Chapter 2. Structure of Android program

Hanoi University of Science and Technology

Nguyen Hong Quang
19/8/2012

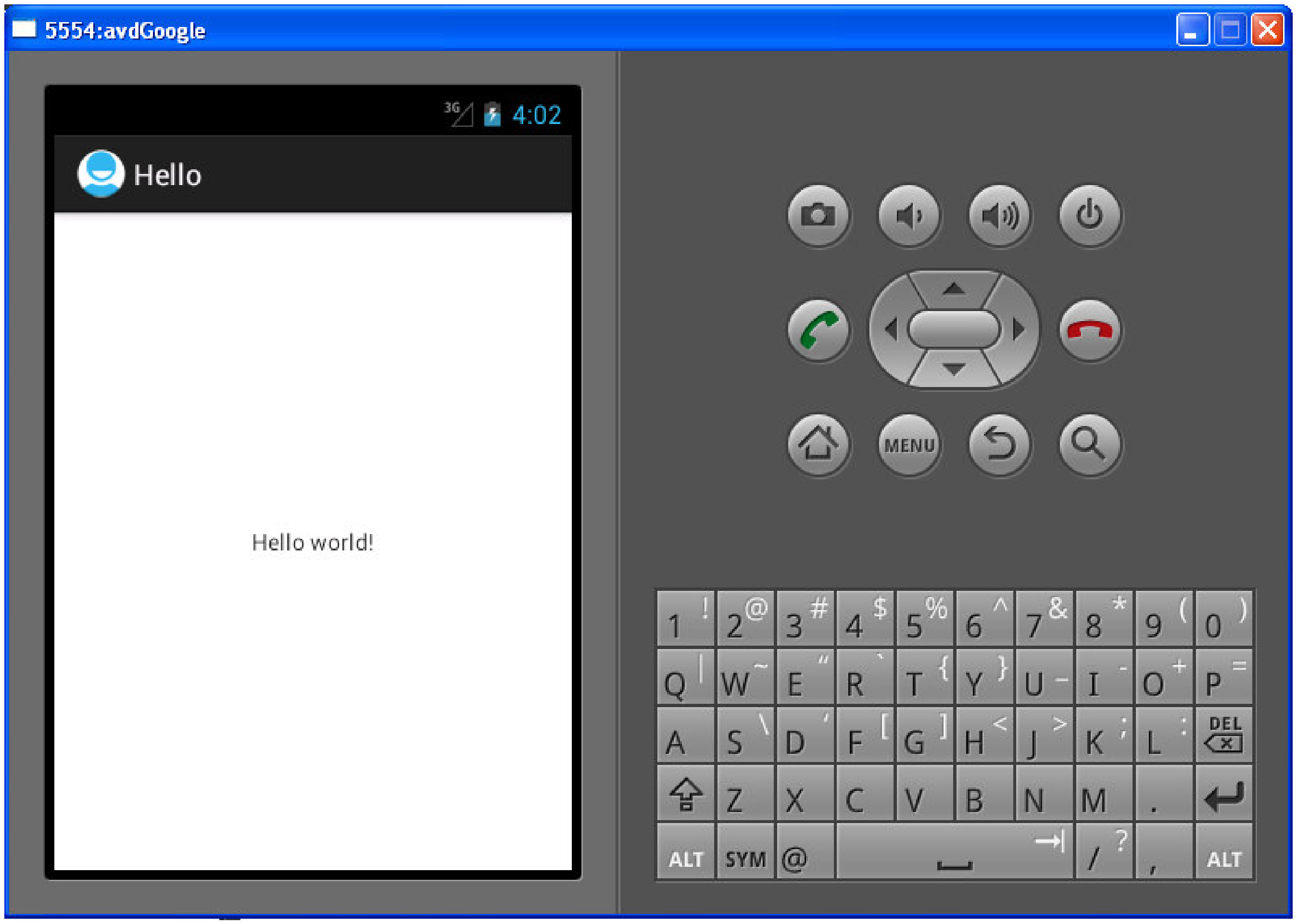
Content

- 2.1. “Hello World” Example
  - Analysis “Hello World”
  - ADT and Log function
- 2.1. Structure of an Android program
  - Analysis AndroidManifest.xml
  - R.java and Resource definition
  - Permissions

Creating "Hello World" Program

- Project name: HelloAndroid
- Build Target: Android 4.1
- Application name: Hello, Android
- Package name: org.example.hello
- Create Activity: Hello
- Min SDK Version: 8 (Android 2.2 – Froyo)

Running on AVD (Android Virtual Devices) Emulator

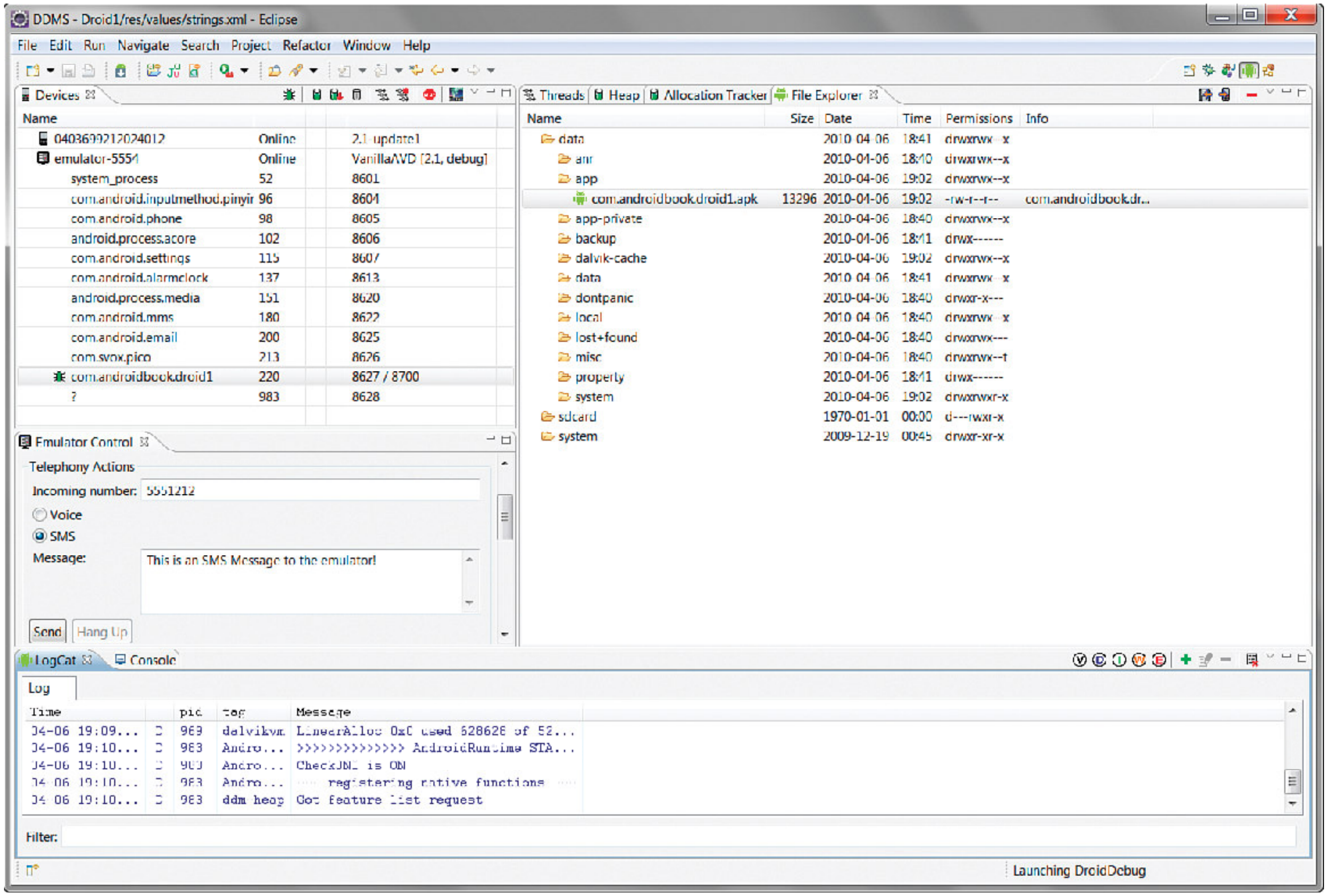


Hello world!

ADT Introduction

- Configure an Android Virtual Device (AVD) for the emulator
- Create a debug configuration for your project
- Build the Android project and launch the debug configuration

Debugging Android Applications Using DDMS



Name		
0403699212024012	Online	2.1-update1
emulator-5554	Online	VanillaAVD [2.1, debug]
system_process	52	8601
com.android.inputmethod.pinyir	96	8604
com.android.phone	98	8605
android.process.acore	102	8606
com.android.settings	115	8607
com.android.alarmclock	137	8613
android.process.media	151	8620
com.android.mms	180	8622
com.android.email	200	8625
com.svox.pico	213	8626
com.androidbook.droid1	220	8627 / 8700
?	983	8628

Name	Size	Date	Time	Permissions	Info
data		2010-04-06	18:41	drwxrwx--x	
anr		2010-04-06	18:40	drwxrwx--x	
app		2010-04-06	19:02	drwxrwx--x	
com.androidbook.droid1.apk	13296	2010-04-06	19:02	-rw-r--r--	com.androidbook.dr...
app-private		2010-04-06	18:40	drwxrwx--x	
backup		2010-04-06	18:41	drwx------	
dalvik-cache		2010-04-06	19:02	drwxrwx--x	
data		2010-04-06	18:41	drwxrwx--x	
dontpanic		2010-04-06	18:40	drwxr-x---	
local		2010-04-06	18:40	drwxrwx--x	
lost+found		2010-04-06	18:40	drwxrwx---	
misc		2010-04-06	18:40	drwxrwx--t	
property		2010-04-06	18:41	drwx------	
system		2010-04-06	19:02	drwxrwxr-x	
sdcard		1970-01-01	00:00	d---rwxr-x	
system		2009-12-19	00:45	drwxr-xr-x	

Telephony Actions

Incoming number: 5551212

Voice [✓ SMS]

Message: This is an SMS Message to the emulator!

```
Time             pid  tag        Message
04-06 19:09...  D  983  dalvikvm   LinearAlloc 0x0 used 628628 of 52...
04-06 19:10...  D  983  Andro...   >>>>>>>>>>>>>> AndroidRuntime STA...
04-06 19:10...  D  983  Andro...   CheckJNI is ON
04-06 19:10...  D  983  Andro...   --- registering native functions ---
04-06 19:10...  D  983  ddm-heap   Got feature list request
```

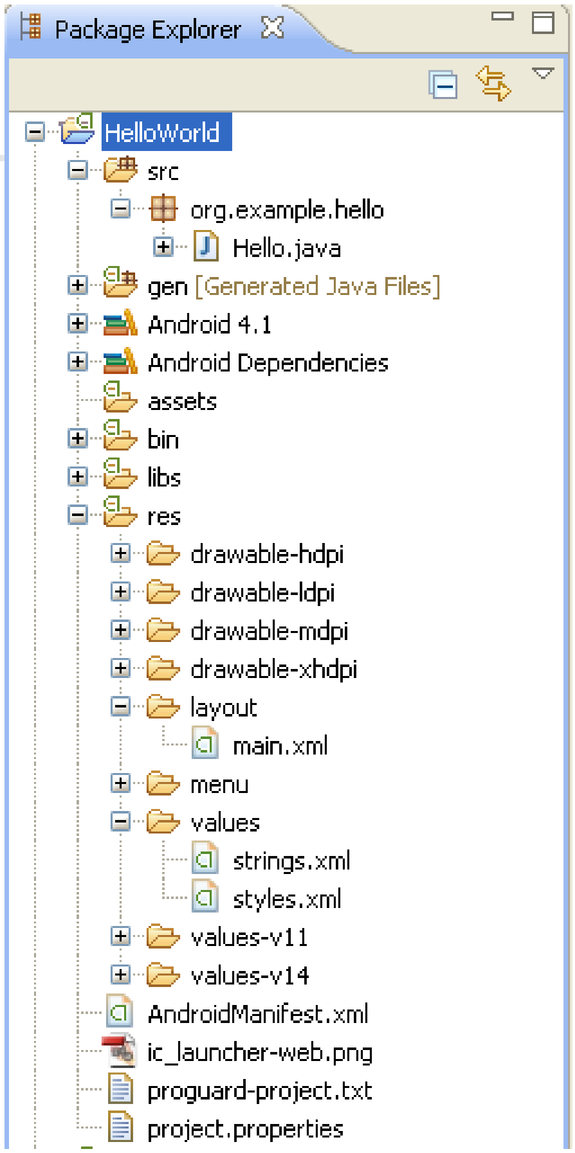
Filter:

Debugging Android Applications Using DDMS

- Task management
  - select individual instances and inspect processes and threads
- File management
  - browse files and directories on the emulator or a device
- Emulator interaction
  - send a number of events, such as simulated calls, SMS messages, and location coordinates, to specific emulator instances
- Screen captures
  - take screenshots of the current screen
- Logging
  - Like “System.out.println”, but more comfortable

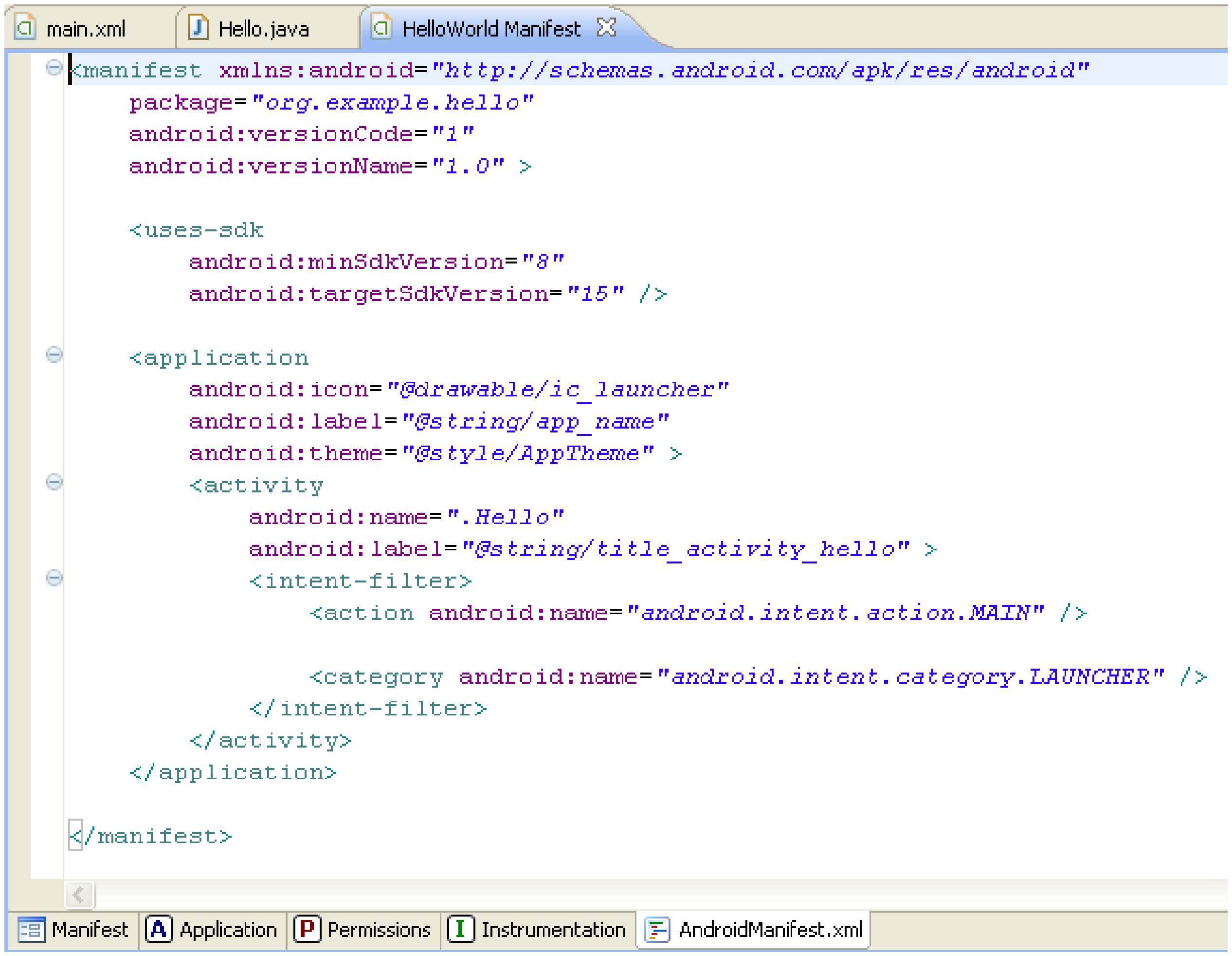
Structure of “HelloWorld” program

- AndroidManifest.xml: where global settings are made.
- Directory:
  - res : resources are held
  - drawable: contains actual image files that application can use and reference.
  - layout : holds an XML file, main.xml, that is referenced by your application when building its interface
  - assets: contains audio files for streaming and animation assets
  - src : contains all the source files



```
HelloWorld
  src
    org.example.hello
      Hello.java
  gen [Generated Java Files]
  Android 4.1
  Android Dependencies
  assets
  bin
  libs
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    drawable-xhdpi
    layout
      main.xml
    menu
    values
      strings.xml
      styles.xml
    values-v11
    values-v14
  AndroidManifest.xml
  ic_launcher-web.png
  proguard-project.txt
  project.properties
```

AndroidManifest.xml Global settings



```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.example.hello"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".Hello"
            android:label="@string/title_activity_hello" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Application manifest file

- The manifest defines the structure and metadata of Android application, its components, and its requirements.
  - uses-sdk: minimum and maximum SDK version
  - uses-configuration: specify each combination of input mechanisms are supported by application.
  - uses-feature: specify which hardware features your application requires (Audio, Bluetooth, Camera, Location, Microphone, NFC, Sensors, Telephony, Touchscreen, USB, Wi-Fi).

Application manifest file

- supports-screens:
  - specify the screen sizes of application
- uses-permission:
  - declare the user permissions your application requires.
- application:
  - specify the metadata for your application (including its title, icon, and theme)
- activity:
  - An activity tag is required for every Activity within Android application.
  - Must include the main launch Activity and any other Activity that may be displayed.

Permissions **Manifest.permission**

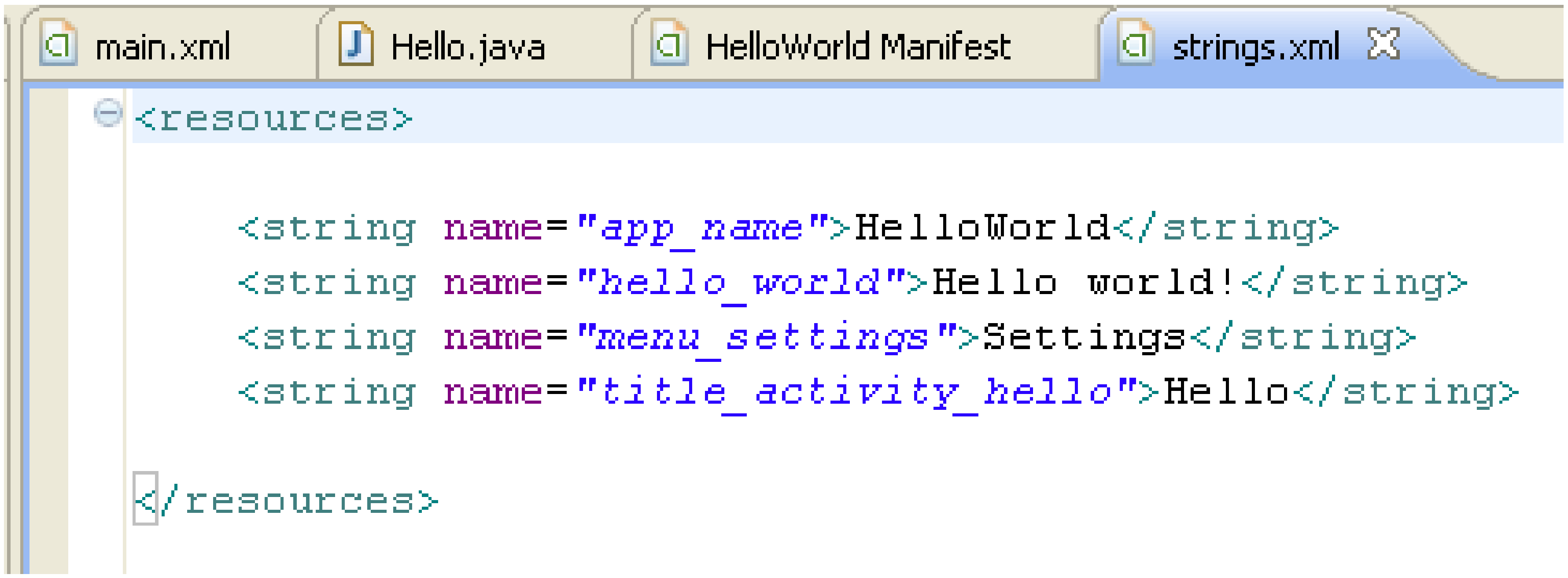
- <uses-permission android:name=”android.permission.ACCESS_FINE_LOCATION”/>
- <uses-permission android:name=”android.permission.BLUETOOTH”/>
- <uses-permission android:name=”android.permission.READ_CONTACTS”/>

EXTERNALIZING RESOURCES

Externalization of resources

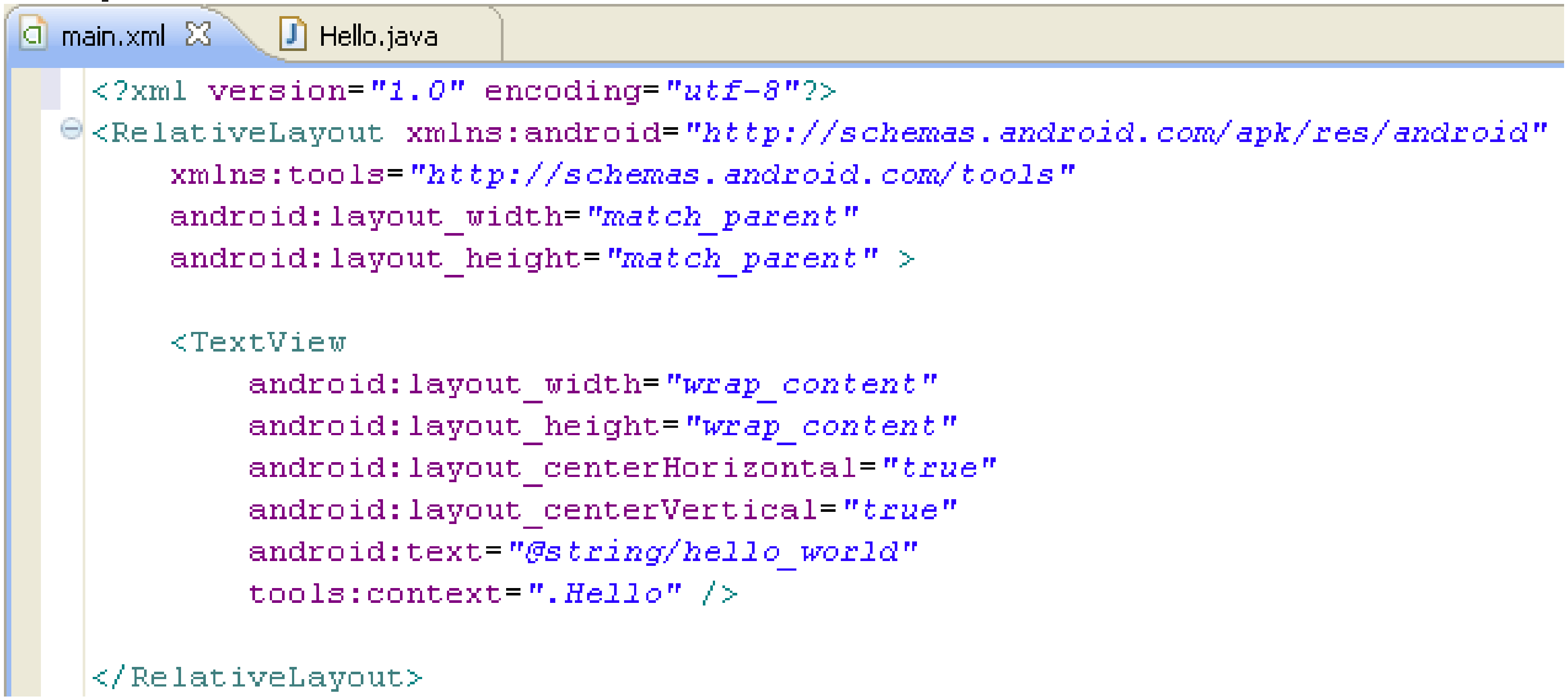
- Ideas:
  - keep non-code resources, such as images and string constants, external to code.
  - ranging from simple values such as strings and colors to more complex resources such as images (Drawables), animations, themes, and menus, layouts
- Advantages:
  - easier to maintain, update, and manage.
  - easily define alternative resource values for internationalization and to include different resources to support variations in hardware — particularly, screen size and resolution

strings.xml



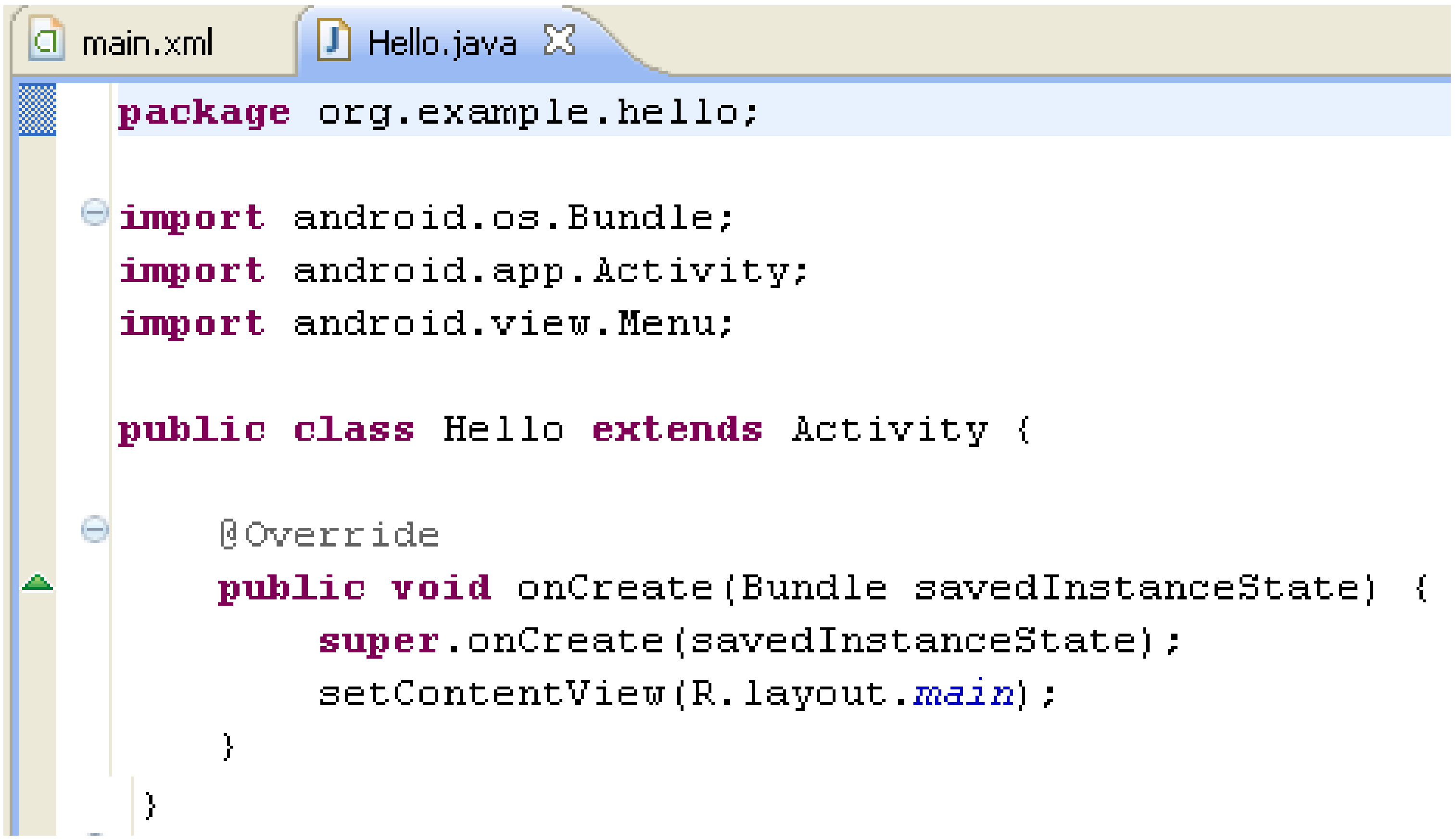
```xml
<resources>

    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_hello">Hello</string>

</resources>
```

main.xml



```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world"
        tools:context=".Hello" />

</RelativeLayout>
```

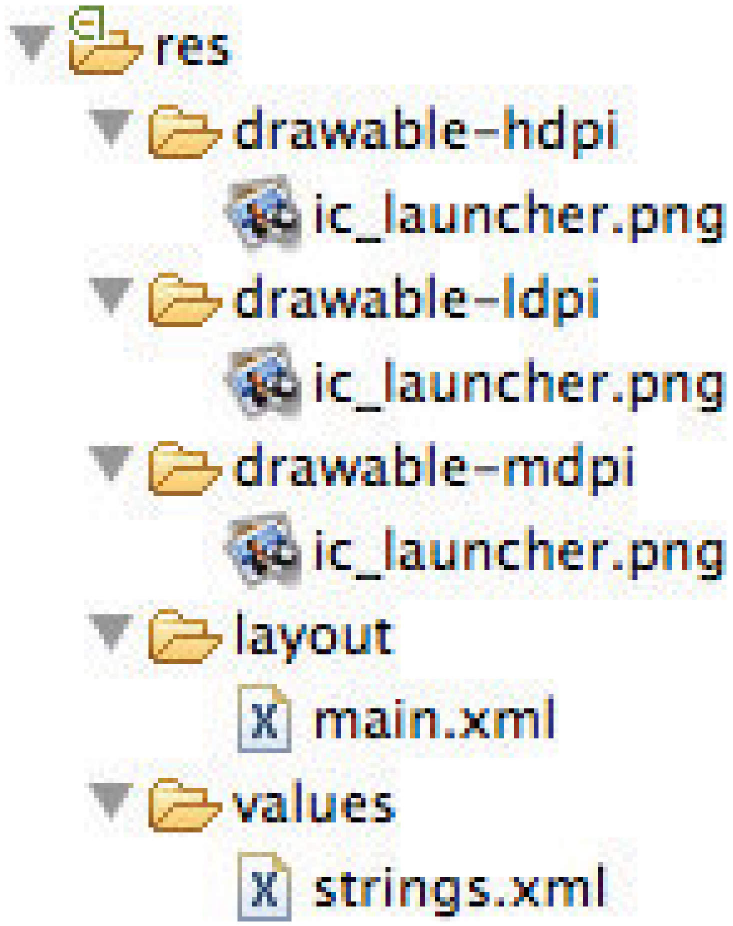
HelloWorld.java



```
package org.example.hello;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class Hello extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Resources

- Application resources are stored under the res folder in your project hierarchy.
- Each of the available resource types is stored in subfolders, grouped by resource type.
- Examples: colors, styles, menus, raw,…



R.java and Resource definition

- R class file that contains references to each of the resources which include in project.
- This enables to reference the resources in code, with the advantage of design-time syntax checking.

Using Resources in Code

- Using the static R class
- The R class contains static subclasses for each of the resource types for which defined at least one resource.
  - For example: the default new project includes the R.string and R.drawable subclasses.
- Each of the subclasses within R exposes its associated resources as variables, with the variable names matching the resource identifiers
  - For example: R.string.app_name or R.drawable.icon.

Resource types

- Simple Values
  - strings, colors, dimensions, styles, and string or integer arrays
  - strings.xml, colors.xml,…
- Styles and Themes
- Drawables
  - bitmaps and NinePatches (stretchable PNG images)

Resource types

- Layouts
  - enable you to decouple your presentation layer from business logic by designing UI layouts in XML rather than constructing them in code
- Animations
- Menus
  - Design menu layouts in XML