JUnit Framework

Terminology: assertions, annotations, fixtures

Produced by: Dr. Siobhán Drohan
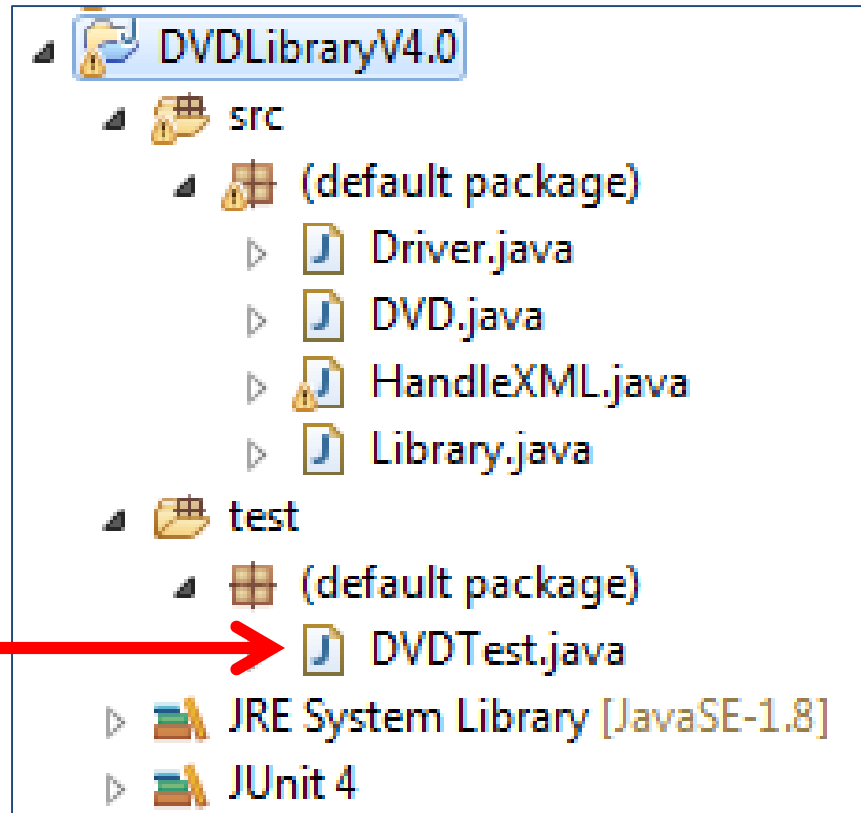   Mairead Meagher

Topic List

- General Terminology

- Assertions

- Annotations

- Fixtures

Setting up the test fixture

Tearing down the test fixture

Test Methods

```java
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class DVDTest {

    private DVD dvd1, dvd2, dvd3, dvd4;

    @Before
    public void setUp(){
        dvd1 = new DVD("The Hobbit(Director)");  //title with 20 characters
        dvd2 = new DVD("The Steve Jobs Film");   //title with 19 characters
        dvd3 = new DVD("Avatar: Directors Cut"); //title with 21 characters
        dvd4 = new DVD();
    }

    @After
    public void tearDown(){
    }

    @Test
    public void testConstructors(){
        assertEquals("The Hobbit(Director)", dvd1.getTitle());
        assertEquals("The Steve Jobs Film", dvd2.getTitle());
        assertEquals("Avatar: Directors Cu", dvd3.getTitle());
        assertEquals(null, dvd4.getTitle());
    }

    @Test
    public void testGetTitle(){
        assertEquals("The Hobbit(Director)", dvd1.getTitle());
        assertEquals("The Steve Jobs Film", dvd2.getTitle());
```
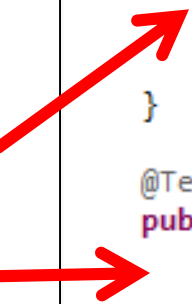
Topic List

- General Terminology

- Assertions

- Annotations

- Fixtures

First you import the Assert class from org.junit

Test Methods contain assertions

```java
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class DVDTest {

    private DVD dvd1, dvd2, dvd3, dvd4;

    @Before
    public void setUp(){
        dvd1 = new DVD("The Hobbit(Director)");  //title with 20 characters
        dvd2 = new DVD("The Steve Jobs Film");   //title with 19 characters
        dvd3 = new DVD("Avatar: Directors Cut"); //title with 21 characters
        dvd4 = new DVD();
    }

    @After
    public void tearDown(){
    }

    @Test
    public void testConstructors(){
        assertEquals("The Hobbit(Director)", dvd1.getTitle());
        assertEquals("The Steve Jobs Film", dvd2.getTitle());
        assertEquals("Avatar: Directors Cu", dvd3.getTitle());
        assertEquals(null, dvd4.getTitle());
    }

    @Test
    public void testGetTitle(){
        assertEquals("The Hobbit(Director)", dvd1.getTitle());
        assertEquals("The Steve Jobs Film", dvd2.getTitle());
```

The Assert Class

- To check if code is behaving as you expect, you use an assertion.

- An assertion is a simple method call that verifies that something is true.

- The Assert class Contains a set of assertion methods useful for writing JUnit tests.

- Only failed assertions are recorded i.e. an **AssertionError** is thrown and handled by JUnit.

http://junit.org/javadoc/latest/

The Assert Class

- These methods can be used directly:
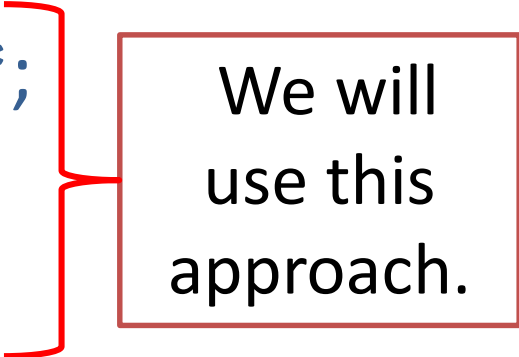
  Assert.assertEquals(…);

- However, they read better if they are referenced through a static import:

  import static org.junit.Assert.*;

  …

  assertEquals(…);

  We will use this approach.

Some common Assert methods (1)

Method Summary	
static void	**assertEquals**(double expected, double actual, double delta) Asserts that two doubles are equal to within a positive delta.
static void	**assertEquals**(long expected, long actual) Asserts that two longs are equal.
static void	**assertEquals**(Object expected, Object actual) Asserts that two objects are equal.
static void	**assertNotEquals**(double unexpected, double actual, double delta) Asserts that two doubles are **not** equal to within a positive delta.
static void	**assertNotEquals**(Object unexpected, Object actual) Asserts that two objects are **not** equals.
static void	**assertNotSame**(Object unexpected, Object actual) Asserts that two objects do not refer to the same object.
static void	**assertSame**(Object expected, Object actual) Asserts that two objects refer to the same object.

http://junit.org/javadoc/latest/

Some common Assert methods (2)

	Method Summary
static void	**assertNotNull**(Object object) Asserts that an object isn't null.
static void	**assertNotNull**(String message, Object object) Asserts that an object isn't null.
static void	**assertNull**(Object object) Asserts that an object is null.
static void	**assertFalse**(boolean condition) Asserts that a condition is false.
static void	**assertTrue**(String message, boolean condition) Asserts that a condition is true.
static void	**fail**() Fails a test with no message.
static void	**fail**(String message) Fails a test with the given message.

http://junit.org/javadoc/latest/

assertEquals in DVDTest.java

```java
 DVDTest.java ⅹ   LibraryTest.java    DVD.java
 1 import static org.junit.Assert.*;

 5
 6 public class DVDTest {

 7
 8     private DVD dvd1, dvd2, dvd3;

 9
10     @Before
11     public void setUp(){
12         dvd1 = new DVD("The Hobbit(Director)");  //title with 20 characters
13         dvd2 = new DVD("The Steve Jobs Film");   //title with 19 characters
14         dvd3 = new DVD("Avatar: Directors Cut"); //title with 21 characters
15     }
16
18     public void tearDown(){
20
22     public void testConstructors(){
27
29     public void testGetTitle(){
34
35     @Test
36     public void testSetTitle(){
37         dvd1.setTitle("The Hobbit");
38         assertEquals ("The Hobbit", dvd1.getTitle());
39         dvd1.setTitle("The Hobbit (Director)");  //attempting to set title to 21 characters
40         assertEquals ("The Hobbit (Director", dvd1.getTitle());
41         dvd1.setTitle("The Hobbit(Director)");  //attempting to set title to 20 characters
42         assertEquals ("The Hobbit(Director)", dvd1.getTitle());
43         dvd1.setTitle("The Hobbit:Director");  //attempting to set title to 20 characters
44         assertEquals ("The Hobbit:Director", dvd1.getTitle());
45     }
46
48     public void testToString(){
53
54 }
```

Recall the generated DVDTest.java?

```java
import static org.junit.Assert.*;

import org.junit.Test;

public class DVDTest {

    @Test
    public void test() {
        fail("Not yet implemented");
    }

}
```

Topic List

- General Terminology

- Assertions

- Annotations

- Fixtures

These methods have Annotations

@someword

```java
DVD.java        *DVDTest.java ⊠

import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class DVDTest {

    private DVD dvd1, dvd2, dvd3, dvd4;

    @Before
    public void setUp(){
        dvd1 = new DVD("The Hobbit(Director)");  //title with 20 characters
        dvd2 = new DVD("The Steve Jobs Film");   //title with 19 characters
        dvd3 = new DVD("Avatar: Directors Cut"); //title with 21 characters
        dvd4 = new DVD();
    }

    @After
    public void tearDown(){
    }

    @Test
    public void testConstructors(){
        assertEquals("The Hobbit(Director)", dvd1.getTitle());
        assertEquals("The Steve Jobs Film", dvd2.getTitle());
        assertEquals("Avatar: Directors Cu", dvd3.getTitle());
        assertEquals(null, dvd4.getTitle());
    }

    @Test
    public void testGetTitle(){
        assertEquals("The Hobbit(Director)", dvd1.getTitle());
        assertEquals("The Steve Jobs Film", dvd2.getTitle());
```

What are Annotations?

- Annotations provide data about a program that is not part of the program itself. They have no direct effect on the operation of the code they annotate.

- Annotations can be applied to a program's declarations of classes, fields, methods, and other program elements.

What are Annotations?

- Annotations have a number of uses, among them:

    - Information for the compiler — Annotations can be used by the compiler to detect errors or suppress warnings.

    - Compiler-time and deployment-time processing — Software tools can process annotation information to generate code, XML files, and so forth.

    - Runtime processing — Some annotations are available to be examined at runtime.

Import the required Annotation class(es) from org.junit.

@Before – run the method before each test.

@After – run the method after each test.

@Test -identifies that a method is a test method.

```java
 D.java        J *DVDTest.java ⊠

import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class DVDTest {

    private DVD dvd1, dvd2, dvd3, dvd4;

    @Before
    public void setUp(){
        dvd1 = new DVD("The Hobbit(Director)");  //title with 20 characters
        dvd2 = new DVD("The Steve Jobs Film");   //title with 19 characters
        dvd3 = new DVD("Avatar: Directors Cut"); //title with 21 characters
        dvd4 = new DVD();
    }

    @After
    public void tearDown(){
    }

    @Test
    public void testConstructors(){
        assertEquals("The Hobbit(Director)", dvd1.getTitle());
        assertEquals("The Steve Jobs Film", dvd2.getTitle());
        assertEquals("Avatar: Directors Cu", dvd3.getTitle());
        assertEquals(null, dvd4.getTitle());
    }

    @Test
    public void testGetTitle(){
        assertEquals("The Hobbit(Director)", dvd1.getTitle());
        assertEquals("The Steve Jobs Film", dvd2.getTitle());
```

Other Useful Junit Annotations

@BeforeClass **public void method()**	Will execute the method once, before the start of all tests. This can be used to perform time intensive activities, for example to connect to a database.
@AfterClass **public void method()**	Will execute the method once, after all tests have finished. This can be used to perform clean-up activities, for example to disconnect from a database.
@Ignore	Will ignore the test method. 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.

http://crunchify.com/simple-junit-4-tutorial-hello-world-example/

Topic List

- General Terminology

- Assertions

- Annotations

- Fixtures

Test fixture fields.

Setting up the test fixture.

Tearing down the test fixture.

```java
                                                              *DVDTest.java  ⊠
D.java

import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class DVDTest {

    private DVD dvd1, dvd2, dvd3, dvd4;

    @Before
    public void setUp(){
        dvd1 = new DVD("The Hobbit(Director)");   //title with 20 characters
        dvd2 = new DVD("The Steve Jobs Film");    //title with 19 characters
        dvd3 = new DVD("Avatar: Directors Cut"); //title with 21 characters
        dvd4 = new DVD();
    }

    @After
    public void tearDown(){
    }

    @Test
    public void testConstructors(){
        assertEquals("The Hobbit(Director)", dvd1.getTitle());
        assertEquals("The Steve Jobs Film", dvd2.getTitle());
        assertEquals("Avatar: Directors Cu", dvd3.getTitle());
        assertEquals(null, dvd4.getTitle());
    }

    @Test
    public void testGetTitle(){
        assertEquals("The Hobbit(Director)", dvd1.getTitle());
        assertEquals("The Steve Jobs Film", dvd2.getTitle());
```

Fixtures

- A **Fixture** is a fixed state of a set of objects used as a baseline for running tests.

- Test fixtures allow tests to share common test data.

- The purpose of a test fixture is to ensure that there is a fixed environment in which tests are run so that results are repeatable.

- It includes:
  - setUp() method which runs before every test method.
  - tearDown() method which runs after every test method.

Waterford Institute *of* Technology

INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/