Testing

Prof. Michele Loreti

Programmazione Avanzata

Corso di Laurea in Informatica (L31)

Scuola di Scienze e Tecnologie

Testing...

What are software tests? A software test is a piece of software, which executes another piece of software, to check if that code results in the expected state (state testing) or executes the expected sequence of events (behavior testing).

Testing...

What are software tests? A software test is a piece of software, which executes another piece of software, to check if that code results in the expected state (state testing) or executes the expected sequence of events (behavior testing).

Why are software tests helpful? Software unit tests help the developer to verify that the logic of a piece of the program is correct.

Testing...

What are software tests? A software test is a piece of software, which executes another piece of software, to check if that code results in the expected state (state testing) or executes the expected sequence of events (behavior testing).

Why are software tests helpful? Software unit tests help the developer to verify that the logic of a piece of the program is correct.

Running tests automatically helps to identify software regressions introduced by changes in the source code.

Testing...

What are software tests? A software test is a piece of software, which executes another piece of software, to check if that code results in the expected state (state testing) or executes the expected sequence of events (behavior testing).

Why are software tests helpful? Software unit tests help the developer to verify that the logic of a piece of the program is correct.

Running tests automatically helps to identify software regressions introduced by changes in the source code.

Having a high test coverage of your code allows you to continue developing features without having to perform lots of manual tests.

Terminology

The code which is tested is typically called the code under test. If you are testing an application, this is called the application under test.

Terminology

The code which is tested is typically called the code under test. If you are testing an application, this is called the application under test.

A software test fixture sets up the system for the testing process by providing it with all the necessary code to initialise it, thereby satisfying whatever preconditions there may be.

Terminology

The code which is tested is typically called the code under test. If you are testing an application, this is called the application under test.

A software test fixture sets up the system for the testing process by providing it with all the necessary code to initialise it, thereby satisfying whatever preconditions there may be.

Example: load a database with known parameters from a customer site before running your test.

Testing levels

Unit Testing: Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level.

Testing levels

Unit Testing: Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level.

Integration testing: Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design.

Testing levels

Unit Testing: Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level.

Integration testing: Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design.

System testing: System testing tests a completely integrated system to verify that the system meets its requirements.

Unit test and unit testing

A unit test is a piece of code written by a developer that executes a specific functionality in the code to be tested and asserts a certain behaviour or state.

Unit test and unit testing

A unit test is a piece of code written by a developer that executes a specific functionality in the code to be tested and asserts a certain behaviour or state.

The percentage of code which is tested by unit tests is typically called test coverage.

Unit test and unit testing

A unit test is a piece of code written by a developer that executes a specific functionality in the code to be tested and asserts a certain behaviour or state.

The percentage of code which is tested by unit tests is typically called test coverage.

A unit test targets a small unit of code, e.g., a method or a class. External dependencies should be removed from unit tests, e.g., by replacing the dependency with a test implementation or a (mock) object created by a test framework.

Unit test and unit testing

A unit test is a piece of code written by a developer that executes a specific functionality in the code to be tested and asserts a certain behaviour or state.

The percentage of code which is tested by unit tests is typically called test coverage.

A unit test targets a small unit of code, e.g., a method or a class. External dependencies should be removed from unit tests, e.g., by replacing the dependency with a test implementation or a (mock) object created by a test framework.

Unit tests are not suitable for testing complex user interface or component interaction. For this, you should develop integration tests.

Which part of the software should be tested?

What should be tested is a highly controversial topic. Some developers believe every statement in your code should be tested.

Which part of the software should be tested?

What should be tested is a highly controversial topic. Some developers believe every statement in your code should be tested.

You should write software tests for the critical and complex parts of your application. If you introduce new features a solid test suite also protects you against regression in existing code.

Which part of the software should be tested?

What should be tested is a highly controversial topic. Some developers believe every statement in your code should be tested.

You should write software tests for the critical and complex parts of your application. If you introduce new features a solid test suite also protects you against regression in existing code.

In general it it safe to ignore trivial code. For example, it is typical useless to write tests for getter and setter methods which simply assign values to fields.

Which part of the software should be tested?

What should be tested is a highly controversial topic. Some developers believe every statement in your code should be tested.

You should write software tests for the critical and complex parts of your application. If you introduce new features a solid test suite also protects you against regression in existing code.

In general it it safe to ignore trivial code. For example, it is typical useless to write tests for getter and setter methods which simply assign values to fields.

If you start developing tests for an existing code base without any tests, it is good practice to start writing tests for code in which most of the errors happened in the past. This way you can focus on the critical parts of your application.

JUnit...

JUnit is a test framework which uses annotations to identify methods that specify a test. JUnit is an open source project hosted at Github.

JUnit...

JUnit is a test framework which uses annotations to identify methods that specify a test. JUnit is an open source project hosted at Github.

A JUnit test is a method contained in a class which is only used for testing. This is called a Test class. To define that a certain method is a test method, annotate it with the `@Test` annotation.

JUnit...

JUnit is a test framework which uses annotations to identify methods that specify a test. JUnit is an open source project hosted at Github.

A JUnit test is a method contained in a class which is only used for testing. This is called a Test class. To define that a certain method is a test method, annotate it with the `@Test` annotation.

This method executes the code under test...

... assert methods, provided by JUnit or another assert framework, can be used to check an expected result versus the actual result.

JUnit...

JUnit is a test framework which uses annotations to identify methods that specify a test. JUnit is an open source project hosted at Github.

A JUnit test is a method contained in a class which is only used for testing. This is called a Test class. To define that a certain method is a test method, annotate it with the `@Test` annotation.

This method executes the code under test...

... assert methods, provided by JUnit or another assert framework, can be used to check an expected result versus the actual result.

You should provide meaningful messages in assert statements. That makes it easier for the user to identify and fix the problem.

JUnit: Example

```java
package it.unicam.cs.pa.battleship19;

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

class MatrixBattleFieldTest {
  @Test
  void shouldBeValid() {
    int size = 10;
    MatrixBattleField field = new MatrixBattleField(size);
    for( int i=0 ; i<size ; i++ ) {
      for( int j=0 ; j<size ; j++ ) {
        assertTrue( field.isValid(new Location(i, j)));
      }
    }
  }
}
```

Naming conventions...

There are several potential naming conventions for JUnit tests. A widely-used solution for classes is to use the "Test" suffix at the end of test classes names.

Naming conventions...

There are several potential naming conventions for JUnit tests. A widely-used solution for classes is to use the "Test" suffix at the end of test classes names.

As a general rule, a test name should explain what the test does. If that is done correctly, reading the actual implementation can be avoided.

Naming conventions...

There are several potential naming conventions for JUnit tests. A widely-used solution for classes is to use the "Test" suffix at the end of test classes names.

As a general rule, a test name should explain what the test does. If that is done correctly, reading the actual implementation can be avoided.

One possible convention is to use the *should* in the test method name. For example:

- ordersShouldBeCreated
- menuShouldGetActive

JUnit 5...

JUnit 5 is the latest major release of JUnit.

JUnit 5...

JUnit 5 is the latest major release of JUnit.

JUnit 5 consists of a number of discrete components:

- JUnit Platform: foundation layer which enables different testing frameworks to be launched on the JVM.

JUnit 5...

JUnit 5 is the latest major release of JUnit.

JUnit 5 consists of a number of discrete components:

- JUnit Platform: foundation layer which enables different testing frameworks to be launched on the JVM.

- Junit Jupiter: is the JUnit 5 test framework which is launched by JUnit Platform.

JUnit 5...

JUnit 5 is the latest major release of JUnit.

JUnit 5 consists of a number of discrete components:

- JUnit Platform: foundation layer which enables different testing frameworks to be launched on the JVM.
- Junit Jupiter: is the JUnit 5 test framework which is launched by JUnit Platform.
- JUnit Vintage: legacy TestEngine which runs older tests.

Defining test methods

JUnit uses annotations to mark methods as test methods and to configure them:

- `@Test`, Identifies a method as a test method.

Defining test methods

JUnit uses annotations to mark methods as test methods and to configure them:

- `@Test`, Identifies a method as a test method.
- `@RepeatedTest(n)`, Repeats the test a `n` times.

Defining test methods

JUnit uses annotations to mark methods as test methods and to configure them:

- `@Test`, Identifies a method as a test method.
- `@RepeatedTest(n)`, Repeats the test a `n` times.
- `@BeforeEach`, Executed before each test. It is used to prepare the test environment (e.g., read input data, initialise the class).

Defining test methods

JUnit uses annotations to mark methods as test methods and to configure them:

- `@Test`, Identifies a method as a test method.
- `@RepeatedTest(n)`, Repeats the test a `n` times.
- `@BeforeEach`, Executed before each test. It is used to prepare the test environment (e.g., read input data, initialise the class).
- `@AfterEach`, Executed after each test. It is used to cleanup the test environment (e.g., delete temporary data, restore defaults). It can also save memory by cleaning up expensive memory structures.

Defining test methods

JUnit uses annotations to mark methods as test methods and to configure them:

- `@Test`, Identifies a method as a test method.
- `@RepeatedTest(n)`, Repeats the test a `n` times.
- `@BeforeEach`, Executed before each test. It is used to prepare the test environment (e.g., read input data, initialise the class).
- `@AfterEach`, Executed after each test. It is used to cleanup the test environment (e.g., delete temporary data, restore defaults). It can also save memory by cleaning up expensive memory structures.
- `@BeforeAll`, Executed once, before the start of all tests. It is used to perform time intensive activities, for example, to connect to a database. Methods marked with this annotation need to be defined as static to work with JUnit.

Defining test methods

JUnit uses annotations to mark methods as test methods and to configure them:

- `@Test`, Identifies a method as a test method.
- `@RepeatedTest(n)`, Repeats the test a `n` times.
- `@BeforeEach`, Executed before each test. It is used to prepare the test environment (e.g., read input data, initialise the class).
- `@AfterEach`, Executed after each test. It is used to cleanup the test environment (e.g., delete temporary data, restore defaults). It can also save memory by cleaning up expensive memory structures.
- `@BeforeAll`, Executed once, before the start of all tests. It is used to perform time intensive activities, for example, to connect to a database. Methods marked with this annotation need to be defined as static to work with JUnit.
- ...

Defining test methods

- ...
- @AfterAll, Executed once, after all tests have been finished. It is used to perform clean-up activities, for example, to disconnect from a database. Methods annotated with this annotation need to be defined as static to work with JUnit.
- @Tag("<TagName>"), Tests in JUnit 5 can be filtered by tag. Eg., run only tests with a specific tag.
- @Disabled or @Disabled("Why disabled"), Marks that the test should be disabled. This is useful when the underlying code has been changed and the test case has not yet been adapted. Or if the execution time of this test is too long to be included. It is best practice to provide the optional description, why the test is disabled.
- @DisplayName("<Name>"), <Name> that will be displayed by the test runner. In contrast to method names the DisplayName can contain spaces.

Test Suites...

To run multiple tests together, you can use test suites. They allow to aggregate multiple test classes. JUnit 5 provides two annotations:

Test Suites...

To run multiple tests together, you can use test suites. They allow to aggregate multiple test classes. JUnit 5 provides two annotations:

- @SelectPackages, used to specify the names of packages for the test suite:

```
@RunWith(JUnitPlatform.class)
@SelectPackages("it.unicam.cs.pa.battleship19.tests")
public class AllTests {}
```

Test Suites...

To run multiple tests together, you can use test suites. They allow to aggregate multiple test classes. JUnit 5 provides two annotations:

- @SelectPackages, used to specify the names of packages for the test suite:

```
@RunWith(JUnitPlatform.class)
@SelectPackages("it.unicam.cs.pa.battleship19.tests")
public class AllTests {}
```

- @SelectClasses, used to specify the classes for the test suite. They can be located in different packages.

```
@RunWith(JUnitPlatform.class)
@SelectClasses({AssertionTest.class, AssumptionTest.class
    , ExceptionTest.class})
public class AllTests {}
```

Expecting Exceptions

Exception is handling with org.junit.jupiter.api.Assertions.expectThrows(). You define the expected Exception class and provide code that should throw the exception:

```
@Test
void shouldThrowException() {
  int size = 10;
  BattleField field = new MatrixBattleField(size);
  IllegalLaunchException exception =  assertThrows(
  IllegalLaunchException.class , () -> field.launch(new
  Location(size+1,size+1)));
  assertEquals("Illegal location",exception.getMessage());
}
```

Timeout tests

If you want to ensure that a test fails if it isnt done in a certain amount of time you can use the assertTimeout() method:

Timeout tests

If you want to ensure that a test fails if it isnt done in a certain amount of time you can use the assertTimeout() method:

```
@Test
void timeoutNotExceeded() {
    assertTimeout(ofMinutes(1), () -> service.doBackup());
}
```

JUnit references...

`https://junit.org/junit5/`

https://junit.org/junit5/docs/current/user-guide/

To be continued...