SWE 619–Spring'24: Final Exam

Your Name:

5/1/2024

Read carefully these instructions

1. This is an open-book exam. This means that you can access course materials in the book/lecture notes/videos.
2. It is a violation of the honor code to communicate, discuss, or share the contents of this exam with any other person (except the instructor or TA) about this exam. You are also not allowed to use AI-based services (e.g., ChatGPT, etc.) to get help on this exam. If in doubt, ask the instructor.
3. You must type all solutions. You can use plain text format or markdown. If you use something else such as Word or LaTeX, you need to export to PDF and submit the PDF. **Do Not** submit any code (.java) file. If you need to change the code, put the modified code directly in your submission.
4. You need to submit on Blackboard by the deadline. 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.
5. In the exam, whenever there is a mentioning of a java file (e.g., Queue.java), then you can click on it to access the file.
6. The last two questions of the exam are optional and do not affect the grade of the final. They help me determine grade for group functioning and to help me improve the class. If you have done them last Friday, you do not need to do them again. Otherwise, I would appreciate your feedback.

Section	Points	Score
Question 1	25	
Question 2	20	
Question 3	20	
Question 4	35	
Question 5	0	
Question 6	0	
Total	100	

1 Question 1

Consider Queue.java (click on the link to open the file).

1. For `enQueue`, write (i) a partial contract and (ii) a total contract. For each part, if you need to change the code for the contract, do so and explain what you did.
2. Write the rep invs for this class. Explain what they are.
3. Write a reasonable `toString()` implementation. Explain what you did
4. 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. Explain what you did
5. Rewrite the `deQueue()` method for an immutable version of this class. Explain what you did
6. Explain the differences between rep invariants, loop invariants, and contract/specifications. Use the answers you provided above (or create your own examples) to demonstrate the differences.

2 Question 2

Consider this Chooser code GenericChooser.java.

1. What would be good rep invariants for this class? Explain each.
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.

3 Question 3

Consider StackInClass.java.

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. Explain what you did.
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.

4 Question 4

Consider the program below (y is the input).

```
1 {y ≥ 1} // precondition
2
3 x := 0;
4 while(x < y)
5   x += 2;
6
7 {x ≥ y} // post condition
```

1. Informally argue that this program satisfies the given specification (pre/post conditions).
2. Provide the steps you would need to do in Hoare logic to verify the pre and post conditions (e.g., starting from the postcondition, what do you do to get to the top and create the verification condition). You do not need to actually do these steps, instead just summarize and give them.
3. Give 3 loop invariants for the while loop in this program. For each loop invariant, informally argue why it is a loop invariant. Do not give the trivial loop invariants `True` and equivalent ones (e.g., if you say $x = y$ is an invariant, then do not give $y = x$ as another one).
4. *Sufficiently strong loop invariants:* Use a sufficiently strong loop invariant to formally prove that the program is correct with respect to given specification. This loop invariant can be one of those you computed in the previous question or something new.
   - Note: show all works for this step (e.g., obtain weakest preconditions, verification condition, and analyze the verification condition).
   - Recall that if the loop invariant is strong enough, then you will be able to do the proof. In contrast, if it is not strong enough, then you cannot do the proof.
5. Instead of **proving** the given specifications like above, now you just want to **test** the program.
   - Provide some short JUnit Theories code to test this program. You do not need to give compilable code. You can reuse/adapt code from the lecture notes. But be sure to explain what you are doing.
   - Gives the pros/cons of using (i) JUnit Theories, (ii) JUnit tests, and (iii) the Hoare logic proof style above.

5 Question 5

This question helps me determine the grade for group functioning. It does not affect the grade of this final.

1. Who are your group members?
2. For each group member, rate their participation in the group on the following scale:
   (a) Completely absent
   (b) Occasionally attended, but didn't contribute reliably
   (c) Regular participant; contributed reliably
3. If you believe you did not contribute to your group, you can try to explain why here.

6 Question 6

There is no right or wrong answer for the below questions, but they can help me improve the class. I might present your text verbatim (but anonymously) to next year's students when they are considering taking the course (e.g., in the first week of class) and also add your advice to the project description pages.

1. What were your favorite things about the class? What were your least favorite things?
2. Favorite things the professor did or didn't do?
3. What would you change for next time?