Lab 4: Exceptions and Assertions

Team Members:

1. Dhrumil Amish Shah

Date:

2021-02-04

—

Subject:

Software Development Concepts

—

Professor:

Matthew Amy

Q1.) We usually want you to re-use existing code and infrastructure whenever possible. Why might you create your own exception?

We create our own exception if the available exceptions do not cover our needs and are unable to provide meaningful information. For example, we created the MaximumRecursionDepth exception class in part 1 of this lab assignment so that we can add custom variables (message and depth) and custom methods (getMessage() and getDepth()) that returns message and depth. Now when recursion depth exceeds the maximum allowed depth, we can throw a more meaningful exception with an appropriate message. It leads to better code readability.

Q2 .) 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?

Create a separate method that only takes meaningful parameters from the user and enclose the original function call inside this method.

For example, if we only want to take a number from the user and return the calculated Fibonacci number. (i.e., n**(12)** -> fibonacci(12)**(144)**), we can do as below.
Note: maxDepthAllowed can be any constant value and 0 is the initial depth (starting depth).

```
public int fibo(int n){ calFibo(n, maxDepthAllowed, 0) }

private int fibonacci(int n, int maxDepthAllowed, int initalDepth) {
        // Write code here
}
```

And, if we want to take maxDepthAllowed from the user, we can do as below. 0 is the initial depth (starting depth).

```
public int fibonacci(int n, int maxDepthAllowed){ calFibo(n, maxDepthAllowed, 0) }

private int calFibo(int n, int maxDepthAllowed, int initalDepth) {
        // Write code here
}
```

Q3 .) How would you recommend for someone to develop a loop invariant?

First, think of conditions that must be true before the first execution of the loop. (For example, in the binary search problem we check if the array is sorted and start is less or equal to end).

Second, think of conditions that are true before the loop iteration. The same conditions should also be true after the iteration. (For example, in the binary search problem we check whether the start, mid and end are correct)

Last, think of termination condition that will tell something meaningful about the algorithm. (For example, in the binary search problem termination condition is when start > end)

Q4 .) How can loop invariants help you in programming, even if you don't include them directly as assertions in your code?

Loop invariants make sure that the developed algorithm is correct by giving useful properties. It helps us to understand the code and keeps us away from accidentally breaking any invariant in future. Also, it makes sure that loop terminates.