CSCI 3901

Software Development Concepts

Faculty of Computer Science

Lab 4: "EXCEPTIONS & ASSERTIONS"

Kishan Kahodariya B00864907

Dhruv Patel B00868931

Questions

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

- User defined exception are used to customize exception according to user need.
- Sometimes, we need to create exceptions around a specific business logic which can help the user or developer to understand the exact problem.
- Creating user defined exception helps to provide specific type of treatment to existing subset of exceptions class. In other words, we can pinpoint to exact problem in a code if that code throws exceptions from existing exception classes.
- For example, if a code throws general **IOException**, we can use custom exception to indicate the exact method or variable of code which is throwing this 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 general user?

- The parameter we added to the Fibonacci method are bit unusual for a user as he/she might only be interested in the total of Fibonacci sequence of **n** value, as provided by user.
- In terms of making this method more accessible, we can just use single parameter which accepts **n**, which denotes number for which the Fibonacci sequence is to be found.
- For finding the recursive depth, we can use a global variable that increments every time Fibonacci method is recursively called and that variable will be displayed as **Total Depth,** after successful recursive calls or even when exception is called.

How would you recommend for someone to develop loop invariant?

- Loop invariant is a user defined condition which holds true before and after every iteration of a loop depending on the need of the program.
- For an efficient loop invariant, it should basically satisfy 3 basic conditions, as follow:
    a. It should hold true before execution of the very first loop.
    b. If it holds true before the execution of the loop, then it should definitely hold true after the loop as well.
    c. Invariant should return something useful when that particular loop terminates that might help user or developer to understand flow of the program.

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

- As mentioned in the previous question, loop invariant can be created in such a way that at the end of loop execution, it can deliver useful information regarding working of that loop, which ultimately can help in understanding overall data flow.
- On the other hand, assertions are just kind of Boolean conditions which only checks if the loop satisfies that condition or not. If it does, then the loop executes otherwise code moves to next condition.
- Loop invariants stays true on both occasions i.e. before and after the loop is executed. Whereas assertions are conditions which are expected to be always true only at a certain place during the loop.