic berry that weighs 0.117 grams and contains 1 calorie.
7) **[4 points]** IceCream is Edible but is not a Fruit.
8) **[0 points -- it is already there]** The Restaurant is able to maintain a list of Edible items using a `void addItem(Edible item)` method.
9) **[10 points]** A CalorieComparator implements a Comparator for Edible objects that determines which Edible object has less calories.
   The CalorieComparator is used in the `orderByCalories()` method on Restaurant to implement a method that re-orders the Restaurant's list of Edible objects.

10) **[4 points]** The restaurant also has a method to produce a String for a menu that is ordered by calories. Given the following code, it prints the output:

```
Restaurant mcds = new Restaurant();
mcds.addEdibleItem(new IceCream());
mcds.addEdibleItem(new Orange(8));
mcds.addEdibleItem(new NavalOrange(4));
mcds.addEdibleItem(new Banana(6));
mcds.addEdibleItem(new NavalOrange(100));
System.out.println(mcds.getMenu());
```

Output:

```
NavalOrange
Orange
Banana
ReallyBigNavalOrange
IceCream
```

You should make sure the Restaurant getMenu tests work without changing the test or Restaurant (you need a correct CalorieComparator and toString methods in your classes to make the test pass).

PART B[50 POINTS]: WORKING WITH ARRAYS AND LOOPS OVER ARRAYS OF PRIMITIVE TYPES

Create a class named ArrayStaticMethods. This class will consist solely of *public static* methods useful for operating on arrays.

1. **[5 points]** Write the method:

   public static double mean(double[] data)

   where the returned double is the average of all values in data.

   - Throughout Part B you will be adding test cases to the existing tests in PS4PartBTests.java by following the instructions indicated in the comments. Add the test cases for test_mean.

2. **[10 points]** Write the method **countWords**. Given a sentence as an array of characters, compute how many words are included. The only characters that will appear in our sentences are letters, spaces, commas, and a period will always end the sentence. Every comma will be followed by a space.
   - Add the test cases for test_countWords

3. **[10 points]** Write the method, **replace**, that takes an integer array and two integer values as parameters. The method **produces a new array** with exactly the same contents as the parameter, except that all occurrences of value1 have been replaced

with value2.

```
public static int[] replace(int[] values, int value1, int value2)
```

- Add the test cases for test_replace

4. **[15 points]** Write the method, **evenFront,** that takes an array of integers as parameter. It returns a new array (do not modify the original data) that contains the exact same numbers as the given array, but arranged so that all the even numbers are grouped at the start of the array. The order of the numbers within the even/odd groupings must be the same as the original array.

   - Add the test cases for test_evenFront

5. **[10 points]** Write the method, **surroundedCharacter**, that takes an array of characters as a parameter and returns true if any character is surrounded by a different character. For example, in 'abcdc' the character d would be surrounded by c, but in 'abccc' no characters are surrounded by a different character.

   - Add the test cases for test_surroundedCharacter

Submit your PS4 project to Sakai by exporting it from Eclipse as an archive.