SWE 619 Fall 2020
Final Paul Ammann

This is an open-book/open-notes exam. This means that you can access course materials on paper or on the internet. The time limit on this exam is **2 hours and 30 minutes**.

It is a violation of the honor code to communicate with any other person (except me, the instructor) about this exam while you are taking it.

It is a violation of the honor code to discuss or share the contents of this exam in any way with any student who is currently registered for this course but who has not yet completed this exam.

When you see a reference to code, you should map that to the relevant Java file I provided as part of your study guide for this exam.

Capture your answers as a PDF document and submit on Blackboard by 7:00PM. If, for any reason, you have a problem submitting to BB, submit your final on Piazza in a private post. Your post should also explain your problem.

Please make sure your **NAME** is at the top of your submission.

Code for the examples used in this exam is published on guide discussed in class. See the course web page for the relevant link. Make sure you are getting the right code!

Section	Points	Score
Question 1	30	
Question 2	35	
Question 3	35	
Total	100	

Question 1

Consider `MapPoly.java`.

1. Write a reasonable rep-invariant for `MapPoly`.
2. As written, the contract for the `coeff()` method is inconsistent with other contracts in the class.
   - What is the inconsistency with the contract?
   - Fix the inconsistency with the contract.

- Fix the code to match the revised contract.

3. Argue that the implementation of the `coeff()` method is correct (with respect to your repaired contract, of course.)

Question 2

Consider Bloch's final version of his `Chooser` example, namely `GenericChooser.java`. For this exercise, you may assume that the generic parameter `T` represents an immutable type such as `String`.

1. Provide an implementation of `equals()` appropriate for this class. Think carefully about this; why is this more complex than it might first appear? Give an example to explain. Provide an appropriate implementation of `hashCode()`.

2. Provide a contract and an implementation for another method,

   ```
   public void addChoice(T t)
   ```

   This method should do the obvious thing based on the name.

3. Suppose we decide to make this class immutable. Would that be reasonable? What changes, if any, would be necessary to the code? Why would those changes be necessary? Be specific: explain how the class would exhibit mutable behavior without this change. Include the `addChoice` method from the prior question in your analysis.

Question 3

Consider `StackInClass.java`. You should take special note of the `push()` method; it is a variation on Bloch's code.

1. What is wrong with `toString()`? Fix it.

2. As written, `pushAll()` requires documentation that violates encapsulation. Explain why and then write a contract for `pushAll()`.

3. Rewrite the `pop()` method for an immutable version of the `Stack` class. Keep the same instance variables.

4. Implementing the `equals()` method for this class is a messy exercise, but would be much easier if the array was replaced by a list. Explain why. Note: You are not required to provide a implementation in your answer, but if you find it helpful to do so, that's fine.