Introduction to Android Programming

- Development Environments
- Activities and Layouts
- Android Hello World

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Introduction to Android Programming

- Android is a Linux-based, open-source operating system from Google that runs on phones, tablets and other devices
- Android has gone through several releases including Froyo (2.2), Gingerbread (2.3), Honeycomb (3.x), Ice Cream Sandwich (4.0), Jelly Bean (4.1 - 4.3) and KitKat (4.4)



Android was acquired by Google in 2005 and as of this writing, has the largest market share for smart phones.

While Android has code names and versions like KitKat (4.4), recent API revision levels are:

```
Android 2.2 - 2.2.3 Froyo (API level 8)
Android 2.3 - 2.3.2 Gingerbread (API level 9)
Android 2.3.3 - 2.3.7 Gingerbread (API level 10)
Android 3.0 Honeycomb (API level 11)
Android 3.1 Honeycomb (API level 12)
Android 3.2 Honeycomb (API level 13)
Android 4.0 - 4.0.2 Ice Cream Sandwich (API level 14)
Android 4.0.3–4.0.4 Ice Cream Sandwich (API level 15)
Android 4.1 Jelly Bean (API level 16)
Android 4.2 Jelly Bean (API level 17)
Android 4.3 Jelly Bean (API level 18)
Android 4.4 KitKat (API level 19)
```

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Android Development is Based on Java

- Most Android applications are written in Java, but other languages are available including Scala, C or C++ (using the Native Development Kit) or C# (using Mono for Android)
- Google provides the **Dalvik** virtual machine and a translator that converts standard Java bytecode to Dalvik executable format





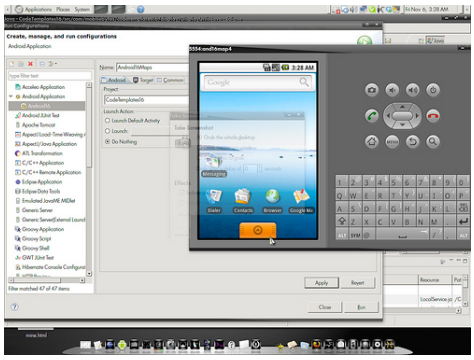
Dalvik, like Android itself, is open-source and named after a town in Iceland.

The dx tool is part of the Android SDK's "platform tools". You normally don't need to run it manually, but instead let your IDE such as Eclipse ADT invoke dx.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Developing Android Applications

- Google provides an Eclipse IDE with the **Android Development Tools** plugin
- You can download an ADT bundle that includes Eclipse and plugins and the Android SDK and other tools
- As of this writing, Google is previewing an alternative IntelliJ-based IDE named **Android Studio**



It's also possible to configure existing Eclipse installations with the ADT plugins. See https://developer.android.com/sdk/index.html for details.

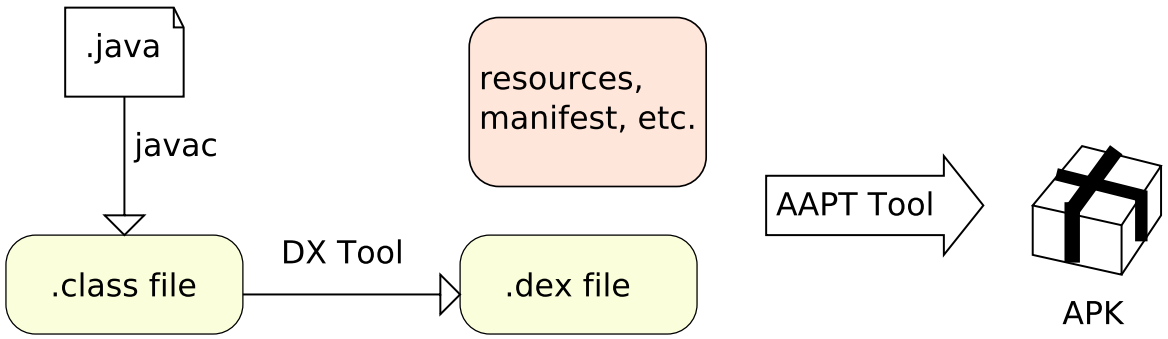
You can also use a plain old text editor and the command line if you so choose, or configure the NetBeans IDE for Android development.

IntelliJ is a popular Java IDE from JetBrains.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Deploying Android Applications

- The final result of developing an app is an **Android Package** (APK) file, which includes the app's Dalvik Executables and resources
- You can test your application on an emulator or deploy to a real device for testing
- Once finished, you can publish to the Google Play store



The Android Asset Packaging Tool (aapt) is part of the Android SDK's platform tools. Like the dx tool, you normally don't need to run it manually, but you can, especially from within a script to build the app.

Note that to test on a real device, the device must be configured to allow installation of APKs from outside of Google Play and to allow debugging.

For publishing to Google play, you first need to digitally sign the APK and follow the steps described at:

`https://developer.android.com/distribute/googleplay/publish/preparing.html`

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Android Hello World

- A simple Android application consists of:
  - A **layout** XML file that defines the user interface
  - An **activity** class that manages the user interface
  - A **menu** XML file that defines the app's menu
  - A **manifest** XML file that configures the app
  - A **strings** XML file that defines user-displayable text

For the application shown in this chapter, we will use generated menu and manifest, so we will not show them here.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

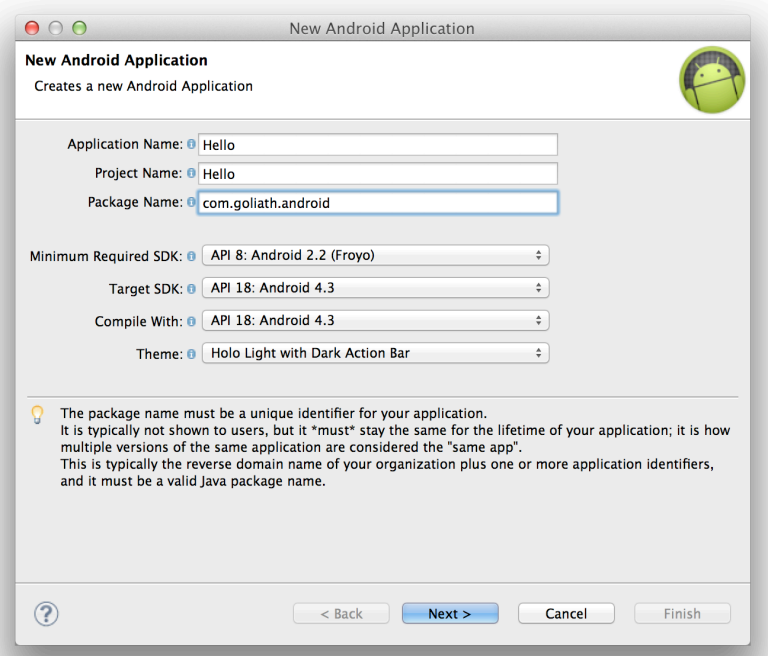
Development Steps for Hello World

- Create a project using an Eclipse wizard
- Edit the layout XML to define the user interface, updating strings.xml as needed
- Edit the activity class to respond to user interaction as needed
- Test on emulator or real device

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Creating an ADT Android Project

- Eclipse provides a wizard to create a new project
- The wizard creates starter files for layout, activity and so forth

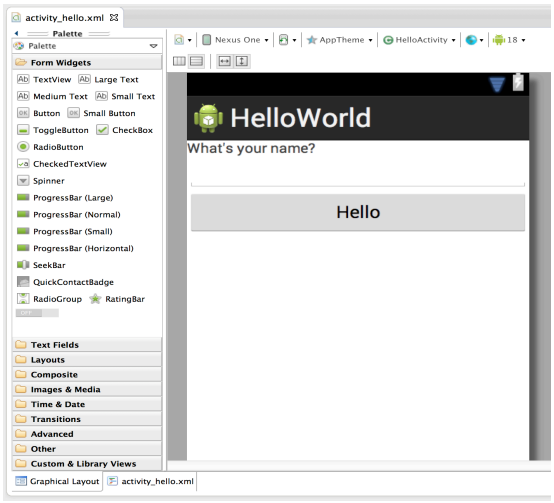


The wizard also creates an empty menu and a minimal AndroidManifest.xml that we will not examine in this chapter.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is a Layout?

- You can define the appearance and user-interface widgets for an application in an XML file
- Eclipse provides both graphical and text views of the layout





activity_hello.xml

As you create and edit widgets, you can define the text for the user interface in the strings.xml resource file.

When the application starts running, it creates objects from the widgets defined in the layout.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

```
1    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2        xmlns:tools="http://schemas.android.com/tools"
3        android:layout_width="match_parent"
4        android:layout_height="match_parent"
5        android:orientation="vertical"
6        tools:context=".HelloActivity" >
7
8        <TextView
9            android:layout_width="wrap_content"
10           android:layout_height="wrap_content"
11           android:text="@string/name_label" />
12
13       <EditText
14           android:id="@+id/name_text"
15           android:layout_width="match_parent"
16           android:layout_height="wrap_content"
17           android:inputType="text" />
18
19       <Button
20           android:id="@+id/hello_button"
21           android:layout_width="match_parent"
22           android:layout_height="wrap_content"
23           android:text="@string/hello_button" />
24
25   </LinearLayout>
```

Strings XML for Hello World

- To facilitate translation to other languages, most Android developers define displayable text in a **strings.xml** resource file

strings.xml

When the application runs, it creates string objects from the strings defined here.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

```
1    <?xml version="1.0" encoding="utf-8"?>
2    <resources>
3
4        <string name="app_name">HelloWorld</string>
5        <string name="action_settings">Settings</string>
6        <string name="name_label">What\'s your name?</string>
7        <string name="hello_button">Hello</string>
8        <string name="hello">Hello, Sam!</string>
9        <string name="who">Who are you?</string>
10
11   </resources>
```

Resource IDs and Generated Classes

- As you save the layout and strings XML, Eclipse generates Java code into the **gen** folder
- Eclipse assigns unique **resource IDs** for each widget and string

R.java

The application can then retrieve widget and string objects using the resource ID.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

```
1    /* AUTO-GENERATED FILE.  DO NOT MODIFY.
2     *
3     * This class was automatically generated by the
4     * aapt tool from the resource data it found.  It
5     * should not be modified by hand.
6     */
7
8    package com.example.helloworld;
9
10   public final class R {
11       public static final class attr {
12       }
13       public static final class dimen {
14           /**  Default screen margins, per the Android Design guidelines.
15
16                Customize dimensions originally defined in res/values/dimens.xml (such as
17                screen margins) for sw720dp devices (e.g. 10" tablets) in landscape here.
18
19                */
20           public static final int activity_horizontal_margin=0x7f040000;
21           public static final int activity_vertical_margin=0x7f040001;
22       }
23       public static final class drawable {
24           public static final int ic_launcher=0x7f020000;
25       }
26       public static final class id {
27           public static final int action_settings=0x7f080002;
28           public static final int hello_button=0x7f080001;
29           public static final int name_text=0x7f080000;
30       }
31       public static final class layout {
32           public static final int activity_hello=0x7f030000;
33       }
34       public static final class menu {
35           public static final int hello=0x7f070000;
36       }
37       public static final class string {
38           public static final int action_settings=0x7f050001;
39           public static final int app_name=0x7f050000;
40           public static final int hello=0x7f050004;
41           public static final int hello_button=0x7f050003;
42           public static final int name_label=0x7f050002;
43           public static final int who=0x7f050005;
44       }
45       public static final class style {
```

```
46          /**
47          Base application theme, dependent on API level. This theme is replaced
48          by AppBaseTheme from res/values-vXX/styles.xml on newer devices.
49
50
51              Theme customizations available in newer API levels can go in
52              res/values-vXX/styles.xml, while customizations related to
53              backward-compatibility can go here.
54
55
56          Base application theme for API 11+. This theme completely replaces
57          AppBaseTheme from res/values/styles.xml on API 11+ devices.
58
59    API 11 theme customizations can go here.
60
61          Base application theme for API 14+. This theme completely replaces
62          AppBaseTheme from BOTH res/values/styles.xml and
63          res/values-v11/styles.xml on API 14+ devices.
64
65    API 14 theme customizations can go here.
66           */
67          public static final int AppBaseTheme=0x7f060000;
68          /**  Application theme.
69    All customizations that are NOT specific to a particular API-level can go here.
70           */
71          public static final int AppTheme=0x7f060001;
72       }
73   }
```

What is an Activity?

- To manage the layout, you write code in an Activity subclass
- In the **onCreate** method, you can retrieve resource IDs and hook up **event handlers** for user-interface widgets such as buttons
- Android provides a **Toast** view that displays timed messages to the user

HelloActivity.java

The wizard provides skeleton methods for onCreate() and onCreateOptionsMenu(). We will not do anything with the options menu in the Hello, World program. The first lines in the onCreate() method were generated by the wizard; they create objects from the XML layout.

It's common to use anonymous inner classes to hook up event handlers for user-interface widgets such as buttons.

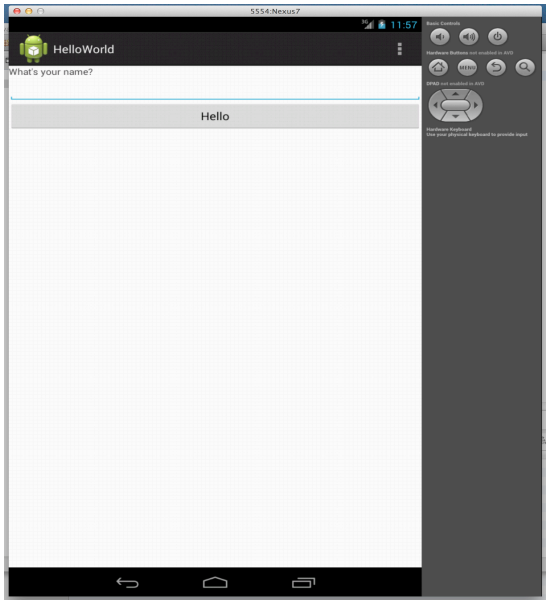
Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

```
1    package com.example.helloworld;
2
3    import android.app.Activity;
4    import android.os.Bundle;
5    import android.view.Menu;
6    import android.view.View;
7    import android.widget.Button;
8    import android.widget.EditText;
9    import android.widget.Toast;
10
11   public class HelloActivity extends Activity
12   {
13       private Button helloButton;
14       private EditText nameEditText;
15
16       @Override
17       protected void onCreate(Bundle savedInstanceState)
18       {
19           super.onCreate(savedInstanceState);
20           setContentView(R.layout.activity_hello);
21           nameEditText = (EditText) findViewById(R.id.name_text);
22
23           helloButton = (Button) findViewById(R.id.hello_button);
24           helloButton.setOnClickListener(new View.OnClickListener()
25           {
26               @Override
27               public void onClick(View v)
28               {
29                   String name = nameEditText.getText().toString();
30                   if (name.equalsIgnoreCase("SAM"))
31                   {
32                       Toast.makeText(HelloActivity.this,
33                               R.string.hello, Toast.LENGTH_SHORT)
34                               .show();
35                   }
36                   else
37                   {
38                       Toast.makeText(HelloActivity.this, R.string.who,
39                               Toast.LENGTH_SHORT).show();
40                   }
41               }
42           });
43
44       }
45
```

```
46      @Override
47      public boolean onCreateOptionsMenu(Menu menu)
48      {
49          // Inflate the menu; this adds items to the action bar if it is present.
50          getMenuInflater().inflate(R.menu.hello, menu);
51          return true;
52      }
53  }
```

Running on an Emulator

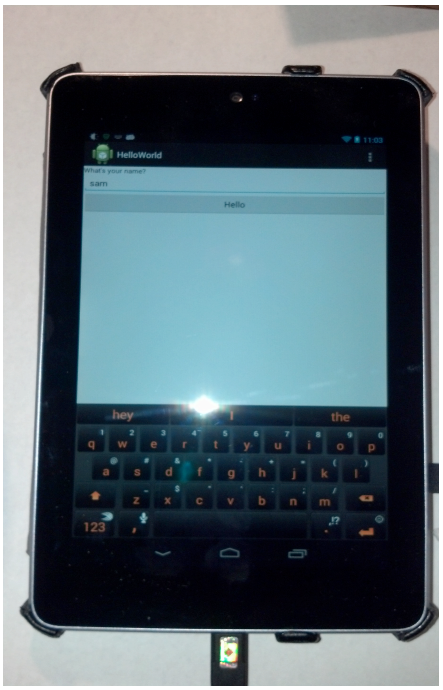
- In Eclipse, you can right-click on the project and run it on an emulator
- Eclipse builds the APK, starts the emulator and runs the app



Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Running on an Actual Device

- The Eclipse ADT can also deploy the APK to a real device



You can also build the APK from the command line using an Ant script and then deploy to a real device either with the Android debugger (adb) or by copying the APK to the device.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Chapter Summary

In this chapter, you learned:

- The basics of Android development using the Eclipse-based ADT
- How to write a simple "Hello, world" application and deploy it

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum