FH | JOANNEUM
University of Applied Sciences

JUnit

JUnit4

JUnit is a simple Java framework to write repeatable tests. **http://www.junit.org/**

- JUnit tests do not require human judgment to interpret, and it is easy to run many of them at the same time.
- When you need to test something, here is what you do:
  - Annotate a method with **@org.junit.Test or @Test**
  - When you want to check a value, import **org.junit.Assert.*** statically, call **assertTrue()** and pass a boolean that is true if the test succeeds.

Example

```
public class MyFirstTest {
    @Before
    public void setUp() { /*...*/ }

    @After
    public void tearDown() { /* ... */ }

    @Test
    public void firstTestMethod() {
        /* ... */

        Assert.assertTrue(/*...*/);
    }
}
```

Setup

Teardown

Exercise

Verify

JUnit Annotations

Since JUnit4, Java annotations are used to mark test, setup and teardown methods.

- **@Test** identifies a test method. A test method must return void and have no parameters.
- **@Ignore** marks test methods that will be ignored by the test runner.
- **@Before**, **@After** are used to initialize and release resources per test method.

FH | JOANNEUM
University of Applied Sciences

JUnit Annotations

- **@BeforeClass, @AfterClass** are used to initialize and release resources per test class.
- **@Test(expected=Exception.class)**
  The **@Test** annotation supports the optional **expected** parameter which declares that a test method should throw an exception. If the method doesn't throw the expected exception, the test fails.
- **@Test(timeout=1000)**
  The optional **timeout** parameter causes a test to fail if it takes longer than a specified amount of clock time (in milliseconds).

FH | JOANNEUM
University of Applied Sciences

JUnit Assert Methods

JUnit **assert methods** are helper methods which determine whether a method under test is performing correctly or not. They let us assert that some condition is true.

- **Assert.assertTrue(boolean condition)**
  Asserts that a condition is true.
- **Assert.assertFalse(boolean condition)**
  Asserts that a condition is false.
- **Assert.assertEquals(Object expected, Object actual)**
  - If **primitive** values are passed and then the values are **compared**.
  - If **objects** are passed, then the **equals() method** is invoked.

FH | JOANNEUM
University of Applied Sciences

JUnit Assert Methods

- **Assert.assertNull(Object object)**
  Asserts that an object is null.
- **Assert.assertNotNull(Object object)**
  Asserts that an object isn't null.
- **Assert.fail(String message)**
  Fails a test with the given message.
- **Assert.assertArrayEquals(Object[] exp, Object[] act)**
  Asserts that two object arrays are equal.
- **Assert.assertSame(Object expected, Object actual)**
  Asserts that two objects refer to the same object.

FH | JOANNEUM
University of Applied Sciences

JUnit Test Suite

- To run several test classes into a suite we write an empty class with @RunWith and @Suite annotations. The names of these classes are defined in the @Suite.SuiteClasses annotation.

```
@RunWith(Suite.class)
@Suite.SuiteClasses({SimpleTest.class /*,...*/})
public class AllTests { }
```

The **@RunWith** annotation is telling JUnit to use the **org.junit.runner.Suite** runner which allows us to manually build a suite containing tests from many classes.

FH | JOANNEUM
University of Applied Sciences

References

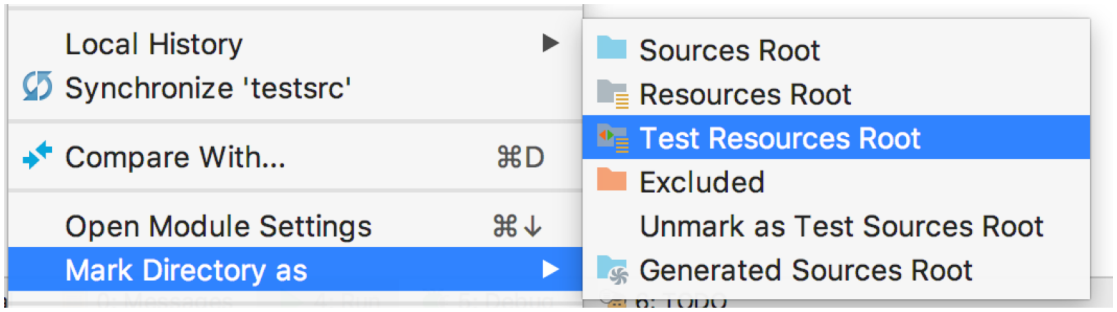
- *AndrewHunt,DavidThomas*
  Pragmatic Unit Testing
  2006, The Pragmatic Bookshelf
- *J.B.Rainsberger*
  JUnit Recipes
  2005, Manning
- GerardMeszaros
  xUnit Test Patterns, Refactoring Test Code
  2007, Addison Wesley

FH | JOANNEUM
University of Applied Sciences

JUnit with IntelliJ

Create and setup a "tests" folder

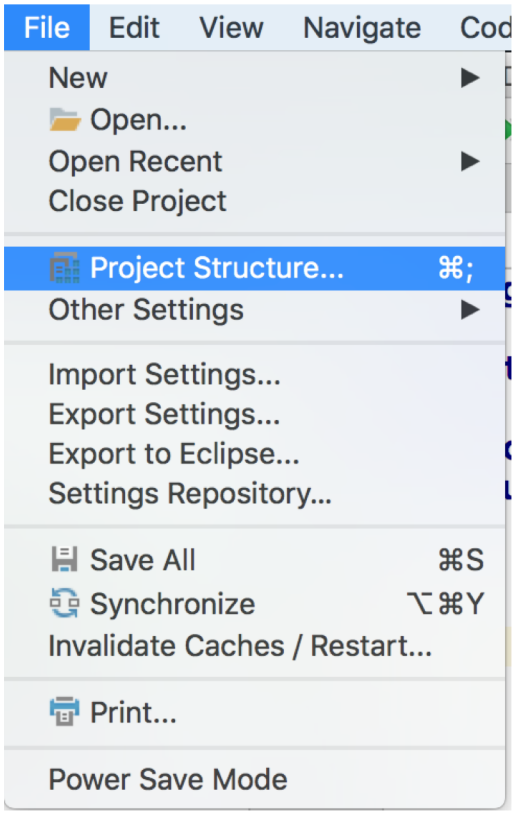
- In the Project sidebar on the left, right-click your project and do *New* > *Directory*. Name it "test" or whatever you like.
- Right-click the folder and choose *Mark Directory As* > *Test Source Root*.



FH | JOANNEUM
University of Applied Sciences

Adding JUnit library

- *File > Project Structure*
- Go to the "*Libraries*" group, click the little green plus (look up), and choose "*From Maven...*".
- Search for "junit" -- you're looking for something like "junit:junit:4.11".
- Check whichever boxes you want (Sources, JavaDocs) then hit OK.
- Keep hitting OK until you're back to the code.



FH | JOANNEUM
University of Applied Sciences

Write your unit test

- Right-click on your test folder,
  - "*New* > *Java Class*", call it whatever, e.g. MyFirstTest.
- Write a JUnit test -- here's mine:

```
import org.junit.Assert;
import org.junit.Test;

public class MyFirstTest {
    @Test
    public void firstTest() {
        Assert.assertTrue(true);
    }
}
```

Run your tests

- Right-click on your test folder and choose "Run *'All Tests*'".
- To run again, you can either hit the green "*Play*"-style button that appeared in the new section that popped on the bottom of your window, or you can hit the green "Play"-style button in the top bar.

FH | JOANNEUM
University of Applied Sciences

Slides partly taken out of

- the course **Configuration Management** by Egon Teiniker and
- the book **Java Foundations by** Lewis, DePasquale, Chase (Pearson, 4th Edition)