

JUnit in the context and JUnit Format

TAN, Shin Hwei

陈馨慧

Southern University of Science and Technology

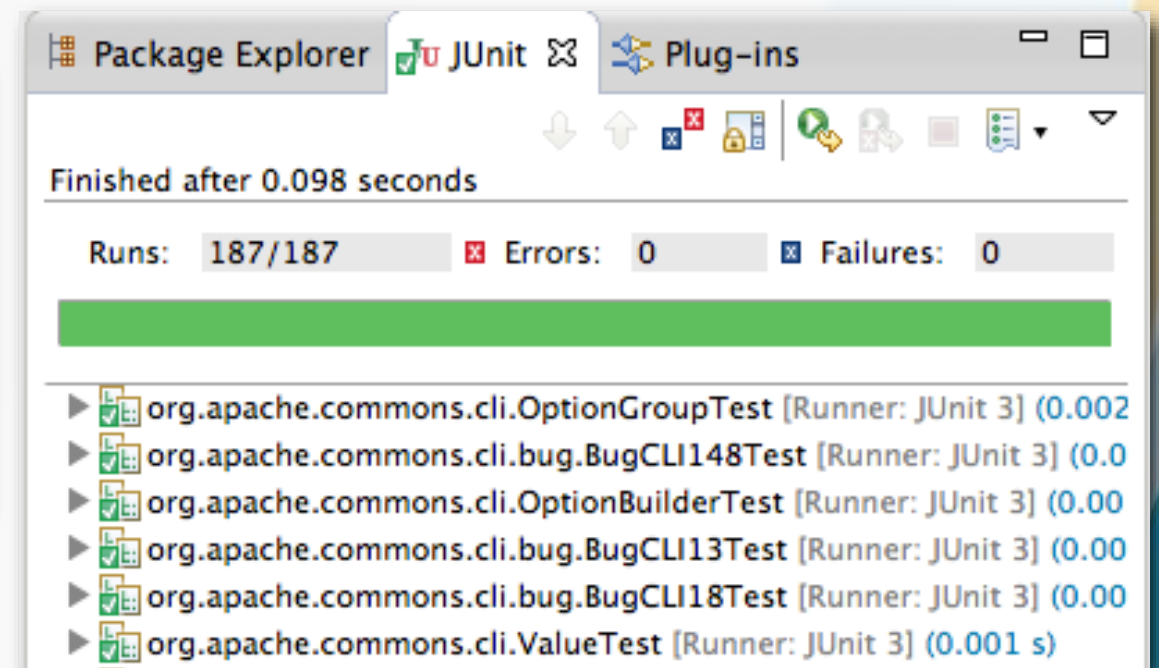
Slides adapted from cs4218 (NUS)

Credits: Konstantin Rubinov

JUnit is a format and a framework for Java programs

by Erich Gamma and Kent Beck

```
@Test
public void testAdd() {
    Calculator calculator = new Calculator();
    double result = calculator.add(10, 50);
    assertEquals(60, result, 0);
}
```

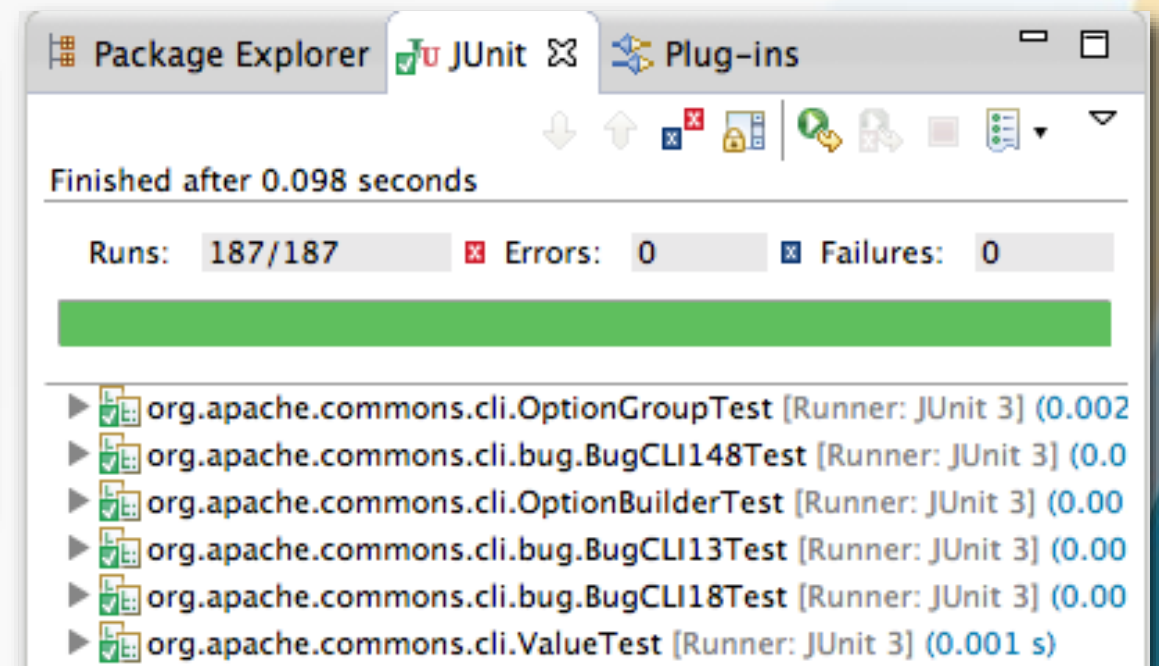


for specifying and executing test cases
automatically

JUnit is a format and a framework for Java programs

by Erich Gamma and Kent Beck

```
@Test
public void testAdd() {
    Calculator calculator = new Calculator();
    double result = calculator.add(10, 50);
    assertEquals(60, result, 0);
}
```



JUnit Framework is integrated into Eclipse through a graphical plug-in and it provides a set of classes and conventions

JUnit test cases are Java code

```
@Test
public void testAdd() {
    Calculator calculator = new Calculator();
    double result = calculator.add(10, 50);
    assertEquals(60, result, 0);
}
}
```

Why JUnit?

Developer testing is more advanced and simpler than println-ing

Why JUnit?

- Naturally supports continuous testing
- Aids fault localization
- Automates test execution process

Developer testing is more advanced and simpler than println-ing

Why JUnit?

- Aids alternative thinking about system under test
- Is a software specification (intended behavior)
- Consolidates test results and execution statistics

JUnit testing

```
@Test
public void testAdd() {
    Calculator calculator = new Calculator();
    double result = calculator.add(10, 50);
    assertEquals(60, result, 0);
}
}
```

VS.

Println-ing

```
public class CalculatorTest {
    public static void main(String[] args) {
        Calculator calculator = new Calculator();
        double result = calculator.add(10, 50);
        if (result != 60) {
            System.out.println("Bad result: " + result);
        }
    }
}
```


JUnit testing

```
@Test
public void testAdd() {
    Calculator calculator = new Calculator();
    double result = calculator.add(10, 50);
    assertEquals(60, result, 0);
}
}
```

vs.

Println-ing

- unnecessary main() method
- placing and then removing 'System.out.println(...)'
- inspecting output manually

JUnit testing

- appropriate use of `main()` method
- automatic test execution and generated reports
- no need for manual inspection

vs.

Println-ing

- unnecessary `main()` method
- placing and then removing `'System.out.println(...)'`
- inspecting output manually

String Wrapper testing:

```
/**  
 * Purpose: program for wrapping strings on spaces and  
 indenting strings if we break them before '+' or '=' symbols as in Java.  
 */  
public class Wrapper {  
    public String wrap(String s, int length) {  
        ...  
    }  
}
```

‘Developer testing’ perspective helps to see
system behavior in new ways

```
@Test
public void testWrapNull() {
    assertEquals("", wrapper.wrap(null, 10));
}
```

```
@Test
public void ThreeWordsOverTheLimitShouldWrapAtSecondWord() {
    assertEquals("word word\nword", wrapper.wrap("word word word", 5));
}
```

```
@Test
public void TwoWordsLongerThanLimitShouldNotWrap() {
    assertEquals("word word", wrapper.wrap("word word", 6));
}
```

```
@Test
public void testWrapNull() {
    assertEquals("", wrapper.wrap(null, 10));
}
```

```
@Test
public void ThreeWordsOverTheLimitShouldWrapAtSecondWord() {
    assertEquals("word word\nword", wrapper.wrap("word word
word", 5));
}
```

```
@Test
public void TwoWordsLongerThanLimitShouldNotWrap() {
```

How should we wrap a string of spaces?

```
@Test
public void testWrapConsecutiveSpaces() {
    assertEquals("  ", wrapper.wrap("  ", 2));
}
```

```
@Test
public void testWrapNull() {
    assertEquals("", wrapper.wrap(null, 10));
}
```

```
@Test
public void ThreeWordsOverTheLimitShouldWrapAtSecondWord() {
```

```
}
```

JUnit test case as a specification of intended behavior

```
public void TwoWordsLongerThanLimitShouldNotWrap() {
    assertEquals("word word", wrapper.wrap("word word", 6));
}
```

```
@Test
public void testWrapConsecutiveSpaces() {
    assertEquals("  ", wrapper.wrap("  ", 2));
}
```

Package Explorer JUnit Plug-ins

Finished after 5.297 seconds

Runs: 537/537 Errors: 0 Failures: 4

- org.jgrapht.experimental.dag.AcyclicGraphTest (0.000 s)
- org.jgrapht.experimental.dag.AcyclicGraphTest (0.676 s)
- org.jgrapht.experimental.equivalence.EquivalenceGroupCreator
- org.jgrapht.experimental.graphReaderTest (0.003 s)
- org.jgrapht.experimental.isomorphism.IsomorphismInspectorT
- org.jgrapht.experimental.permutation.CompoundPermutationIt
- org.jgrapht.ext.DOTExporterTest (0.004 s)
- org.jgrapht.ext.GmlExporterTest (0.003 s)
- org.jgrapht.ext.GraphMLExporterTest (0.007 s)
- testUndirected (0.007 s)
- org.jgrapht.ext.MatrixExporterTest (0.001 s)
- org.jgrapht.generate.GraphGeneratorTest (0.000 s)
- org.jgrapht.generate.RandomGraphC
- org.jgrapht.graph.AsUndirectedGrap
- org.jgrapht.graph.AsUnweightedGrap
- org.jgrapht.graph.AsWeightedGrap
- org.jgrapht.graph.CloneTest (0.001 s)
- org.jgrapht.graph.DefaultDirectedGr
- org.jgrapht.graph.GenericGraphs

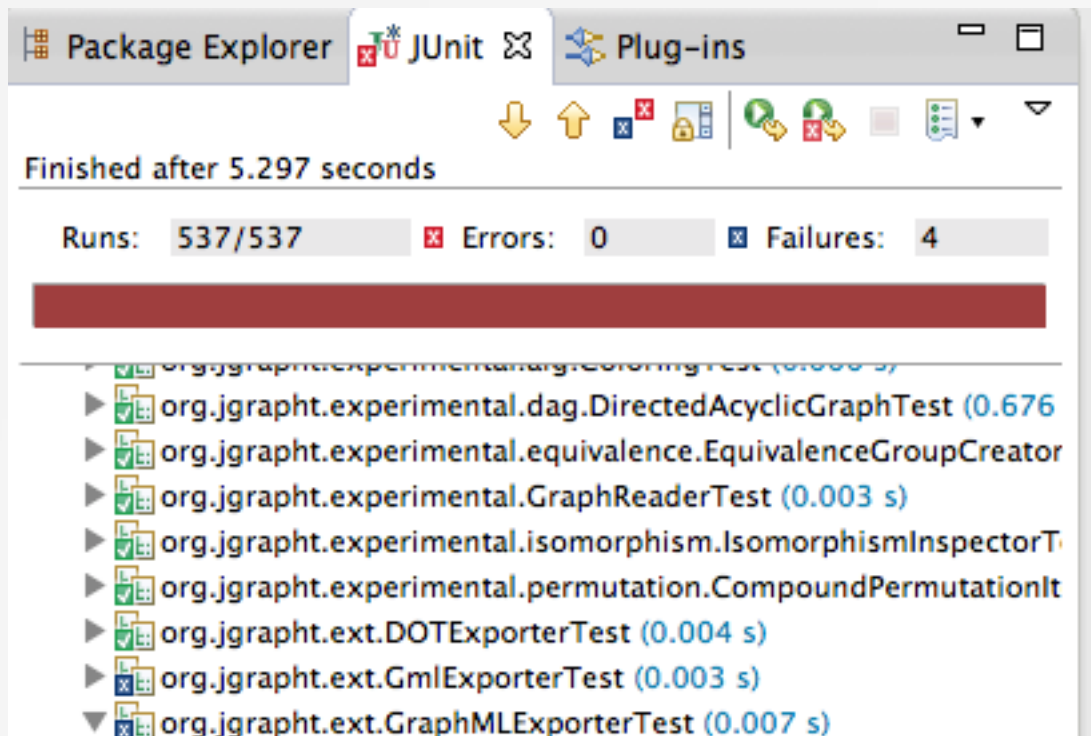
Failure Trace

junit.framework.AssertionFailedError: org.custommonkey.xmlunit.Dif
[different] Expected attribute value '1' but was '3' - comparing <edge

at org.custommonkey.xmlunit.XMLAssert.assertXMLEqual(XMLAssert
at org.custommonkey.xmlunit.XMLAssert.assertXMLEqual(XMLAssert
at org.jgrapht.ext.GraphMLExporterTest.testUndirected(GraphMLExp

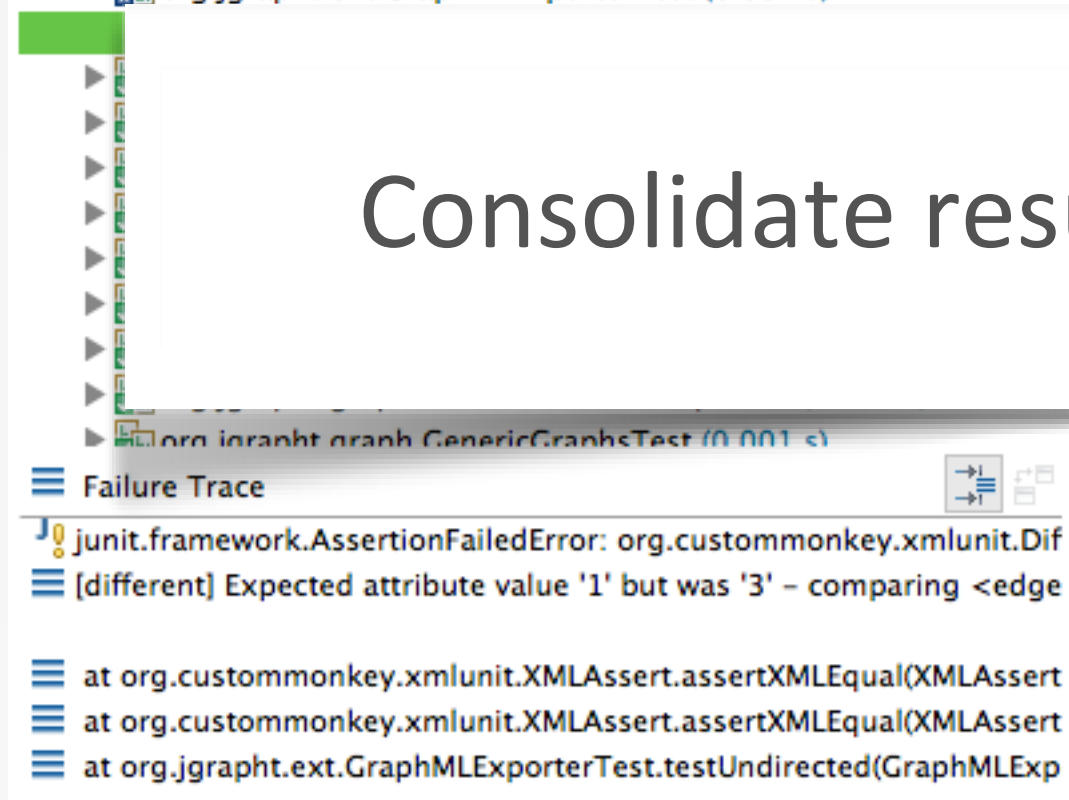
```
81 public void testUndirected()  
82     throws Exception  
83 {  
84     UndirectedGraph<String, DefaultEdge> g =  
85         new SimpleGraph<String, DefaultEdge>(DefaultEdge.class);  
86     g.addVertex(V1);  
87     g.addVertex(V2);  
88     g.addEdge(V1, V2);  
89     g.addVertex(V3);  
90     g.addEdge(V3, V1);  
91  
92     StringWriter w = new StringWriter();  
93     exporter.export(w, g);  
94  
95     if (System.getProperty("java.vm.version").startsWith("1.4")) {  
96         // NOTE jvs 16-Mar-2007: XML prefix mapping comes out  
97         // with missing info on 1.4, so skip the verification part  
98         // of the test.  
99         return;  
100    }  
101  
102    XMLAssert.assertXMLEqual(UNDIRECTED, w.toString());  
103 }  
104 }
```

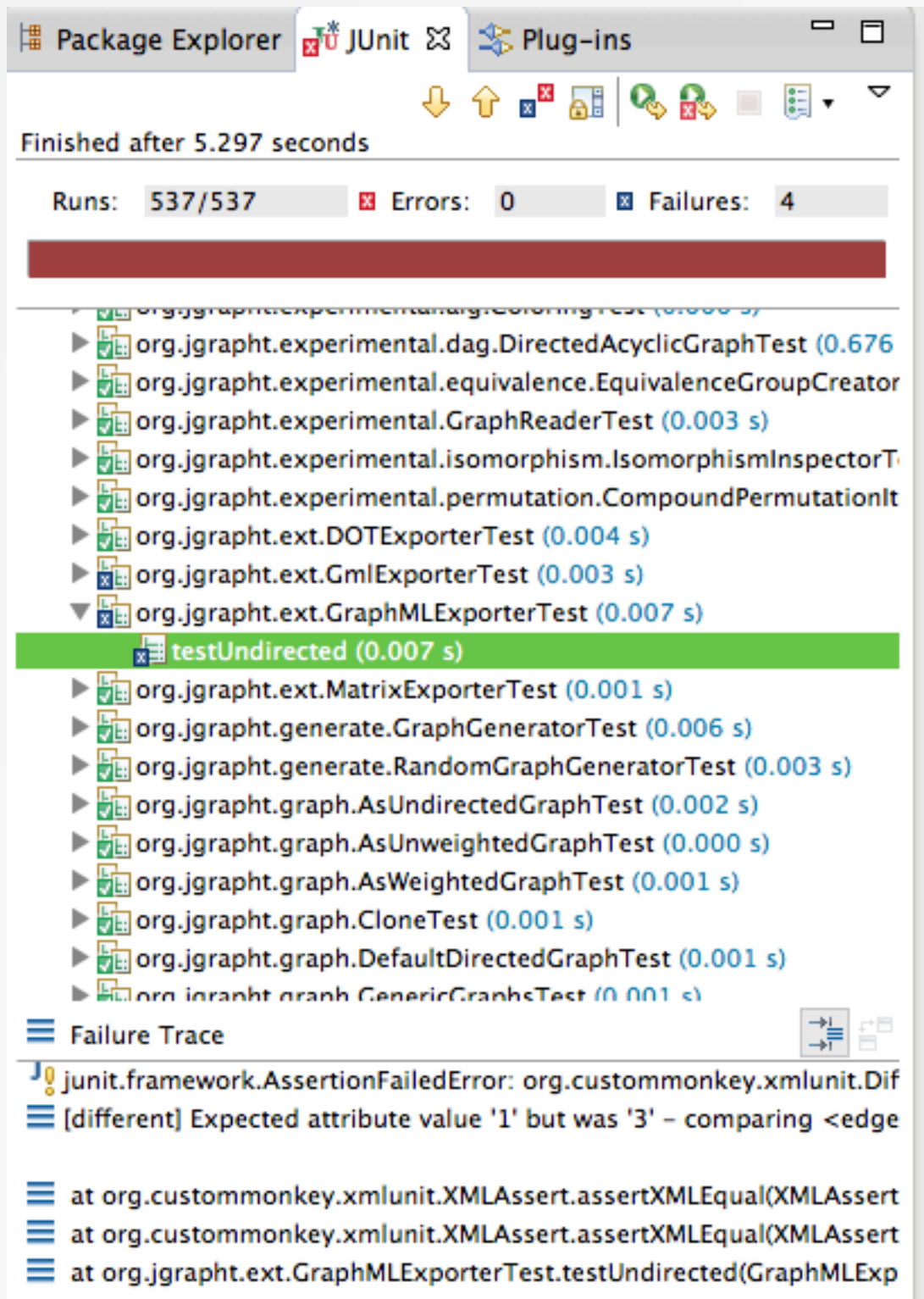
Fault localization



- Execution time
- Failures/Errors
- Stack traces
- Contrast expected and actual

Consolidate results and provide statistics





Failure vs. Error

Failure: if assertion is violated

Error: compilation problems, unexpected exception

Elements of a Test case

@test method

Methods annotated with @Test are executed by JUnit

@Test

```
public void testWrapNull() {  
    Wrapper wrapper = new Wrapper();  
    String result = wrapper.wrap(null, 10);  
    assertEquals("", result);  
}
```

Logical structure

- Object under test
- Method under test with parameters
- Comparison of expected and actual data

@Test

```
public void testWrapNull() {  
    Wrapper wrapper = new Wrapper();  
    String result = wrapper.wrap(null, 10);  
    assertEquals("", result);  
}
```

What?
When?
Then...

Logic:

What?
When?
Then...

assert:

a family of methods to check conditions

```
assertEquals("", result);
```

“check if expected and actual values are equal”



@Test

```
public void testWrapNull() {  
    Wrapper wrapper = new Wrapper();  
    String result = wrapper.wrap(null, 10);  
    assertEquals("", result);  
}
```

assert:

a family of methods to check conditions

```
assertEquals("", result);
```

“check if expected and actual values are equal”



If assertion fails (checked condition is not satisfied):

- current test is marked as ‘failed’
- JUnit skips execution of the rest of the test method and proceeds executing remaining test methods

assert

- for a boolean condition:
`assertTrue("message for fail", condition);`
`assertFalse("message", condition);`
- for object, int, long, and byte values, array:
`assertEquals(expected_value, expression);`
- for float and double values:
`assertEquals(expected, expression, error);`
- for objects references:
`assertNull(obj_ref)`
`assertNotNull(obj_ref)`
`assertSame(obj_ref, obj_ref2)`

...

assert: example

@Test

```
public void testPush() {  
    Stack aStack = new Stack();  
    assertTrue("Stack should be empty!",  
                aStack.isEmpty());  
    aStack.push(10);  
    assertFalse("Stack should not be empty!",  
                aStack.isEmpty());  
    aStack.push(4);  
    assertEquals(4, aStack.pop());  
    assertEquals(10, aStack.pop());  
}
```

assert: better example

@Test

```
public void testStackEmpty() {  
    Stack aStack = new Stack();  
    assertTrue("Stack should be empty!", aStack.isEmpty());  
    aStack.push(10);  
    assertFalse("Stack should not be empty!", aStack.isEmpty());  
}
```

@Test

```
public void testStackOperations() {  
    Stack aStack = new Stack();  
    aStack.push(10);  
    aStack.push(-4);  
    assertEquals(-4, aStack.pop());  
    assertEquals(10, aStack.pop());  
}
```

Separate @Test
methods for testing
individual scenarios
and methods

assertEquals vs. assertEquals

for values:

```
assertEquals(expected_value, expression);  
assertEquals(expected, expression, error);
```

for objects references:

```
assertEquals(obj_ref, obj_ref2)
```

Important distinction:

- assertEquals() and assertEquals() are used for different purposes
- careful when comparing Strings

assertEquals vs. assertEquals

Will the test fail?
Where?

```
@Test
public void testEquality() {
    String a = "abcde";
    String b = new String(a);
    assertTrue(a.equals(b));
    assertFalse(a == b);
    assertEquals(a, b);

    String c = "abcde";
    assertNotSame(a, b);
    assertEquals(a, c);
}
```

Important distinction:

- assertEquals() and assertEquals() are used for different purposes
- be prudent when comparing Strings

Test class

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

public class CalculatorTest {

    @Test
    public void testAdd() {
        Calculator calculator = new Calculator();
        double result = calculator.add(10, 50);
        assertEquals(60, result, 0);
    }
}
```

TestMethod \in TestClass

```

public class ListTest {
    protected List<Integer> fEmpty;
    protected List<Integer> fFull;
    protected static List<Integer> fgHeavy;

    @BeforeClass
    public static void setUpOnce() {
        fgHeavy = new ArrayList<Integer>();
        for (int i = 0; i < 1000; i++) {
            fgHeavy.add(i);
        }
    }

    @Before
    public void setUp() {
        fEmpty = new ArrayList<Integer>();
        fFull = new ArrayList<Integer>();
        fFull.add(1);
        fFull.add(2);
        fFull.add(3);
    }

    @Test
    public void testCopy() {
        List<Integer> copy = new ArrayList<Integer>(fFull.size());
        copy.addAll(fFull);
        assertTrue(copy.size() == fFull.size());
        assertTrue(copy.contains(1));
    }

    @Test(expected = IndexOutOfBoundsException.class)
    public void elementAt() {
        int i = fFull.get(0);
        assertTrue(i == 1);
        fFull.get(fFull.size()); // Should throw IndexOutOfBoundsException
    }

    @Test
    public void removeAll() {
        fFull.removeAll(fFull);
        fEmpty.removeAll(fEmpty);
        assertTrue(fFull.isEmpty());
        assertTrue(fEmpty.isEmpty());
    }
}

```

A public class
that aggregates
@Test methods
and special
auxiliary JUnit
methods (setup
and tear-down)

Test class

```

@Test
public void testCopy() {
    List<Integer> copy = new ArrayList<Integer>(fFull.size());
    copy.addAll(fFull);
    assertTrue(copy.size() == fFull.size());
    assertTrue(copy.contains(1));
}

@Test(expected = IndexOutOfBoundsException.class)
public void elementAt() {
    int i = fFull.get(0);
    assertTrue(i == 1);
    fFull.get(fFull.size()); // Should throw IndexOutOfBoundsException
}

@Test
public void removeAll() {
    fFull.removeAll(fFull);
    fEmpty.removeAll(fEmpty);
    assertTrue(fFull.isEmpty());
    assertTrue(fEmpty.isEmpty());
}

```

@Test methods

```

public class ListTest {
    protected List<Integer> fEmpty;
    protected List<Integer> fFull;
    protected static List<Integer> fgHeavy;

    @Before
    public void setup() {
        fEmpty = new ArrayList<Integer>();
        fFull = new ArrayList<Integer>();
        fFull.add(1);
        fFull.add(2);
        fFull.add(3);
    }

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

```

Setup methods
prepare common
objects/resources for
test execution

Tear-down
methods release
common
objects/resources
after test execution

Auxiliary setup and tear-down methods

@Before	@After
---------	--------


```
public class ListTest {  
    protected List<Integer> fEmpty;  
    protected List<Integer> fFull;  
    protected static List<Integer> fgHeavy;
```

```
@BeforeClass  
public static void setUpOnce() {  
    fgHeavy = new ArrayList<Integer>();  
    for (int i = 0; i < 1000; i++) {  
        fgHeavy.add(i);  
    }  
}
```

```
@Before  
public void setUp() {  
    fEmpty = new ArrayList<Integer>();  
    fFull = new ArrayList<Integer>();  
    fFull.add(1);  
    fFull.add(2);  
    fFull.add(3);  
}
```

setup

executed
once for a
test class

executed
before
every test
method

Aid code reuse - extract common object
creation/release from test methods

```
@Test
public void contains() {
    assertTrue(fFull.contains(1));
    assertTrue(!fEmpty.contains(1));
}
```

```
@Test
public void removeAll() {
    fFull.removeAll(fFull);
    fEmpty.removeAll(fEmpty);
    assertTrue(fFull.isEmpty());
    assertTrue(fEmpty.isEmpty());
}
```

If `removeAll()` test case is executed first, then `contains()` test case will give wrong results!

Caution: common objects may cause unwanted dependence between test methods

```
@Test
public void contains() {
    assertTrue(fFull.contains(1));
    assertTrue(!fEmpty.contains(1));
}
```

```
@Test
public void removeAll() {
    fFull.removeAll(fFull);
    fEmpty.removeAll(fEmpty);
    assertTrue(fFull.isEmpty());
    assertTrue(fEmpty.isEmpty());
}
```

If removeAll() test case is executed first, then contains() test case will give wrong results!

Caution: common setup/tear-down may cause unwanted dependence between test methods

```
@Test
public void contains() {
    assertTrue(fFull.contains(1));
}
```

If removeAll() test

It is a developer (your) responsibility to make sure test methods are isolated from each other!

```
    fFull.removeAll(fFull);
    fEmpty.removeAll(fEmpty);
    assertTrue(fFull.isEmpty());
    assertTrue(fEmpty.isEmpty());
}
```

test case will
give wrong
results!

Caution: common setup/tear-down may cause unwanted dependence between test methods

TestClass \in TestSuite

```
@RunWith(Suite.class)
@SuiteClasses({CSVRendererTest.class, EmacsRendererTest.class, XMLRendererTest.class,
TextPadRendererTest.class})
public class RenderersTests {
}
```

Allows grouping test classes in different sets for test execution

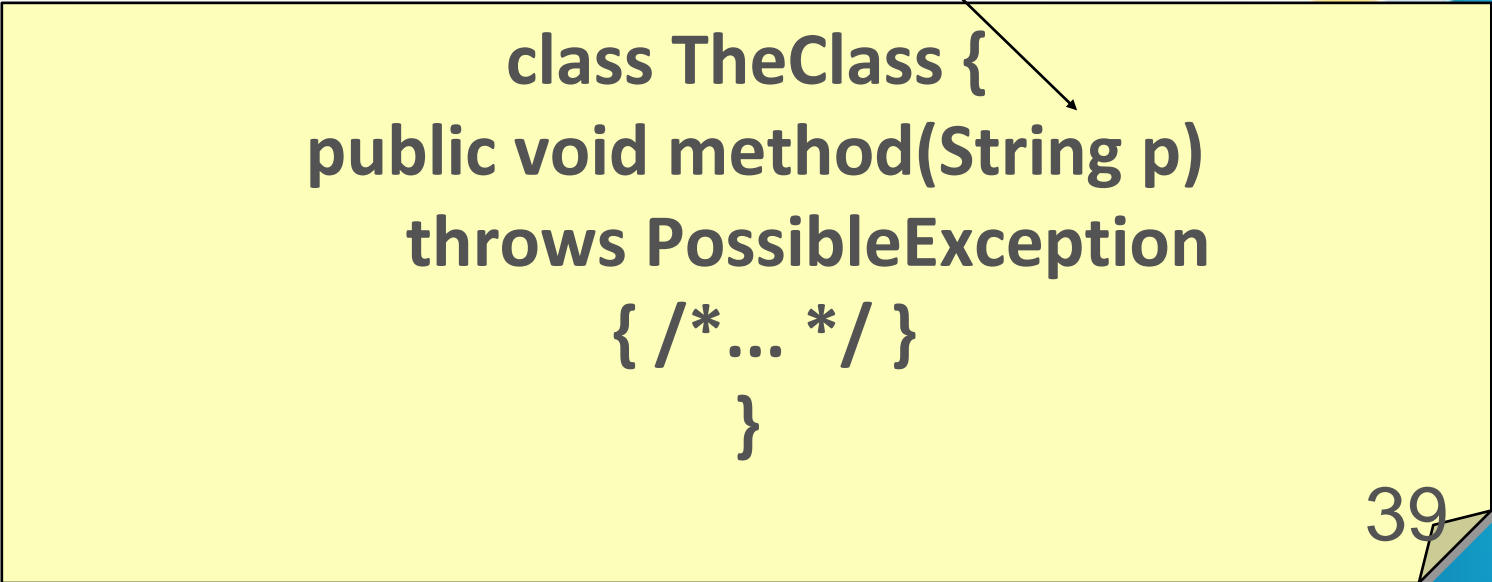
Special JUnit runner -> `org.junit.runners.Suite.class`

Test of “Exceptions”

- There are two cases:
 1. We expect a normal behavior and then no exceptions.
 2. We expect an anomalous behavior and then an exception.

We expect a normal behavior ...

```
try {  
    // We call the method with correct parameters  
    object.method("Parameter");  
    assertTrue(true); // OK  
} catch(PossibleException e){  
    fail("method should not fail !!!");  
}
```



```
class TheClass {  
    public void method(String p)  
        throws PossibleException  
        { /*...*/ }  
}
```

We expect an exception ...

```
try {  
    // we call the method with wrong parameters  
    object.method(null);  
    fail("method should fail!!");  
} catch(PossibleException e){  
    assertTrue(true); // OK  
}
```

```
class TheClass {  
    public void method(String p)  
        throws PossibleException  
        { /*...*/ }  
}
```


Do you know about what is it called when two programmers work together at one workstation in agile?

Pair Programming

Let's try “Pair Testing”

Lab exercise





Simple TriType Example

- TriTyp: Given three integers for the lengths of the sides of a triangle, find the type of triangle
- Try testing the “Triang(int, int,int)” method
- Go to this link to get the program:

<https://classroom.github.com/a/0EgnbwO5>

This is an graded (passed/failed) lab assignment. You get full score as long as you accept the invitation link and commit your test (“TriTypTest.java” file) at the last step.

Pair Testing

1. Form a team of two with your neighbors 
2. Write two JUnit tests for TriTyp individually 
3. Write two JUnit tests for TriTyp to check for corner/boundary cases individually 
4. Compile & run your tests & theory to see if they are passing 
5. Show all your tests to your teammate and try the following:
 - a) Convince your teammate that your tests are better
 - b) Convince your teammate that your tests are testing different behavior than his/her tests
 - c) Convince your teammate that your tests are of the same quality
 - d) Let your teammate know if you feel that his/her test is better
6. Combine all the final tests and save as “TriTypTest.java” file. Commit the file to GitHub.

How many tests you have now?

- 2 JUnit Tests?
- 3 JUnit Tests?
- 4 JUnit Tests?
- > 4 JUnit Tests?

How can unit tests help?

Watch the video on how to start working on open-source project (use tests to help you)

- International student:
- <https://www.youtube.com/watch?v=k1T5Wbx0NMw>
- Local student:
- <https://www.bilibili.com/video/BV1dc411h7VH>

How can unit tests help?

The screenshot shows the F-Droid website interface. At the top, there's a navigation bar with links: 浏览 (Browse), 论坛 (Forum), 文档 (Documentation), 新闻 (News), 反馈 (Feedback), 贡献 (Contribute), and 关于 (About). Below this, the main heading is "F-Droid 是什么?" (What is F-Droid?). The text explains that F-Droid is an Android platform FOSS (Free and Open Source Software) directory providing download and installation support. A blue button labeled "下载 F-DROID" (Download F-DROID) is present, along with a QR code and a link to the PGP signature. A central image shows a smartphone displaying the F-Droid app interface with various application tiles like "终端模拟器" (Terminal Emulator), "维基百科" (Wikipedia), "AntennaPod", and "条码扫描器" (Barcode Scanner). On the right side, there's a search bar labeled "查找程序" (Find app) and a "搜索" (Search) button. Below the search bar, there's a "捐赠" (Donate) section with the text "您的捐赠将是 F-Droid 运作的驱动力!" (Your donation will be the driving force for F-Droid's operation!). It includes a "DONATE TO OUR COLLECTIVE" button, a "lp Donate" button, and a Bitcoin address: 15u8aAPK4jJ5N8wpWJ5gutAyyeHtKX5i18. At the bottom, it provides bank transfer details: 使用银行转账 (Use bank transfer), 名称: F-Droid Limited (Name: F-Droid Limited), IBAN: GB92 BARC 2056 7893 4088 92, and SWIFT: BARCGB22.