Week 7

Comment

The purpose of this prac is to practise using the IntelliJ Debugger. A debugger is a more sophisticated and less time-consuming way of finding and removing bugs in your code (when compared to strategies such as using print statements to debug). The debugging guide on Blackboard provides an introduction to debugging in IntelliJ, and will be useful for this prac.

1 Setting up

On Blackboard, you will find a `.zip` file containing 6 classes:

- `CachedFibonacci.java`
- `ChatBot.java`
- `RecursiveFibonacci.java`
- `CachedFibonacciTest.java`
- `ChatBotTest.java`
- `RecursiveFibonacciTest.java`

Create a new IntelliJ project, with a `src/` directory and a `test/` directory (ensuring that the latter is set up as a test source root). From inspecting the tops of the files, you should be able to see that they all need to be in a package to run. Create a directory `debugging/` under both your `src/` and `test/` directories, and copy the files into the appropriate folders. To make sure it worked, try running the tests. You should see 7 tests in total, and 3 of those tests should be failing.

There are approximately 350 lines of buggy code in today's prac. You will need to use the IntelliJ debugger in order to fix it.

1.1 Ground Rules

Since today's prac is specifically centred on learning the debugger, for today only, you are *banned* from modifying the code to add debugging print statements. In fact, you are *banned* from modifying the code altogether until you have found the bug and are ready to start fixing it. This ban applies to the tests as well.

2 Chat Bot

The first file you need to fix is `ChatBot.java`. This is the core of a chat bot that could be run on any chat platform, such as IRC or Facebook Messenger. The purpose of the bot is to let people play a simple guessing game to discover what its “magic number" is. As you can see from the test results, it doesn't work. Using the debugger, find out:

- where the code stops working
- why the code stops working
- how you can fix the code so that it does work

You will find reading the documentation present in the source files useful in working out how the code functions.

3 Recursive Fibonacci

This problem is trickier than the last one, and you may need to change your strategy a little. As you can see, the tests throw a "StackOverflow" exception.

To save you some time, a stack overflow exception is thrown when a function recursively calls itself so many times that the program runs out of stack memory. Think back to writing recursive methods at the start of semester - you need a base case so that your recursive function knows when to stop. Does this code have a base case? Is it being reached?

4 Cached Fibonacci - Challenge Task

Code is broken. Please fix.

Good luck.