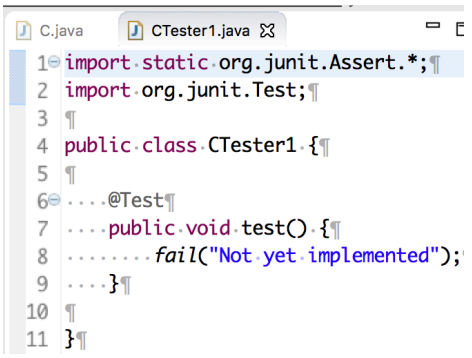
Mechanics of Junit Testing

We discuss the mechanics of JUnit testing and warn you of fatal mistakes you might make. We do this without really testing anything, allowing us to concentrate on the mechanics of testing.

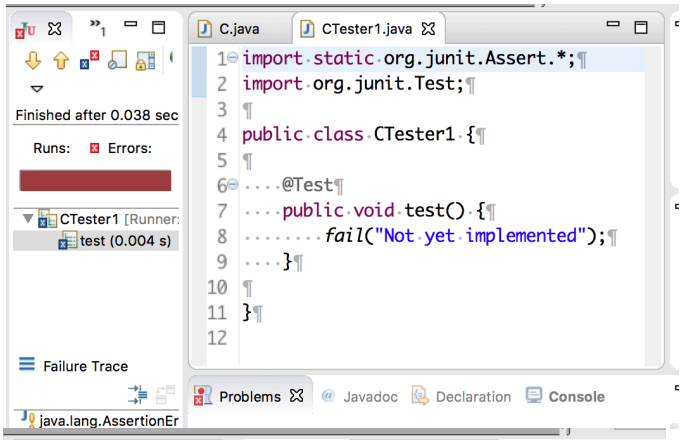
Here is the initial JUnit testing class, CTester1. It has one procedure, which is preceded by the annotation @Test. Procedure test contains a call on procedure fail, which always fails.



A test that fails.

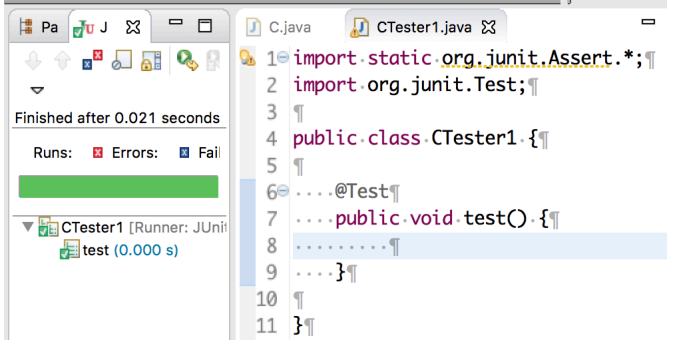
With class CTest1 selected in the Package Explorer pane, use menu item Run → Run.

This causes method test to be called, and the results are displayed in a new JUnit Testing pane where the Package Explorer pane was. A thick red bar appears, indicating that an error occurred. Calling procedure test caused an error because procedure fail *always* causes an error.



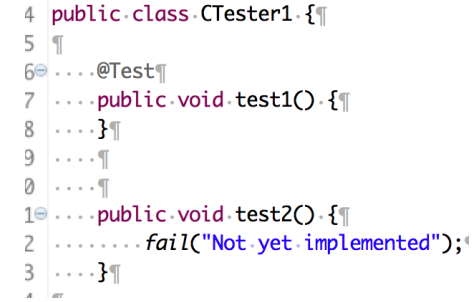
A test that passes.

Let's remove the call on procedure fail in method test and use menu item Run → Run again. Now, execution of a call on procedure test did not produce an error, so the JUnit pane has a green arrow.



A severe warning!

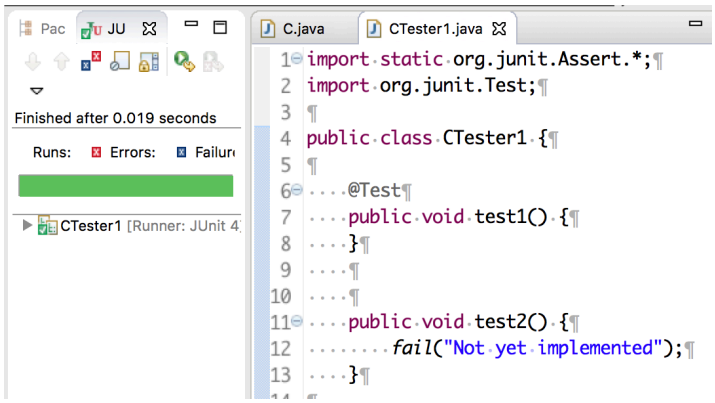
We now have two testing procedure in class CTest1, named test1 and test2. Procedure test1 has an empty procedure body; test2 has the call on procedure fail. Again, with JUnit testing class CTester1 selected in the Package Explorer pane, use menu item Run → Run.



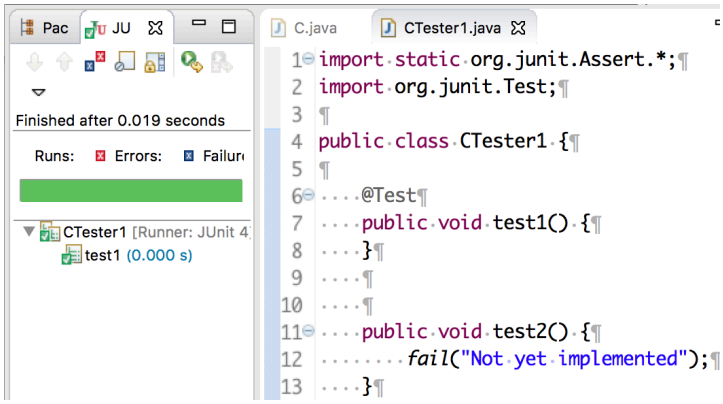
Mechanics of Junit Testing

Hey, the JUnit testing pane indicates that there were no errors! In spite of the fact that method procedure test2 called procedure fail. What happened?

Procedure test1 was executed because it was proceeded by the annotation @Test, but procedure test2 was *not* executed because it was *not* proceeded by annotation @Test. Only procedures that are preceded by that annotation will be executed.



We can see that procedure test2 was not executed in another way. Click the horizontal arrow that precedes CTester1 in the JUnit testing pane. The contents of that folder are revealed —only test1 and not test2 was executed. The green check just before test1 indicates that it ran without error.

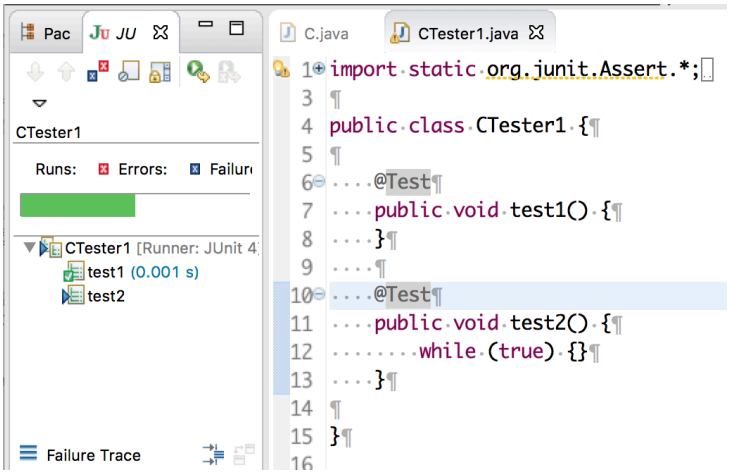


Always check!

Always check that all the procedures were executed. Click the horizontal arrow to reveal the contents of the folder and make sure that (1) all expected procedures are listed and (2) all have green check marks next to them.

Watch out for infinite loops!

We do one more thing. We put the @Test annotation before procedure test2 and change its body to have an infinite while-loop. Again, we run the program using Run → Run. There is a green bar, but it is only half as long as it should be. Secondly, while the contents of CTester1 indicates that procedure test1 completed, procedure test2 didn't. It has an arrow instead of a check mark, which means that the procedure is in an infinite loop.



Summary

Check that (1) all procedures were called and (2) their calls completed.