Part 8: Unit testing Adder and Subtracter Classes

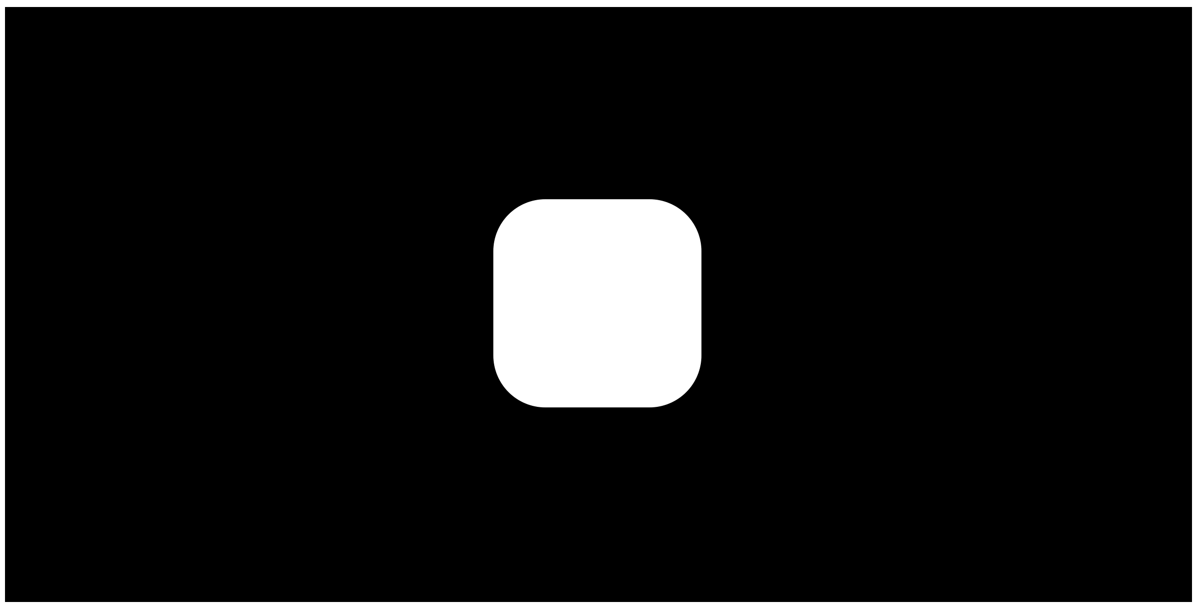
Bookmark this page

Part 8: Unit testing Adder and Subtracter Classes

Let’s begin our tests with unit testing the Adder and Subtracter classes. Recall that unit testing involves testing a single unit of code. This usually means testing a single class all by itself: without involving any other class.

If we can define the correct behavior of a single class and test it in isolation from any other code, then if our unit test fails we know it’s because of a failure in that specific class: if we’ve written a test that only uses Adder and not Subtracter, then we know that if that test fails it is because of a problem in the Adder class, something that is not related with its interaction with other classes (interaction between classes is covered by integration testing).

Unit Testing the Operators



Start of transcript. Skip to the end.

[MUSIC PLAYING]

Let's jump right in with Unit Test for adder and subtractor.

So we've created a [? test ?] [? op ?] [? high ?] file.

And in that, we're going to begin by importing all the things that we need.

Adder, subtractor, multiplier, and divider.

And then, for each set of tests that we

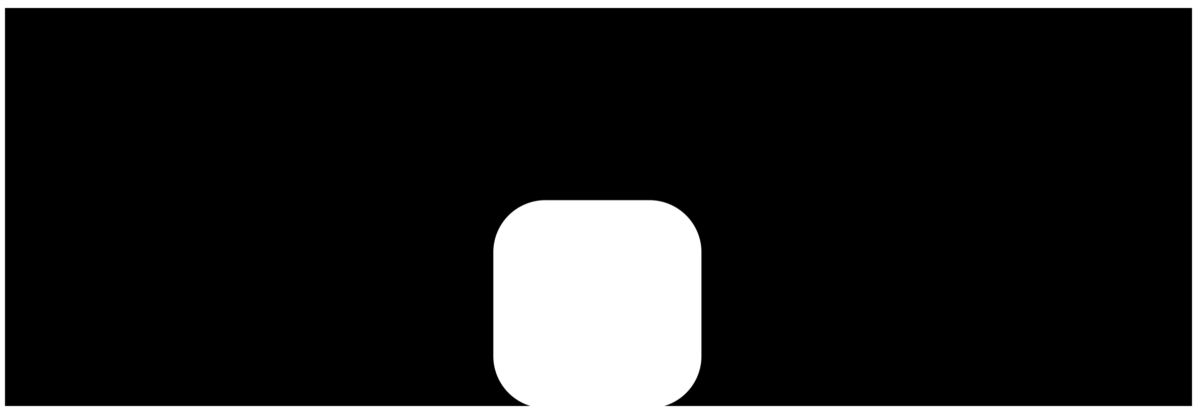
Video
Download video file

Transcripts
Download SubRip (.srt) file
Download Text (.txt) file

Unit testing is a good solution for the Adder and Subtracter classes: Adder and Subtracter don’t make use of any other classes. But just creating a Calculator instance requires creating an Adder, a Subtracter, a Multiplier, and a Divider: how can we possible devise unit tests for Calculator that don’t involve those other classes at all, so that the class can be kept isolated for testing purposes?

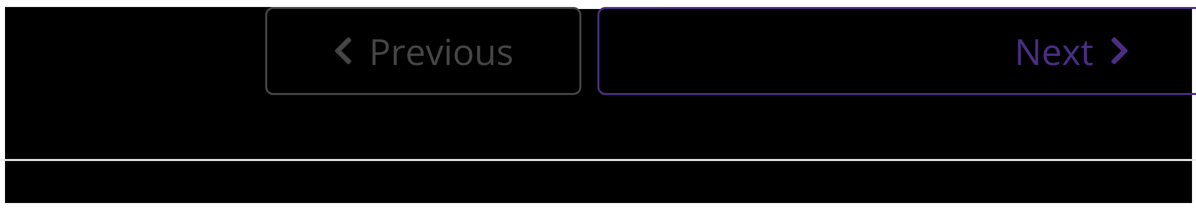
To answer this question, we introduce a new tool library: *mock*.

Unit Testing the Calculator



Start of transcript. Skip to the end.

[MUSIC PLAYING]



Previous Next

Now we can unit test our calculator.

We said that our calculator should raise an insufficient operands

exception if there are not enough operands in the stack

for it to operate on.

So let's test that, and we'll begin by