Module 3 - Exceptions

# Overview and Objectives

## TODO OVERVIEW

## COURSE LEVEL OBJECTIVES (CLO)

Upon completion of this course, you should be able to:

1. Construct modern high quality software systems and reason about them.
2. Properly define software specifications and rep-invariants.
3. Leverage immutability to properly construct threat safe programs.
4. Explain object-oriented concepts such as information hiding, encapsulation, data and type abstraction, and polymorphism.
5. Properly use exception handling
6. Identify when it is appropriate to use inheritance and generics.

## TODO MODULE LEVEL OBJECTIVES (MLO)

Upon completion of this module’s activities, you should be able to:

# Module Video (Wiley-Produced w/Dan Ramos) [3-5 minutes]

# Learning Materials [~100 pages, ~3.5 hours]

## TEXTBOOK READINGS

* Barbara Liskov with John Guttag. Program Development in Java. Addison Wesley, 2001, ISBN 0-201-65768-6.
  + Chapter 4: Exceptions

# TODO Module 3 Learning Unit 1 – Benefits of Abstractions (MLO 1) [~0.5 hour]

# TODO Module 3 Learning Unit 2 – Program Specifications and Abstractions (MLO 1, 2) [~2.5 hour]

## Instructor Screencast: TITLE

# TODO Module 3 Learning Unit 2 – Program Specifications and Abstractions (MLO 1, 2) [~2.5 hour]

# Exercise (MLO 1, 2, 3) [.5 hours]

# Exercise (MLO 1, 2, 3) [.5 hours]

# Module 3 Assignment – (MLO 1, 2) [~2 hours]

## Purpose

## Instructions

## Deliverable

* Submit a .java file for your implementation.

## Due Date

Your assignment is due by Sunday 11:59 PM, ET.

# TODO Module 1 Quiz (MLO 1, 2) [~.5 hour]

## Purpose

Quizzes in this course give you an opportunity to demonstrate your knowledge of the subject material.

## Instructions

The quiz is 30 minutes in length. The quiz is closed-book.

## Deliverable

Use the link above to take the quiz.

## Due Date

Your quiz submission is due by Sunday 11:59 PM, ET.

= Final =

## 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 with any other person (except the instructor or TA) about this exam.
3. 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.
4. 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.
5. 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.

|  |  |  |
| --- | --- | --- |
| Section | Points | Score |
| Question 1 | 20 |  |
| Question 2 | 20 |  |
| Question 3 | 20 |  |
| Question 4 | 20 |  |
| Question 5 | 20 |  |
| Question 6 | 0 |  |
| Question 7 | 0 |  |
| Total | 100 |  |

## Question 1

Consider [Queue.java](https://nguyenthanhvuh.github.io/class-oo/files/Queue.java)

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. Write a reasonable implementation of clone(). Explain what you did.

## Question 2

Consider Bloch’s final version of his Chooser example, namely [GenericChooser.java](https://nguyenthanhvuh.github.io/class-oo/files/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

## Question 3

Consider [StackInClass.java](https://nguyenthanhvuh.github.io/class-oo/files/StackInClass.java). Note of the push() method 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. Rewrite 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.

## 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. Give 3 loop invariants for the while loop in this program. For each loop invariant, informally argue why it is a loop invariant.
3. 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.
4. Insufficiently strong loop invariants: Use another loop invariant (could be one of those you computed previously) and show that you cannot use it to prove the program.
   * Note: show all work as the previous question.

## Question 5

Note: you can reuse your answers/examples in previous questions to help you answer the following questions.

1. What does it mean that a program (or a method) is correct? Give (i) an example showing a program (or method) is correct, an (ii) an example showing a program (or method) is incorrect.
2. Explain the difference between rep invariants, loop invariants, and contract/specifications (i.e., pre/post conds). Use concrete examples to demonstrate the difference.
3. What are the benefits of using JUnit Theories comparing to standard JUnit tests. Use examples to demonstrate your understanding.
4. Explain the differences between proving and testing. In addition, if you cannot prove (e.g., using Hoare logic), then what does that mean about the program (e.g., is it wrong)?
5. Explain the Liskov Substitution Principle (LSP). Use a concrete example to demonstrate LSP. Note: use a different example than the one given in Liskov.

## Question 6

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

## Question 7

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 and least aspects of this class? Favorite topics?
2. Favorite things the professor did or didn’t do?
3. What would you change for next time?