SWE 619 Assignments 12 and 13
Fall 2019

Goal: Applying lessons learned.
You have a choice of possible assignments. Choose one for assignment 12 and another (different one) for assignment 13.

In each case, the deliverable is a *story*. Write a brief report, and include enough evidence (output, screen shots, etc.) that the GTA can figure out that you actually completed the assignment.

1. Bring up the C# contracts mechanism in VisualStudio. Translate one of Liskov's examples to C# and implement the contracts with the Code Contracts mechanism.
2. Consider one of the `copyOf()` methods in the java Arrays utility class. Bloch uses this method in his `Stack` example. Code a corresponding method in C++, changing the argument list as necessary. Provide a specification for the C++ code by translating the JavaDoc and adding preconditions as necessary. Explain what this exercise demonstrates about C++ type safety.
3. For most of the semester, we have focused on design considerations for constructing software that does something we want it to do. For this last assignment, I would like students to appreciate just how vulnerable software is to malicious parties intent on attacking their software. Students who find this assignment amusing might wish to take CS 468: Secure Programming and Systems.

   There are two attacks documented in Bloch's Item 88: *Write `readObject()` methods defensively*. One is called `BogusPeriod`, and the other is called `MutablePeriod`. Implement either (your choice) of these attacks (basically involves typing in code from Bloch) and verify that the attack takes place.

4. Find some existing (Java) code that uses the "int enum pattern" and refactor it to use Java Enums instead. Identify any type-safety issue you uncover in the existing code. To make the exercise interesting, extend your enums beyond simple named-constants in one of the ways discussed by Bloch in Item 34.
5. Do *one* of the following, depending on your current Agile expertise:
   - *TDD Novice*: Work through a simple TDD example one step at a time. Don't skip anything. Koskela, Chapter 2, has an excellent example. Koskela is available online through the GMU's subscription to the Safari service:
     Direct Safari Link
     Off Campus Safari Link.
   - *Continuous Integration Novice*: Download Cruise Control (somewhat dated), Jenkins, Travis-CI, or Gradle (or some combination). Use these tools to set up some toy Java project for continuous integration. For realism, you should set up version control as well. One possibility is Subversion. Another, now more popular, one is GitHub.
   - *Regular JUnit User*: Download a mocking tool such as EasyMock and set up some interaction-based tests. Koskela has examples.
   - *Agile Developer*: Download and explore some Agile tool with which you are currently unfamiliar.

6. Convert an existing JUnit test set to JUnit `Theories`. If you wish to construct JUnit from scratch for this, that's fine too.