

Agile Software Development

Produced
by

Eamonn de Leastar (edeleastar@wit.ie)



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRCE

JUnit Annotations

Test Driven Development Introduction

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

Using Annotations

- The annotation appears first, often (by convention) on its own line, and may include elements with named or unnamed values.
- The annotation must itself be already defined and explicitly imported if necessary:
- Annotations are defined using a special syntax:

```
@Author(name = "Joe Kelly", date = "3/27/2003")  
public class MyClass  
{  
    //...  
}
```

```
import documentation.Author;
```

```
package documentation;  
  
public @interface Author  
{  
    String name();  
    String date();  
}
```

Built in Annotations

- There are three annotation types that are predefined by the language specification itself:
 - `@Deprecated`— indicates that the marked element is deprecated and should no longer be used. The compiler generates a warning whenever a program uses a method, class, or field with the `@Deprecated` annotation.
 - `@Override` annotation informs the compiler that the element is meant to override an element declared in a superclass. It not required to use this annotation when overriding a method, it helps to prevent errors. If a method marked with `@Override` fails to correctly override a method in one of its superclasses, the compiler generates an error.
 - `@SuppressWarnings` annotation tells the compiler to suppress specific warnings that it would otherwise generate

JUnit 3

- The previous slides used JUnit 3 conventions.
- Test class extend TestCase
- setUp/tearDown are overridden from TestCase
- test methods must begin with “test” word.

```
import junit.framework.TestCase;

public class TestLargest extends TestCase
{
    private int[] arr;

    public TestLargest (String name)
    {
        super(name);
    }

    public void setUp()
    {
        arr = new int[] {8,9,7};
    }

    public void tearDown()
    {
        arr = null;
    }

    public void testOrder ()
    {
        assertEquals(9, Largest.largest(arr));
    }

    public void testOrder2 ()
    {
        assertEquals(9, Largest.largest(new int[] { 9, 8, 7 }));
        assertEquals(9, Largest.largest(new int[] { 8, 9, 7 }));
        assertEquals(9, Largest.largest(new int[] { 7, 8, 9 }));
    }
}
```

JUnit 4 Uses Annotations

- @Before - run before each test
- @After - run after each test
- @Test - the test itself
- No need to extend TestCase

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.fail;
import static org.junit.Assert.assertTrue;
import static org.junit.Assert.assertEquals;

public class TestLargest
{
    private int[] arr;

    @Before
    public void setUp()
    {
        arr = new int[] {8,9,7};
    }

    @After
    public void tearDown()
    {
        arr = null;
    }

    @Test
    public void order ()
    {
        assertEquals(9, Largest.largest(arr));
    }

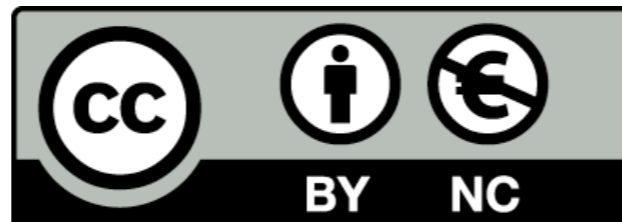
    @Test
    public void dups ()
    {
        assertEquals(9, Largest.largest(new int[] { 9, 7, 9, 8 }));
    }
}
```

Exceptions: JUnit 3 vs JUnit 4

- Use `@Test (expected = ...)` to specify exception
- Simpler, less verbose

```
public void testEmpty ()
{
    try
    {
        Largest.largest(new int[] {});
        fail("Should have thrown an exception");
    }
    catch (RuntimeException e)
    {
        assertTrue(true);
    }
}
```

```
@Test (expected = RuntimeException.class)
public void testEmpty ()
{
    Largest.largest(new int[] {});
}
```

Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>

