CSCI 3901 Lab 5: Exceptions and Assertions

Fall 2021

Objective

In this lab, you will practice using exceptions and assertions.

Working alone or in a group of 2, you will create two small programs that demonstrate how to use exceptions and assertions.

Due: Friday, October 15, 2021 by 4pm in git at the repository https://git.cs.dal.ca/courses/2021-fall/csci3901/lab5/???.git where ???? is your csid.

Preparation

- Review the discussion on exceptions from class.
- Review the recursive code for calculating the Fibonacci numbers from class.
- Prepare pseudocode to implement a non-recursive binary search on an array of integers. You may get a copy from a source online for this exercise if you want.

Resources

- None required.

Procedure

Set-up

1. Get the recursive code for calculating the Fibonacci numbers working in a project.

Lab steps

Part 1 – Exceptions

1. Revise the Fibonacci code as follows to be able to limit the depth of the recursion:
    - Add two parameters to the method: one to say the maximum number of levels of recursion should be allowed and the second to say how deep you are now in the recursion
    - When the code calls itself recursively, increment the parameter on how deep you are into the recursion
2. Create a custom-made exception called “MaximumRecursionDepth” that extends the RuntimeException. The new exception should include a string message and an integer that is the depth of recursion reached. Ensure that your custom-made exception has two methods available:
    - getMessage() to return the string message
    - getDepth() to return the integer depth that was reached at the time of the exception.

3. Modify your Fibonacci code to now throw a MaximumRecursionDepth exception when the recursion depth exceeds the value of the parameter for the maximum number of levels of recursion.
4. Modify your Fibonacci code to stop further recursive calls the first time that a MaximumRecursionDepth exception happens.

Part 2 – Assertions

1. Write a non-recursive binary search code for an array. You can get help from the Internet for this part; just be sure to cite any sources that you use. This step is not the critical one of the lab.
2. Add assertions as a precondition, a loop invariant, and a postcondition to the main loop of your binary search.
3. Ensure that your assertions are working.

Questions

- We usually want you to re-use existing code and infrastructure whenever possible. Why might you create your own exception?
- We added parameters to the Fibonacci method. However, those parameters aren't very meaningful to a general user. What would you do to the code to make it more accessible for a general user?
- How would you recommend for someone to develop a loop invariant?
- How can loop invariants help you in programming, even if you don't include them directly as assertions in your code?

Reporting

1. In one file, list
    - The members of your team.
    - The answers from the Questions section of the lab.
2. Generate a PDF from the document.
3. Submit the PDF and all your Java files in Brightspace in the Lab/Lab 4 section of the course web page.

Assessment

The assessment will be on a letter grade and will reflect how well you implemented the exceptions and assertions, how much you re-used existing code, and how well you demonstrate that you can take the work of this lab and apply it to another situation (the questions section).