CS 696 Mobile Phone Application Development
Fall Semester, 2009
Doc 6 Testing
Sept 16, 2009

Testing & Subversion References

JUnit Cookbook http://junit.sourceforge.net/doc/cookbook/cookbook.htm

JUnit Test Infected: Programmers Love Writing Tests http://junit.sourceforge.net/doc/testinfected/testing.htm

JUnit Javadoc: http://www.junit.org/junit/javadoc/3.8/index.htm, http://junit.org/junit/javadoc/4.5/

JUnit FAQ, http://junit.sourceforge.net/doc/faq/faq.htm

Brian Marick's Testing Web Site: http://www.exampler.com/testing-com/

Testing for Programmers, Brian Marick, Available at: http://www.exampler.com/testing-com/writings.html

Android Examples supplied with Android 1.6

Android Documentation, http://developer.android.com/reference/android/test/ActivityTestCase.html

Testing

Johnson's Law

If it is not tested it does not work

The more time between coding and testing

More effort is needed to write tests
More effort is needed to find bugs
Fewer bugs are found
Time is wasted working with buggy code
Development time increases
Quality decreases

Unit Testing

Tests individual code segments

Automated tests

What wrong with:

Using print statements

Writing driver program in main

Writing small sample programs to run code

Running program and testing it be using it

When to Write Tests

First write the tests

Then write the code to be tested

Writing tests first saves time

- Makes you clear of the interface & functionality of the code

- Removes temptation to skip tests

What to Test

Everything that could possibly break

Test values

- Inside valid range
- Outside valid range
- On the boundary between valid/invalid

GUIs are very hard to test

- Keep GUI layer very thin
- Unit test program behind the GUI, not the GUI

Common Things Programs Handle Incorrectly

Adapted with permission from “A Short Catalog of Test Ideas” by Brian Marick,

http://www.testing.com/writings.html

Strings

Empty String

Collections

Empty Collection

Collection with one element

Collection with duplicate elements

Collections with maximum possible size

Numbers

Zero

The smallest number

Just below the smallest number

The largest number

Just above the largest number

XUnit

Free frameworks for Unit testing

SUnit originally written by Kent Beck 1994

JUnit written by Kent Beck & Erich Gamma

Available at: http://www.junit.org/

Ports to many languages at:

http://www.xprogramming.com/software.htm

Sample JUnit 4.x Example

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

public class HelloWorldTest {
	int testValue;
	@Test
	public void testMe() {
		assertEquals(1, testValue);
	}

	@Test
	public void foo() {
		assertTrue(2 == testValue);
	}
```

```
@Test(expected=IndexOutOfBoundsException.class)
   public void testIndexOutOfBoundsException() {
      ArrayList emptyList = new ArrayList();
      Object notValid = emptyList.get(0);
   }

      @Before
      public void initialize(){
            testValue = 1;
      }
}
```

JUnit Example - JUnit 3.x

Goal: Implement a Stack containing integers.

Tests:

Subclass junit.framework.TestCase

Methods starting with 'test" are run by TestRunner

```
import junit.framework.*;
public class  TestStack extends TestCase {

        public void testDefaultConstructor() {
                Stack test = new Stack();
                assertTrue("Default constructor", test.isEmpty() );
        }

        public void testSizeConstructor() {
                Stack test = new Stack(5);
                assertTrue( test.isEmpty() );
        }
}
```

Start of Stack Class

```
public class Stack  {
        int[] elements;
        int topElement = -1;

        public Stack() {
         this(10);
        }

        public Stack(int size) {
                elements = new int[size];
        }

        public boolean isEmpty() {
                return topElement == -1;
        }
}
```

Assert Methods

assertTrue()
assertFalse()
assertEquals()
assertNotEquals()
assertSame()
assertNotSame()
assertNull()
assertNotNull()
fail()

For a complete list see

http://www.junit.org/junit/javadoc/3.8/index.htm/allclasses-frame.html/junit/junit/framework/Assert.html/Assert.html

Testing the Tests

If can be useful to modify the code to break the tests

```
package example;

public class Stack  {
	int[] elements;
	int topElement = -1;

	etc.

	public boolean isEmpty() {
		return topElement == 1;
	}
}
```

Test Fixtures - JUnit 3.x

Before each test setUp() is run
After each test tearDown() is run

```
package example;
import junit.framework.TestCase;

public class  StackTest extends TestCase {
        Stack test;

        public void setUp() {
                test = new Stack(5);
                for (int k = 1; k <=5;k++)
                        test.push( k);
        }

        public void testPushPop() {
                for (int k = 5; k >= 1; k--)
                        assertEquals( "Pop fail on element " + k,  test.pop() , k);
        }
}
```

Testing Exceptions - JUnit 3.x

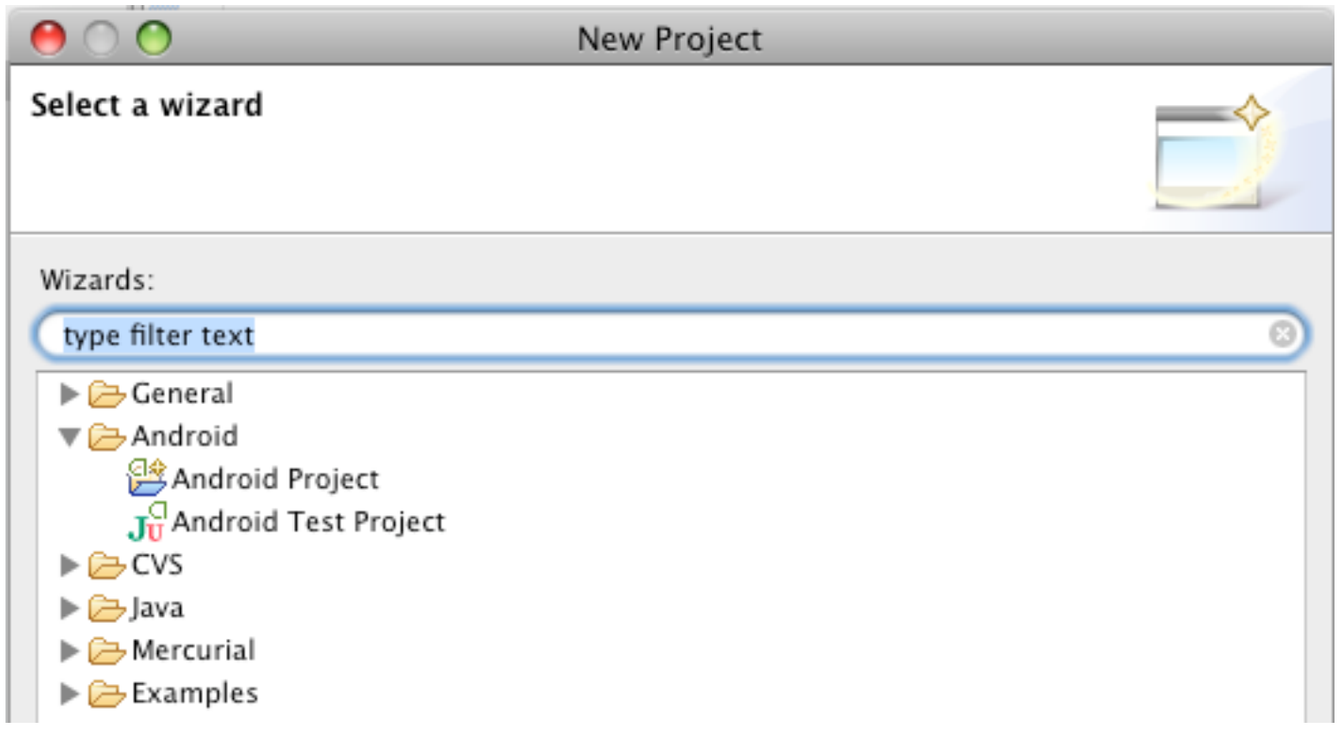
```
public void testIndexOutOfBoundsException() {

    ArrayList list = new ArrayList(10);
    try {
        Object o = list.get(11);
        fail("Should raise an IndexOutOfBoundsException");
    } catch (IndexOutOfBoundsException success) {}
}
```

Example is from the JUnit FAQ

Android Testing

Android JUnit



Main Classes

ActivityUnitTestCase

Unit testing of your Activity

ActivityInstrumentationTestCase2

Functional Testing of activities

DatabaseExampleTest

```
public class DatabaseExampleTest extends
        ActivityInstrumentationTestCase2<DatabaseExample> {

    public DatabaseExampleTest() {
        super("edu.sdsu.cs696", DatabaseExample.class);
    }

    protected void setUp() throws Exception {
        super.setUp();
    }

    public void testTest() {
        assertTrue(2 == 2);
    }
```


testTest is just to show a very simple test. It was the first test I ran to make sure I could run the framework

Simple Test of View Existance

```
public void testPreconditions() {
        Button readButton = getReadButton();
        assertNotNull(getActivity());
        assertNotNull(readButton);
}


 private Button getReadButton() {
        return (Button) getActivity().findViewById(edu.sdsu.cs696.R.id.read);
        }
```


getActivity() is an existing method that returns the activity you are testing.

Testing insert

```
public void testInsert() {
    try {
        Cursor result = getRowsWithName("Cat");
        int rowCount = result.getCount();
        assertTrue(rowCount == 0);
        insert("10", "Cat");
        result = getRowsWithName("Cat");
        rowCount = result.getCount();
        assertTrue(rowCount == 1);
    } finally {
        deleteCatRows();
    }
}
```

getRowsWithName()

```
private Cursor getRowsWithName(String name) {
        Cursor result;
        String[] columns = new String[] { DatabaseHelper.ID,
                        DatabaseHelper.NAME, };
        String[] sqlArguments = new String[] { name };
        result = getActivity().managedQuery(NameProvider.CONTENT_URI, columns,
                        " NAME = ?", sqlArguments, null);
        return result;
}
```

insert

```
private void insert(final String id, final String name) {
    getActivity().runOnUiThread(new Runnable() {
        public void run() {
            EditText idField = getIDEditText();
            EditText nameField = getNameEditText();
            idField.setText((CharSequence) id);
            nameField.setText((CharSequence) name);
            Button insertButton = getInsertButton();
            insertButton.performClick();
        }
    });
    // wait for the request to go through
    getInstrumentation().waitForIdleSync();
  }
}
```

Some Gets

```
private Button getInsertButton() {
      return (Button) getActivity().findViewById(edu.sdsu.cs696.R.id.insert);
}

private EditText getIDEditText() {
      return (EditText) getActivity().findViewById(
                  edu.sdsu.cs696.R.id.databaseId);
}

private EditText getNameEditText() {
      return (EditText) getActivity().findViewById(edu.sdsu.cs696.R.id.name);
}
```

delete

```
private void deleteCatRows() {
      int rowsDeleted = getActivity().getContentResolver().delete(
                  NameProvider.CONTENT_URI, "NAME = 'Cat'", null);
      Log.i("test", "deleted rows after insert " + rowsDeleted);
}
```


The Log.i method is not needed, but logging is one way to help find out what is going on when things don't work

Monkey - Random Testing

Monkey

Generates random events for your activity

Enters text

Click buttons

Selects menus

Rotates screen

etc.

Terse output

```
Al pro 21->adb shell monkey -p edu.sdsu.cs696  500
    // activityResuming(com.android.launcher)
```

First Verbose Level

```
Al pro 22->adb shell monkey -p edu.sdsu.cs696 -v 500
:Monkey: seed=0 count=500
:AllowPackage: edu.sdsu.cs696
:IncludeCategory: android.intent.category.LAUNCHER
:IncludeCategory: android.intent.category.MONKEY
// Event percentages:
//   0: 15.0%
//   1: 10.0%
//   2: 15.0%
//   3: 25.0%
//   4: 15.0%
etc...
:Sending Pointer ACTION_UP x=304.0 y=6.0
:Sending Pointer ACTION_MOVE x=1.0 y=3.0
:Sending Pointer ACTION_MOVE x=-2.0 y=-3.0
:Dropped: keys=0 pointers=0 trackballs=0 flips=0
// Monkey finished
```