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 Bloch's final version of his `Chooser` example, namely `GenericChooser.java`.

1. What would be a good good representation invariant for this class? You may want to argue that the client view of `Chooser` objects should be changed to support your proposed invariant. If so, explain exactly what you are doing and why.
2. Supply suitable contracts for the constructor and the `choose()` method and recode if necessary. The contracts should be consistent with your answer to the previous question. Explain exactly what you are doing and why.

3. Argue that the `choose()` method, as documented and possibly updated in your previous answers, is correct. You don't have to be especially formal, but you do have to ask (and answer) the right questions.

Question 2

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.

Question 3

Consider `Queue.java`.

1. Write a reasonable `toString()` implementation.
2. Consider a new method, `deQueueAll()`, which does exactly what the name suggests. Write a reasonable contract for this method and then implement it. Be sure to follow Bloch's advice with respect to generics.
3. Rewrite the `deQueue()` method for an immutable version of this class.
4. Write a reasonable implementation of `clone()`. Note: Don't worry about Java syntax; "exam" Java is fine.