CS5004-5005-SV – Lab 5 – June 4, 2020
Practice with Static Members and Such

The main goal of today's lab is to practice using static methods and static variables, while continuing to practice creating classes and methods.

0. Make a ***copy*** of the DateText.java code from Lecture 5, in a new folder. Verify that it still works as expected in your IDE (whether VSCode or IntelliJ). (Sometimes you may need to tweak path for imports, etc.) Keep the original version intact for possible use in future lecture activities.

1. Modify your copy as follows:

   a. Refine the dateCheck to consider how many days are in a month and whether or not it is a Leap Year (29 days in February?). Add tests to the main method to verify. (Hint: https://www.geeksforgeeks.org/program-check-given-year-leap-year/))

   b. Complete the monthString method to work for all 12 months. Add tests to the main method to verify correct operation.

2. Consider the Person class, which has private member variables *name* (a String), *born* (a Date), and *died* (a Date). All three types are *objects*, not primitive types, and so the return value is a *reference*. So, why is it OK to return the *name,* without making a copy of it, but in the case of *born* and *died*, it is important for the accessor method to return a *copy*? Draw a diagram similar to slide 25 of the lecture. Create a sample Java code file using Person where the accessor *not* returning a *copy* might lead to a bug. Your response should include a photo (*eg* smartphone) or screenshot of your diagram (as a .png, .jpg or similar file type), your sample code, with a sentence or two of commentary in your sample code file, explaining the issue in the form of comments.

3. Create a new class, Country, with a String instance variable *name*, an int instance variable *population*, and an int instance variable *seniority*. The seniority of the oldest existing country should be 0. The next oldest country should have seniority of 1, etc. To keep track of things, you will need to provide a *static* int variable called NumberOfCountries. It should be private, but have a public accessor method. There should *not* be a mutator method for this member. The constructor(s) for countries should increment this counter. Create test code in a main method to demonstrate correct behavior.

4. Create an initial version of a class, Fraction. To construct a fraction, you will need an integer numerator and an integer denominator. The numerator can be any legal int, but the denominator must be a positive int. Hence, if a fraction is negative, the numerator must be negative.

   a. For now, your constructor should check whether the denominator is legal and, if not, print an error message and call System.exit(-1). You should provide accessors ("getters") for the numerator and denominator. Also create a public Boolean method, *isEqualTo*, which accepts another fraction as input and returns true if they represent the same value. (For example, 1/2 is equal to 2/4, and 3/2 is equal to 9/6.) Hint: cross-multiply.

   b. Also add a method, toDouble, that returns a double that is approximately equal to the value of the fraction, but expressed in decimal notation. (For example, toDouble(1/2) is (approximately) equal to 0.5.) Create a main method with several examples to test your code so far. Turn in this file, but first rename it to *FractionV0.java*, so you can now edit a copy.

   c. Make a copy of the initial version, in a new file, as a new project in your IDE. Edit your copy to add setters for the numerator and denominator. Note that the setter for the denominator should ensure that it is a positive integer, as was required in the above constructor. Hence, you probably now have duplicate/redundant code to handle this error checking. There are two ways to avoid this problem. Describe one in a comment in your file, and then implement a *different* one. (Either way is fine.) Modify the main method to add tests for your new setters as well. Also turn in this file, but you can go ahead and call this *improved* version *Fraction.java*.

5. Create a java class TestFinal using the *final* keyword on an instance variable. Try creating and testing a mutator ("setter") method for it. Also trying setting Math.PI. Turn in TestFinal.java with comments about what happens.

Submit your answers to above items to Canvas for CS5005 *(due next Thursday by midnight). Only your .java files and one picture file (eg .png) for the above need to be uploaded. Prompt submission encouraged, but late submissions are still better than not submitting.*

6. Work on any unfinished labs or CS5004 homework assignments. If finished, ***submit homeworks to Canvas CS5004*** (normally due Mondays at 2 AM).