Programming Handheld Systems

Adam Porter
The Android Development Environment
The Android Platform

A multi-layered software stack for building and running mobile applications
The Android Development Environment

Your workbench for writing Android applications
Today’s Topics

Installing the Android Developer Tools (ADT) Bundle
Using the Eclipse IDE
Using the Android emulator
Debugging Android applications
Other tools
Prerequisites

Supported Operating Systems:
Windows XP, Vista, or Windows 7
Mac OS X 10.5.8 or later (x86 only)
Various Linux distributions

See: http://developer.android.com/sdk
Prerequisites

Make sure you have the Java Development Kit (JDK6) installed

See:

http://www.oracle.com/technetwork/java/javase/downloads
Getting Started

Download & install the Android Developer Tools (ADT) Bundle

See: http://developer.android.com/sdk
ADT Bundle

Latest Android platform
Eclipse + ADT plugin
Latest system image for emulator
Additional development tools
package course.examples.HelloAndroid;

class MainActivity extends Activity {
    Bundle savedInstanceState;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
}
The Android Emulator

Runs virtual devices
package course.examples.HelloAndroid;

public class MainActivity extends Activity {

    // Overriden
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    // Overriden
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
}
```java
package course.examples.HelloAndroid;

import android.os.Bundle;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```
The Android Emulator

Pros

Doesn’t require an actual phone
Hardware is reconfigurable
Changes are non-destructive
The Android Emulator

Cons

Can be very slow

Some features unavailable

E.g., no support for Bluetooth or USB connections

Performance / user experience can be misleading
Advanced Features

Can emulate many different device/user characteristics, such as:

Network speed/latencies
Battery power
Location coordinates
Advanced Features

Emulate incoming phone calls & SMS messages
The Android Emulator can interconnect multiple emulators.
Advanced Features

Many more options

See:
Debugger

Tool for examining the internal state of a running application
Dalvik Debug Monitor Service (DDMS)

General tools for monitoring application behaviors
Dalvik Debug Monitor Service (DDMS)

DDMS includes

File Explorer
Logcat
Traceview
Hierarchyview
Traceview

Graphically display method traces taken from running application
package course.examples.TheAnswer;

import android.app.Activity;

public class TheAnswer extends Activity {
    private static final int[] answers = {42, 19, 8, 16, 30, 3, 28};
    private static final int answer = 42;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        int val = findAnswer();
        String output = (val == answer) ? String.valueOf(answer) : "We may never know...";
        TextView tv = new TextView(this);
        tv.setText("The answer to life, the Universe and Everything is: " + output);
        setContentView(tv);
    }

    private int findAnswer() {
        for (int val : answers) {
            if (val == answer) return val;
        }
        return -1;
    }
}
UI Hierarchy Viewer

Shows the runtime organization of the user interface
package course.example. TheAnswer;
import android.app.Activity;

public class TheAnswer extends Activity {
    private static final int[] answers = {42, 10, 0, 10, 100, 1000};
    private static final int answer = 42;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        int val = findAnswer();
        String output = (val == answer) ? strings.valueOf(answer) : "We may never know...";

        TextView tv = new TextView(this);
        tv.setText("The answer to life, the Universe and Everything is\n" + output);
        setContentView(tv);
    }

    private int findAnswer() {
        for (int val : answers) {
            if (val == answer) return val;
        }
        return -1;
    }
}
Next Time

Application Fundamentals