CS 580 Client-Server Programming
Fall Semester, 2012
Doc 9 Testing
Sept 25, 2012

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

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

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

Android and JUnit

Android supports JUnit tests

Only supports JUnit 3 type tests

JUnit Example - JUnit 3.x

Goal: Implement a Stack containing integers.

Tests:

- Subclass junit.framework.TestCase
- Methods starting with 'test" are run by TestRunner

Class To Test

```
public class Stack {
	private ArrayList data = new ArrayList();

	public boolean isEmpty() {
		return data.size() == 0;
	}

	public void push(Object value) {
		data.add(value);
	}

	public Object pop() throws EmptyStackException {
		if (isEmpty())
			throw new EmptyStackException();
		Object topOfStack = data.get(data.size() -1);
		data.remove(data.size() -1);
		return topOfStack;
	}
}
```

Sample Testcase

```
public final void testPop() throws EmptyStackException {
        Stack test = new Stack();
        assertTrue( test.isEmpty() );
        test.push("A");
        assertFalse( test.isEmpty() );
        test.push("B");
        test.push("C");
        assertEquals("C", test.pop());
        assertEquals("B", test.pop());
        assertEquals("A", test.pop());
        assertTrue( test.isEmpty() );
        try {
            test.pop();
            fail();
        } catch (EmptyStackException e) {
        }
}
```

Assert Methods

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

For a complete list see

http://junit.sourceforge.net/javadoc/org/junit/Assert.html

setUp & tearDown

setUp()

Called before each test method is run

Create resources common to all tests

tearDown()

Called after each test method is run

Release resources - files

Testing Mars

```
public class MarsServer {
        private Socket server;
        private UpToReader in;
        private Writer out;

        public void connect() throws UnknownHostException, IOException {
                server = new Socket("bismarck.sdsu.edu", 8009);
                in = reader();
                out = writer();
        }

        public UpToReader reader() throws UnsupportedEncodingException, IOException
{
                return new UpToReader( new InputStreamReader( server.getInputStream(),
"UTF8"));
        }

        public Writer writer() throws UnsupportedEncodingException, IOException {
                return new OutputStreamWriter( server.getOutputStream(), "UTF8");
        }
```

All IO uses in, out

```
private void send(String message ) throws IOException {
        if (out == null) {
                connect();
        }
        out.append(message);
        out.flush();
}
```

Subclass for Testing

```
public class MarsServerForTesting extends MarsServer {
        private String fromServer;
        private StringWriter toServer = new StringWriter();;

        public MarsServerForTesting(String serverResponse) {
                fromServer = serverResponse;
        }

        public UpToReader reader() throws UnsupportedEncodingException, IOException {
                return new UpToReader( new StringReader(fromServer));
        }

        public Writer writer() throws UnsupportedEncodingException, IOException {
                return toServer;
        }
}
```


Subclass only changes the Reader "in" and Writer "out" which are created in reader(), writer() methods. So all the logic used in MarsServerForTesting is in the parent class.

Test Reading response

```
    public final void testReadResponse() throws UnknownHostException,
IOException {
        String fromServer = "food:5376.4;weight:6.1;;";
        MarsServerForTesting client = new MarsServerForTesting(fromServer);
        client.connect();
        String response = client.readResponse();
        assertEquals(fromServer, response);
    }
```

Testing Trip without connecting

```
public final void testTrip() throws UnknownHostException, IOException {
        String fromServer = "food:5376.4;weight:6.1;;";
        MarsServerForTesting client = new MarsServerForTesting(fromServer);
        client.connect();
        float[] response = client.trip(1,1,1,1);
        String sentToServer = client.writer().toString();
        String expected =
    "trip;destination:mars;people:1;weight:1.0;mpg:1.0;milesperyear:1.0;;";
        assertEquals(expected, sentToServer);
        assertEquals(5376.4f, response[0], 0.0001f);
        assertEquals(6.1f, response[1], 0.0001f);
}
```

Android Testing

Unit Testing

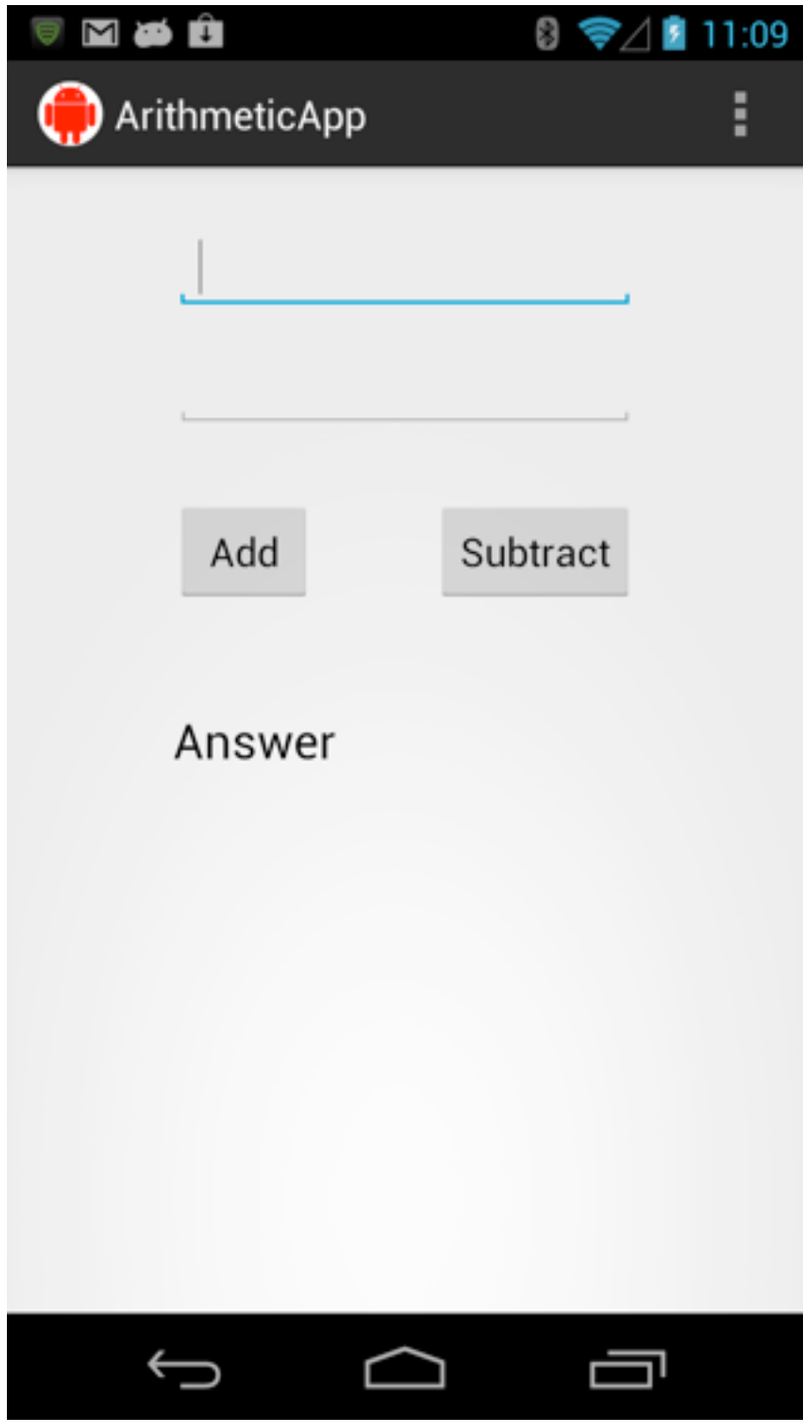
Application logic independent of UI/OS events

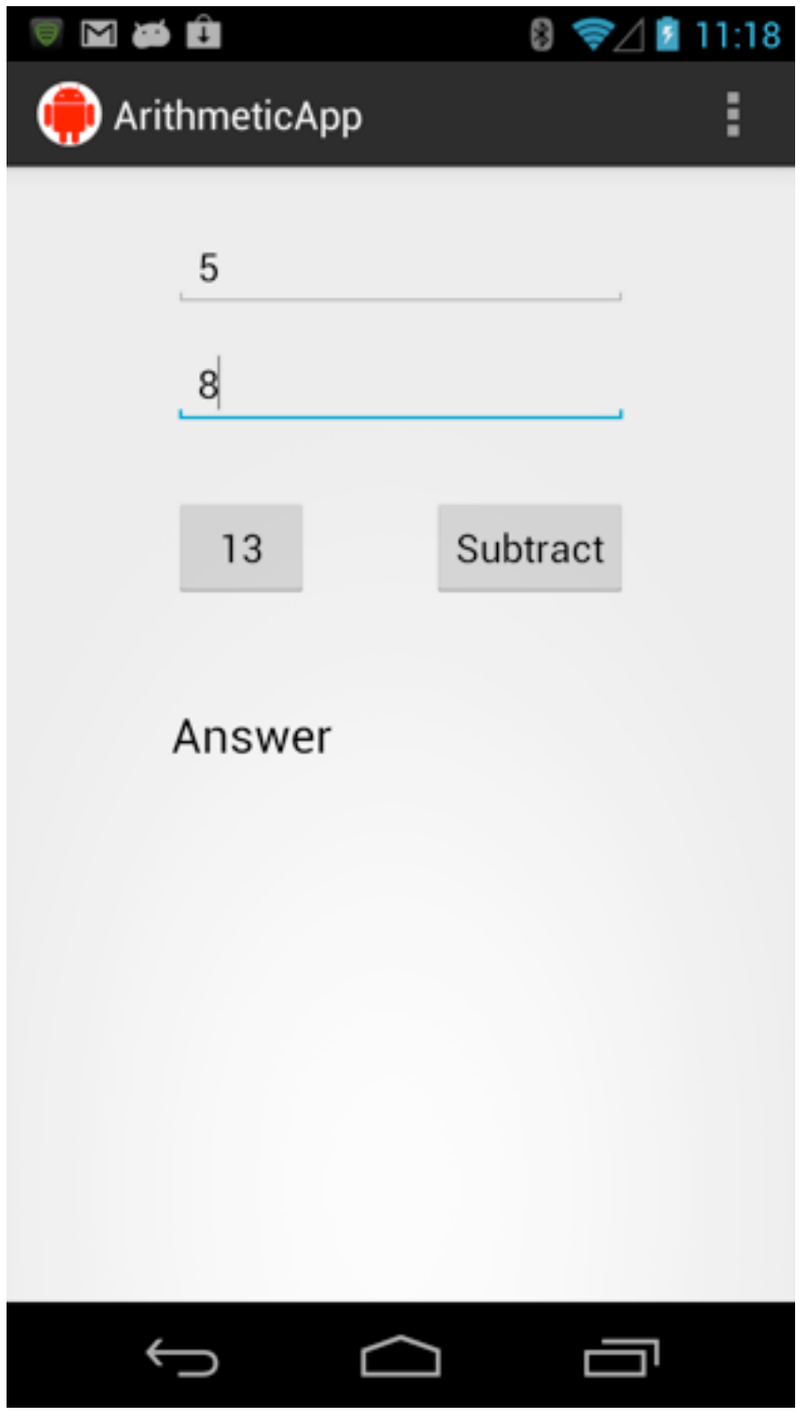
- Normal JUnit tests

Logic dependent on UI/OS events

- Require special environment

Application to test





Application to test

Application to test

```
package edu.sdsu.cs.arithmeticapp;

public class ArithmeticApp extends Activity {
        TextView answer;
        EditText firstOperand;
        EditText secondOperand;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_arithmetic_app);
        answer = (TextView) findViewById(R.id.answer);
        firstOperand = (EditText) findViewById(R.id.topOperand);
        secondOperand = (EditText) findViewById(R.id.bottomOperand);
    }
```

App To Test

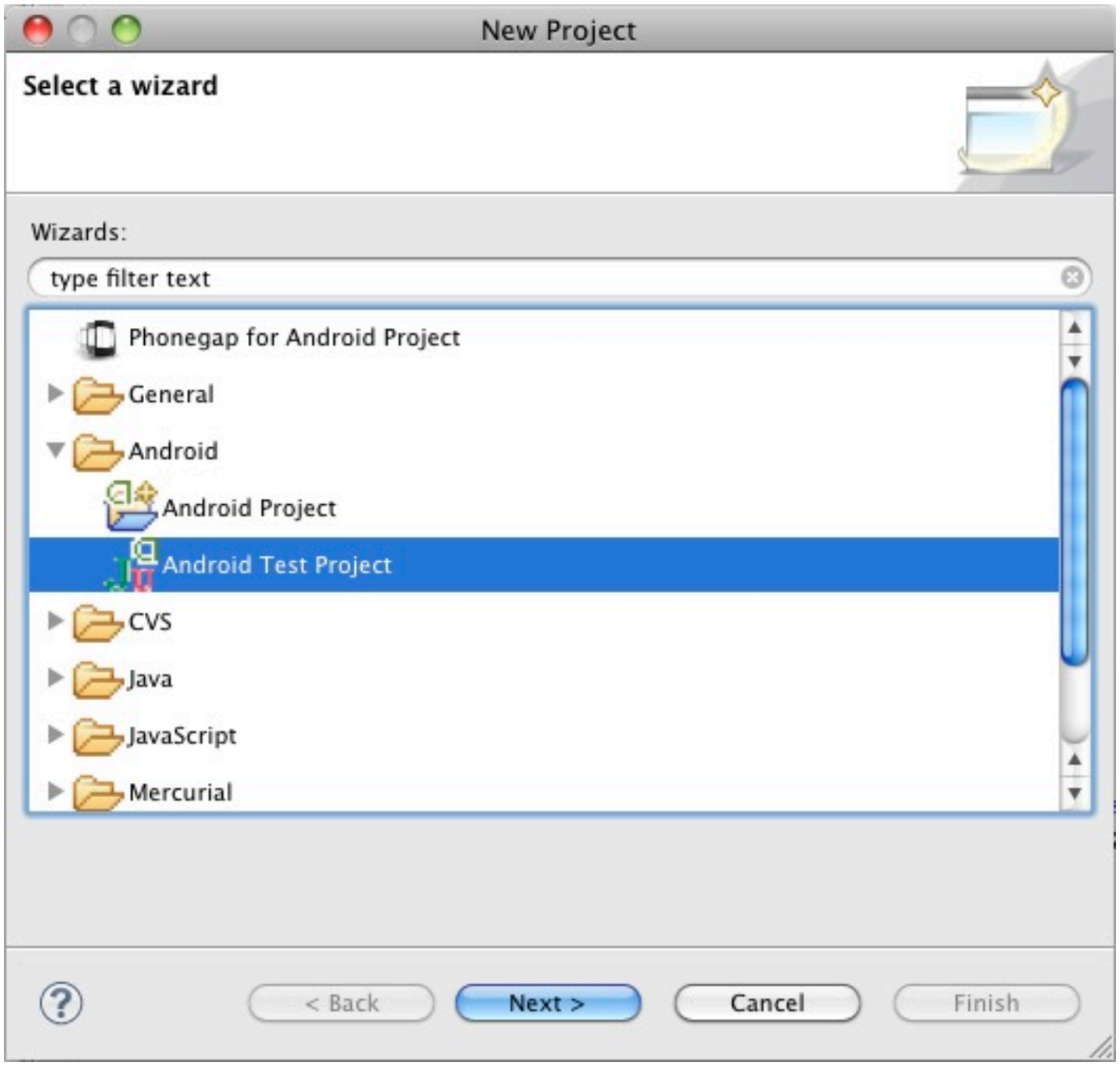
```
public void add(View source) {
    try {
    int sum = intFromField(firstOperand) + intFromField(secondOperand);
    answer.setText("" + sum);
    } catch (Exception error) {
          answer.setText("Invalid input");
    }
 }

  private int intFromField(EditText input) {
        String text = input.getText().toString();
        return Integer.valueOf(text);
  }
```

App To Test

```
    public void subtract(View source) {
       try {
             int difference = intFromField(firstOperand) - intFromField(secondOperand);
             answer.setText("" + difference);
             } catch (Exception error) {
                   answer.setText("Invalid input");
             }
    }
}
```

Test Setup

http://developer.android.com/resources/tutorials/testing/helloandroid_test.html



Start of test

```
package edu.sdsu.cs.arithmeticapp.test;

public class MathUITests extends ActivityInstrumentationTestCase2<ArithmeticApp> {
        private Instrumentation instrumentation;
        ArithmeticApp mathApp;
        Button add;
        Button subtract;
        EditText topOperand;
        EditText bottomOperand;
        TextView answer;

        public MathUITests() {
                super( ArithmeticApp.class);
        }
```

Set up

```
    protected void setUp() throws Exception {
        super.setUp();
        setActivityInitialTouchMode(true);
        instrumentation = getInstrumentation();
        mathApp = getActivity();
        add = (Button)mathApp.findViewById(edu.sdsu.cs.arithmeticapp.R.id.add);
        subtract =
(Button)mathApp.findViewById(edu.sdsu.cs.arithmeticapp.R.id.subtract);
        topOperand =
            (EditText)mathApp.findViewById(edu.sdsu.cs.arithmeticapp.R.id.topOperand);
        bottomOperand =

        (EditText)mathApp.findViewById(edu.sdsu.cs.arithmeticapp.R.id.bottomOperand);
        answer =
(TextView)mathApp.findViewById(edu.sdsu.cs.arithmeticapp.R.id.answer);
    }

    protected void tearDown() throws Exception {
        super.tearDown();
    }
```

Test Setup

```
public void testPreconditions() {
        assertNotNull(instrumentation);
        assertNotNull(mathApp);
        assertNotNull(add);
        assertNotNull(subtract);
        assertNotNull(topOperand);
        assertNotNull(bottomOperand);
        assertNotNull(answer);
        assertEquals("Answer", answer.getText().toString());
        assertEquals("", topOperand.getText().toString());
        assertEquals("", bottomOperand.getText().toString());
    }
```

What all test code can run on UIThread

```
@UiThreadTest
public final void testAdd() {
        topOperand.setText("2");
        bottomOperand.setText("3");
        add.performClick();
        assertEquals("5", answer.getText().toString());
}
```

When some code can't run on UI thread

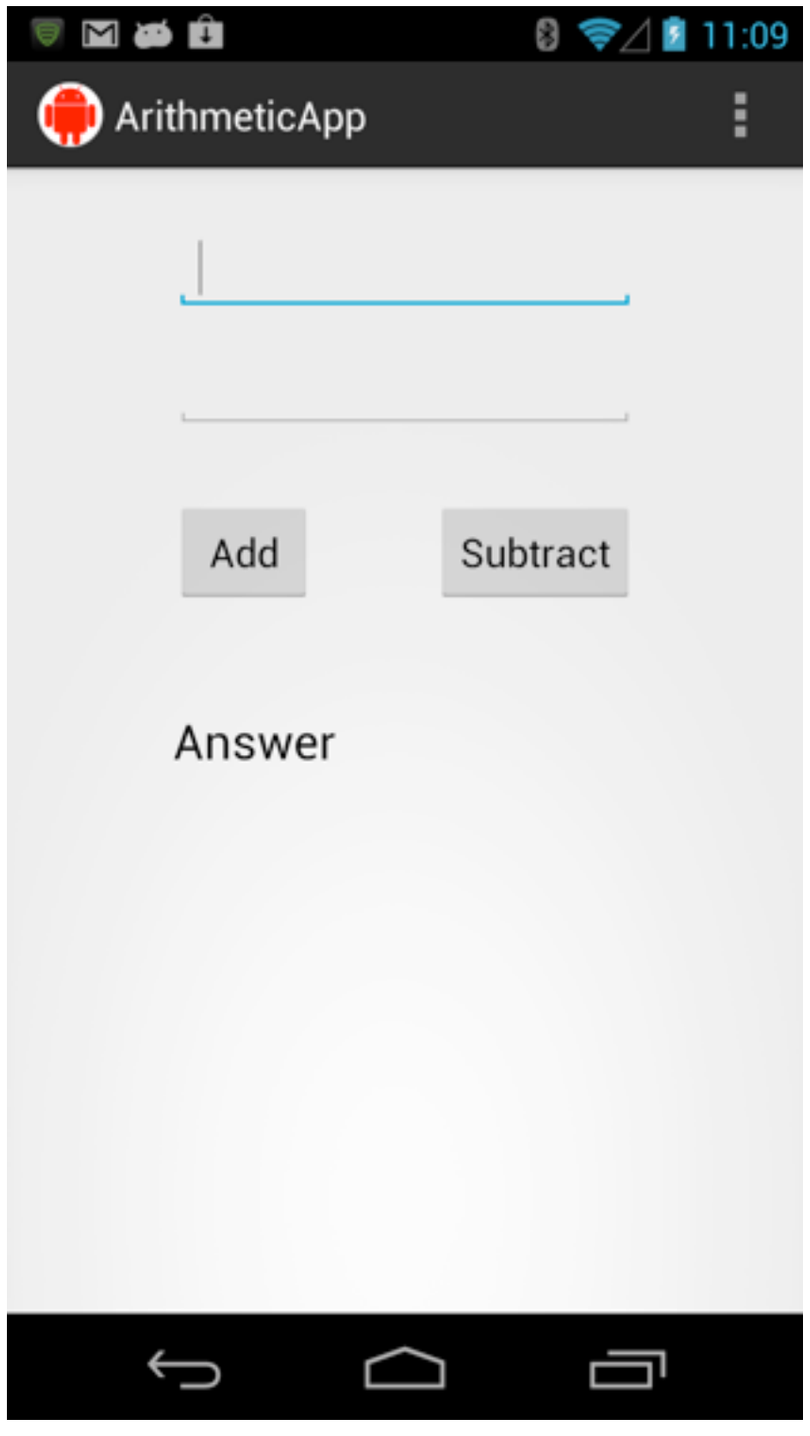
```
public final void testSubstract() {
        mathApp.runOnUiThread(new Runnable() {
                public void run() {
                        topOperand.setText("3");
                        bottomOperand.setText("1");
                        add.performClick();
                }
        });
        instrumentation.waitForIdleSync();
        assertEquals("2", answer.getText().toString());
}
```

To learn more

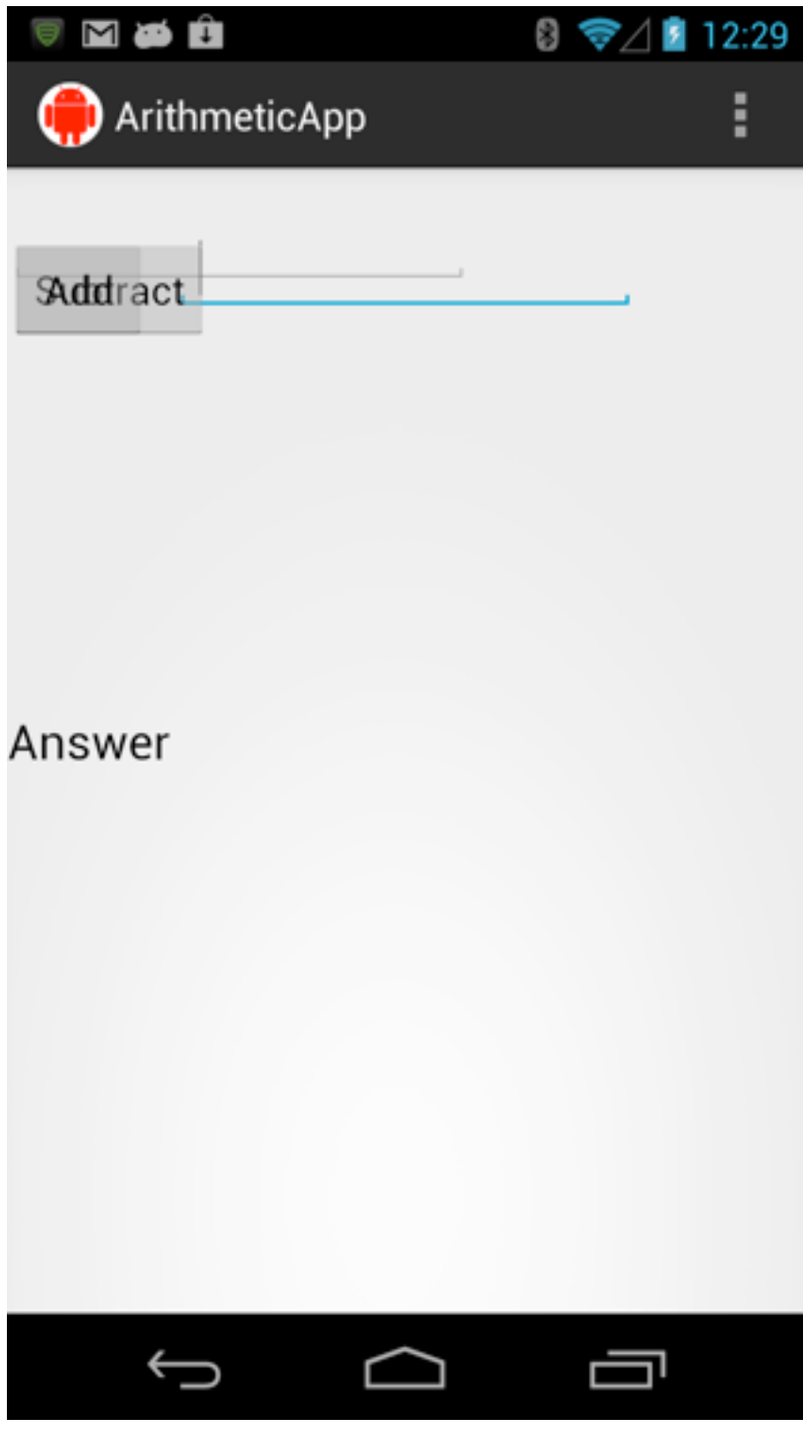
See Android testing Tutorial at:

http://developer.android.com/tools/testing/activity_test.html

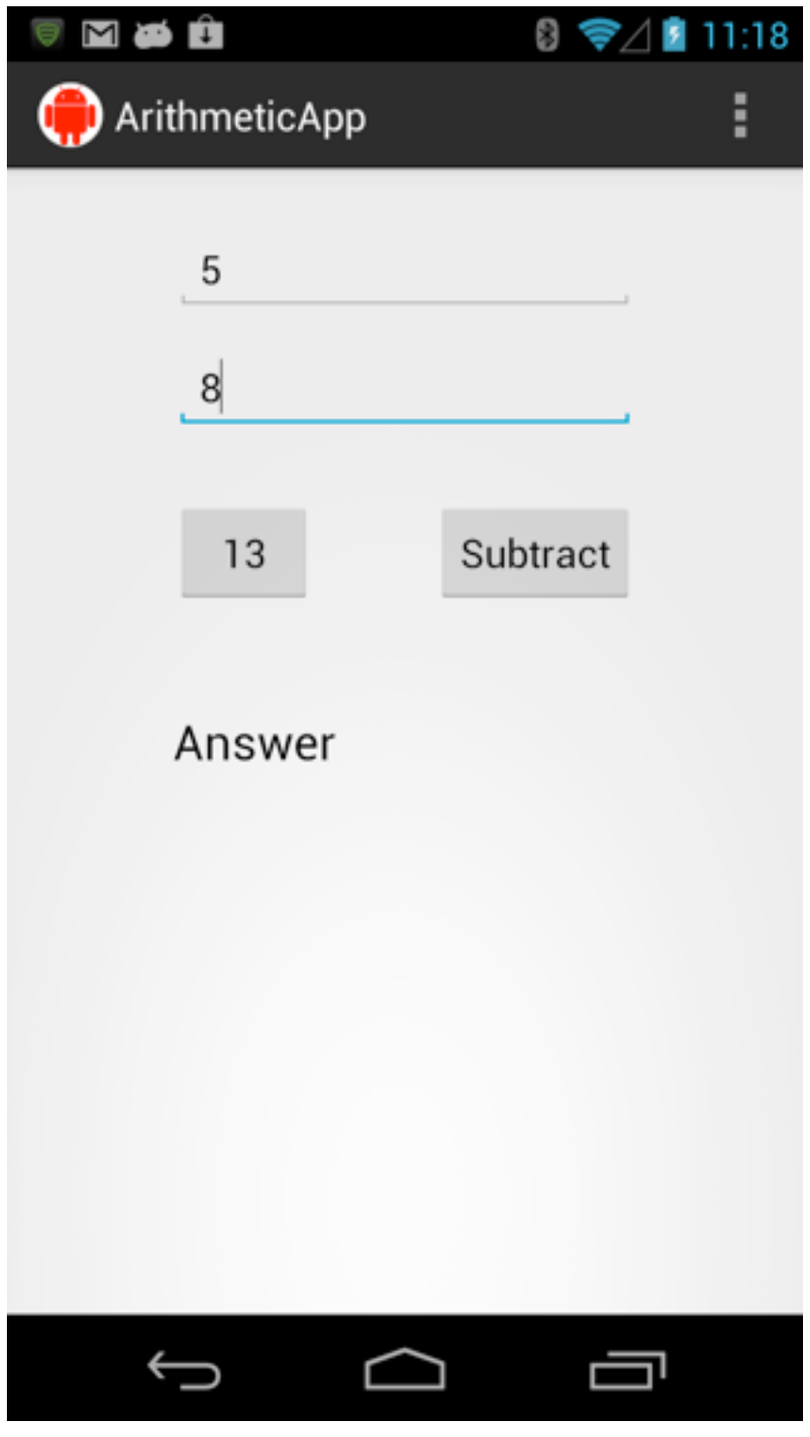
Layout Warning - I started with



I changed the id of two fields and got



Fixed the layout and I then got this



Why

Relative layout

- Relative to other widgets
- Uses widget id

R file

- Was not completely rebuild
- So references were not correct
- Clean project to rebuild R file