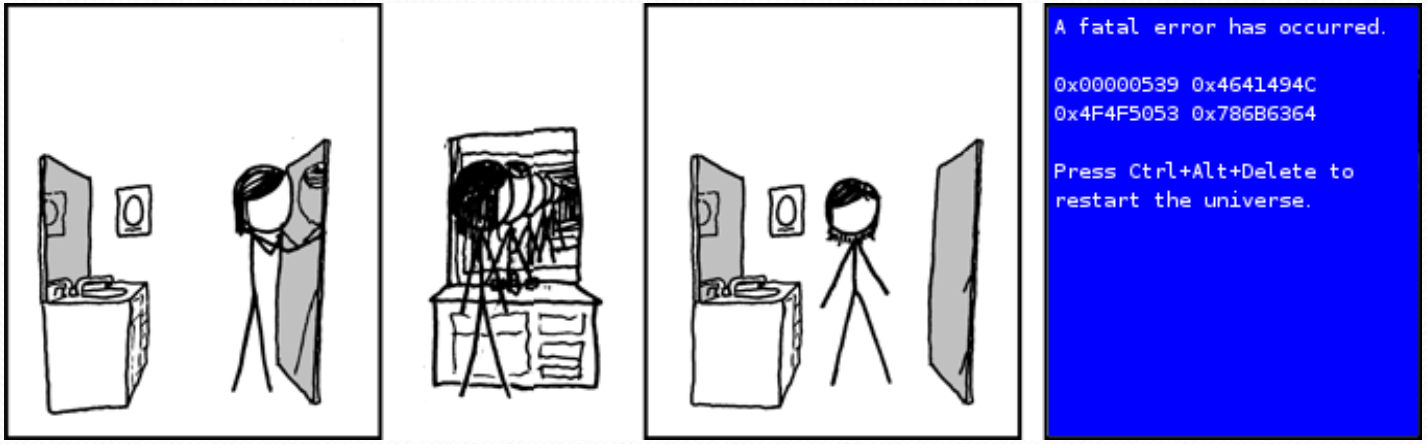
CSE 143

Lecture 14: testing



JUnit and JGrasp

- To add JUnit to a JGrasp project:
  - Download the JUnit jar file from http://junit.org/ and save it on your computer
  - **Tools → JUnit→ Configure**
  - You will now see a dialog box pop up. In the textfield labeled “JUnit Home” browse for the folder you stored the jar file in.
  - If you have properly configured JGrasp green, red and white Junit buttons will appear.

A JUnit test class

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

public class name {
    ...

    @Test
    public void name() {      // a test case method
        ...
    }
}
```

- A method with `@Test` is flagged as a JUnit test case
  - all `@Test` methods run when JUnit runs your test class

JUnit assertion methods

`assertTrue(`**`test`**`)`	fails if the boolean test is `false`
`assertFalse(`**`test`**`)`	fails if the boolean test is `true`
`assertEquals(`**`expected`**`, `**`actual`**`)`	fails if the values are not the same
`fail()`	immediately causes current test to fail
other assertion methods: `assertNull`, `assertNotNull`, `assertSame`, `assertNotSame`, `assertArrayEquals`	

- The idea: Put assertion calls in your `@Test` methods to check things you expect to be true. If they aren't, the test will fail.
  - Why is there no `pass` method?

- Each method can also be passed a string to show if it fails:
  - e.g. `assertEquals(`**`"message"`**`, `**`expected`**`, `**`actual`**`)`

ArrayIntList JUnit test

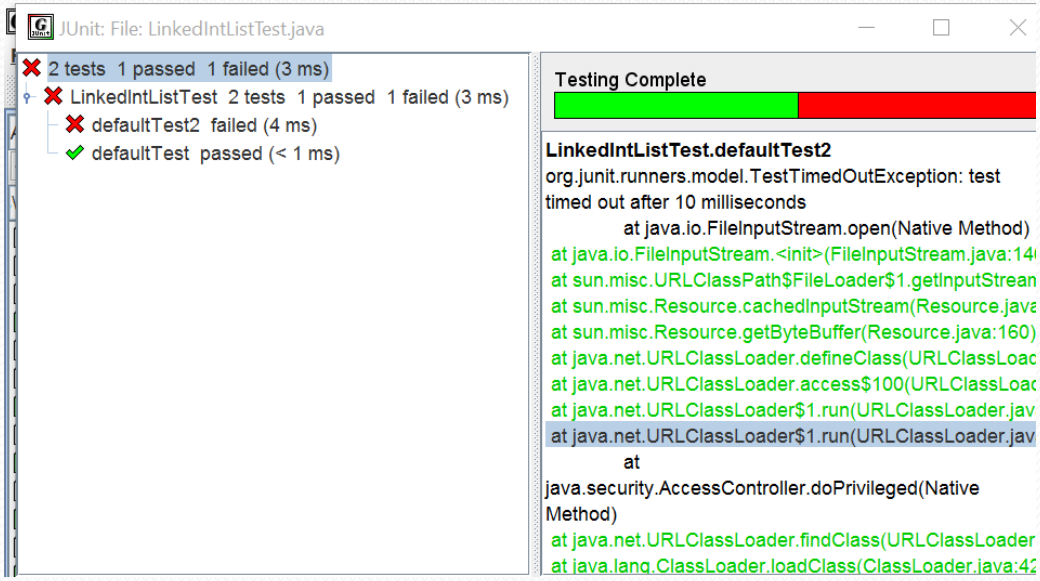
```
import org.junit.*;
import static org.junit.Assert.*;

public class TestArrayIntList {
    @Test
    public void testAddGet1() {
        ArrayIntList list = new ArrayIntList();
        list.add(42);
        list.add(-3);
        list.add(15);
        assertEquals(42, list.get(0));
        assertEquals(-3, list.get(1));
        assertEquals(15, list.get(2));
    }

    @Test
    public void testIsEmpty() {
        ArrayIntList list = new ArrayIntList();
        assertTrue(list.isEmpty());
        list.add(123);
        assertFalse(list.isEmpty());
    }
    ...
}
```

Running a test

- Click the red, white and green plus button to compile the tests. Click the red, green and white running man to run them.
- the JUnit bar will show **green** if all tests pass, **red** if any fail
- the Failure Trace shows which tests failed, if any, and why



Testing for exceptions

```
@Test(expected = ExceptionType.class)
public void name() {
    ...
}
```

- will pass if it *does* throw the given exception, and fail if not
  - use this to test for expected errors

```
@Test(expected = ArrayIndexOutOfBoundsException.class)
public void testBadIndex() {
    ArrayIntList list = new ArrayIntList();
    list.get(4);   // should fail
}
```

Tests with a timeout

```
@Test(timeout = 5000)
public void name() { ... }
```

- The above method will be considered a failure if it doesn't finish running within 5000 ms

```
private static final int TIMEOUT = 2000;
...

@Test(timeout = TIMEOUT)
public void name() { ... }
```

- Times out / fails after 2000 ms

Tips for testing

- You cannot test every possible input, parameter value, etc.
  - So you must think of a limited set of tests likely to expose bugs.

- Think about boundary cases
  - positive; zero; negative numbers
  - right at the edge of an array or collection's size

- Think about empty cases and error cases
  - 0, -1, null; an empty list or array

- test behavior in combination
  - maybe `add` usually works, but fails after you call `remove`
  - make multiple calls; maybe `size` fails the second time only