Java

Unit testing with JUnit

LUKAS FELTES

TINF16B4

03.04.2018

Structure

- General Information: Junit
- Software testing
- Why use JUnit?
- Using JUnit
- Conventions
- Live Demo
- Exercise

General Information: JUnit

Junit is a software test framework

Developed by Erich Gamma and Kent Beck

Used to implement unit tests in Java

Goal accelerate programming and increase quality of code

Family of unit testing frameworks XUnit

Software testing

A piece of software, which executes another piece of software

Helps to verify that logic of a program is correct

Creates a relation ship between developing and testing

Different phases of testing: unit, integration and system testing

Unit testing on individual units of source code

Why use Junit?

Write tests quick and easy

Repeatable tests

Keeps tests small and simple to test specific areas of code

Runs tests concurrently

Makes possible to correct bugs as they are found

Using JUnit

Junit uses annotations to identify methods that specify a test

Methods contained in a class only used for testing

@Test annotation is used for specify a test method

Also other annotations are used in Junit

Using JUnit - Annotations

Annotation	Description
import org.junit.*	Import statement for using the following annotations.
@After	Executed after each test.
@AfterClass	Executed once, after all tests have been finished.
@Before	Executed before each test.
@BeforeClass	Executed once, before the start of all tests.
@Ignore	Marks that the test should be disabled.
@Test	Identifies a method as a test method.

Using Junit – Assert statements

Statement	Description
fail([message])	Let the method fail.
assertTrue([message,] boolean condition)	Checks that the boolean condition is true.
assertFalse([message,] boolean condition)	Checks that the boolean condition is false.
assertEquals([message,] expected, actual)	Tests that two values are the same.
assertEquals([message,] expected, actual, tolerance)	Test that float or double values match.
assertNull([message,] object)	Checks that the object is null.
assertNotNull([message,] object)	Checks that the object is not null.
assertSame([message,] expected, actual)	Checks that both variables refer to the same object.
assertNotSame([message,] expected, actual)	Checks that both variables refer to different objects.

Naming conventions

Use the „Test“ suffix at the of test classes

A test name should explain what it does

Use the „should “ in the test method name

Live Demo

Exercise

Thanks for your attention !