ITSMAP F13 Lesson 3

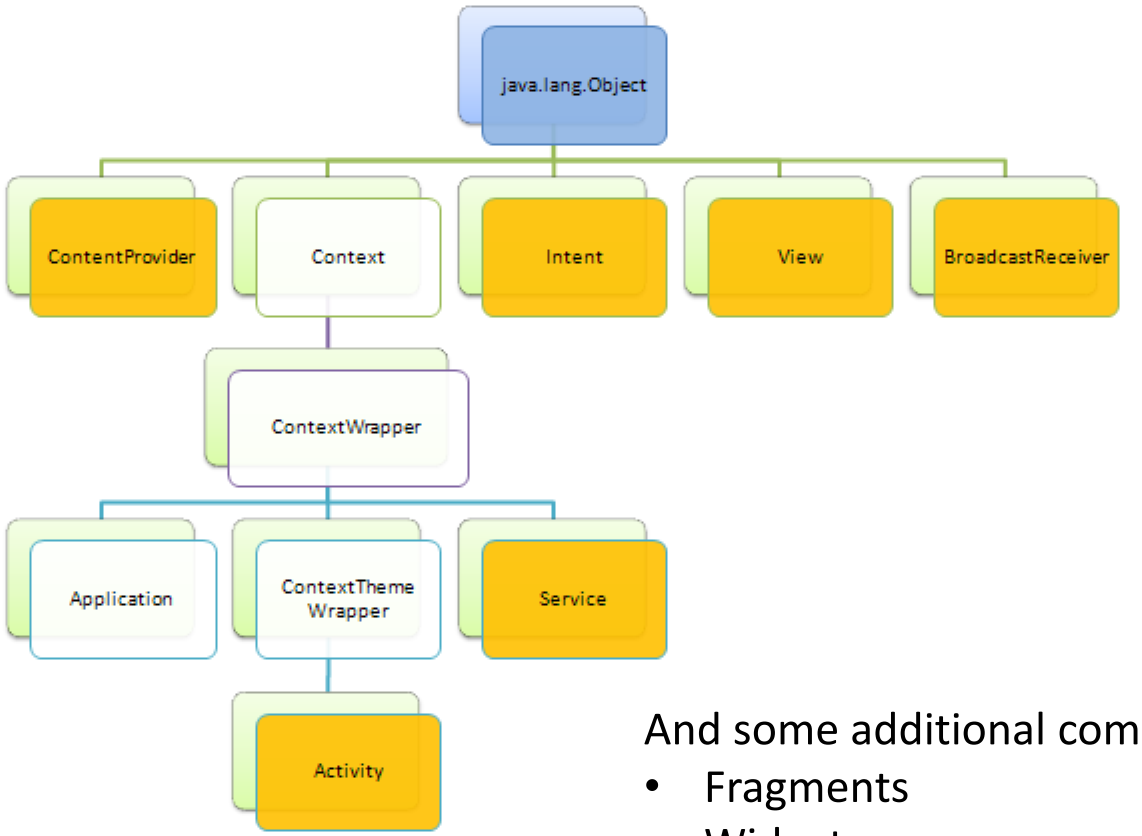
Applications (Apps)
Externalization: Resources
Messaging: Intents

Jesper Rosholm Tørresø

Subjects Lesson 3

1. Activities and resources (Externalization).
   - Using resources in layouts and manifest
     - Example by Region: Language and Icons
   - Retrieving resources at runtime
2. Android Intents
3. Debugging and screencasts

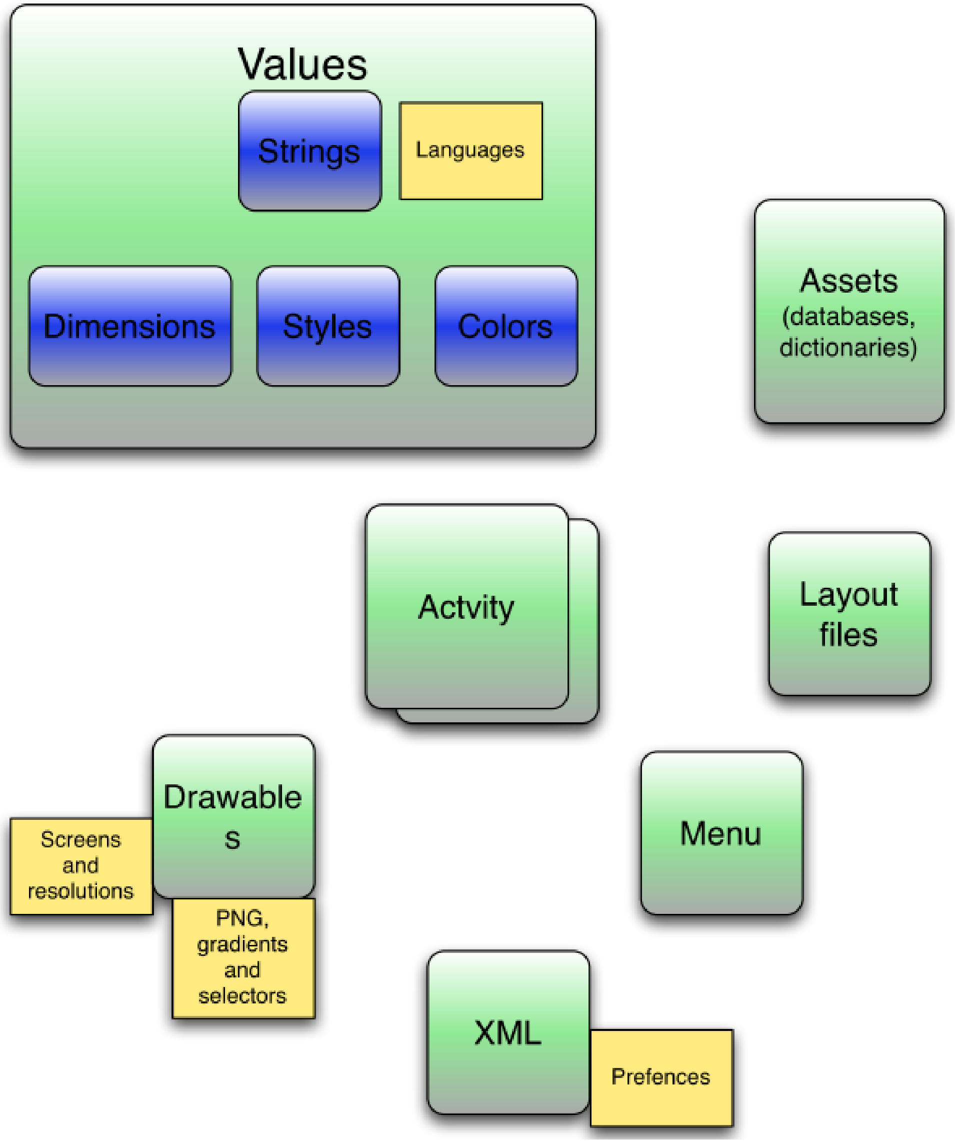
Android application architecture



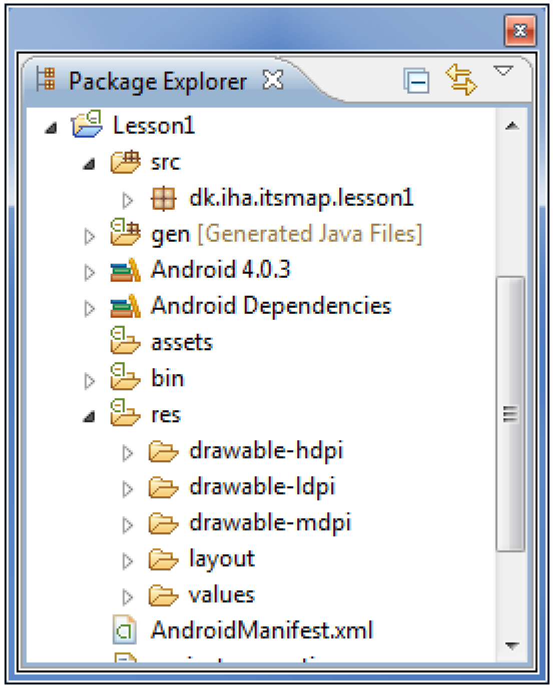
And some additional components

- Fragments
- Widget
- Notifications

Basic Android App Anatomy



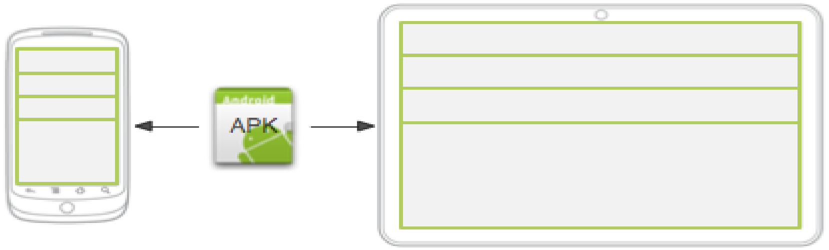
Eclipse



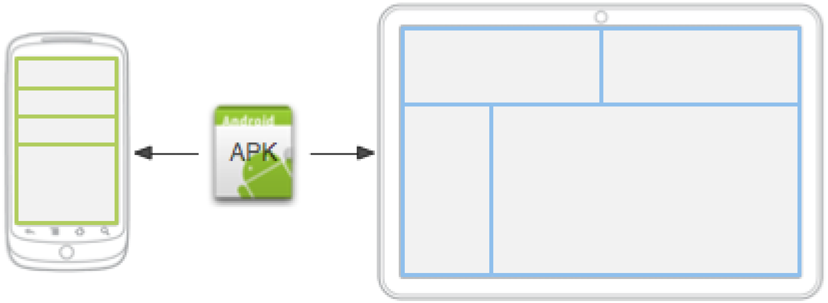
```
Lesson1
  src
    dk.iha.itsmap.lesson1
  gen [Generated Java Files]
  Android 4.0.3
  Android Dependencies
  assets
  bin
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    layout
    values
  AndroidManifest.xml
```

Application resource

- **Default resources** are those that should be used regardless of the **device configuration** or when there are no alternative resources that match the current configuration.



- **Alternative resources** are those that you've designed for use with a **specific configuration**. To specify that a group of resources are for a specific configuration, append an appropriate configuration qualifier to the directory name



http://developer.android.com/guide/topics/resources/index.html
http://developer.android.com/guide/topics/resources/providing-resources.html

Android choose by it self the right external resource according to the handsets current specific configuration like screen specifications, Android version and language settings

Your job is to provide the resources

According to what configuration settings you are focused on!

Android Externalization

- External resources means information/data kept out of your Java code.
  - Nearly all external resources are in XML files
    - Most known is “the layout resource”
  - But images, sounds, database files etc are also included
- All “non coded” resources should be externalized!
- And all non code resources are to a great extend possible to externalized in the Android Framework
- But also possible to put into code (control from code)

Resource Types for UI

- Layouts (a View described in XML)
- Styles and Themes
  - Styles for View
  - Themes for Activity and/or Application
  - More about Styles and Themes later on

All resources
http://developer.android.com/guide/topics/resources/available-resources.html

App Resource Types (Some)

- Strings
  - Is typical an externalization of what would be constant declarations in the App code
- Colors
  - Like Strings, but the numerical value of colors given a name
- Dimension
  - Unit measures

All App resources
http://developer.android.com/guide/topics/resources/available-resources.html

Listing App resource types

- Animation Resources Define pre-determined animations.
  Tween animations are saved in **res/anim/** and accessed from the **R.anim** class.
  Frame animations are saved in **res/drawable/** and accessed from the **R.drawable** class.
- Color State List Resource Define a color resources that changes based on the View state.
  Saved **in res/color/** and accessed from the **R.colo**r class.
- Drawable Resources Define various graphics with bitmaps or XML.
  Saved in **res/drawable/** and accessed from the **R.drawable** class.
- Layout Resource Define the layout for your application UI.
  Saved in **res/layout/** and accessed from the **R.layout** class.
- Menu Resource Define the contents of your application menus.
  Saved in **res/menu/** and accessed from the **R.menu** class.
- String Resources Define strings, string arrays, and plurals (and include string formatting and styling).
  Saved in **res/values/** and accessed from the **R.string**, **R.array**, and **R.plurals** classes.
- Style Resource Define the look and format for UI elements.
  Saved in **res/values/** and accessed from the **R.style** class.

Click links for details

Listing simple App resource types

- Bool XML resource that carries a boolean value.
- Color XML resource that carries a color value (a hexadecimal color).
- Dimension XML resource that carries a dimension value (with a unit of measure).
- ID XML resource that provides a unique identifier for application resources and components.
- Integer XML resource that carries an integer value.
- Integer Array XML resource that provides an array of integers.
- Typed Array XML resource that provides a TypedArray (which you can use for an array of drawables).

Click links for details

How?
Example by Handset Language

- For resource folder name and configuration qualifier convention see

http://developer.android.com/guide/topics/resources/providing-resources.html

- Especially remark the **Table 2** as it lists the valid configuration qualifiers, in order of precedence—if you use multiple qualifiers for a resource directory, **you must add them to the directory name <u>in the order</u> as they are listed in the Table 2.**
- Ex. /res/drawable-da-ldpi
- Ex /res/layout-da-port or /res/layout-da-land.

Localization
(region control)

Create the following directories with strings.xml

- res/values/strings.xml Contains English text for all the strings that the application uses, including text for a string named title.
- res/values-fr/strings.xml Contain French text for all the strings, including title.
- res/values-ja/strings.xml Contain Japanese text for all the strings ***except*** title.

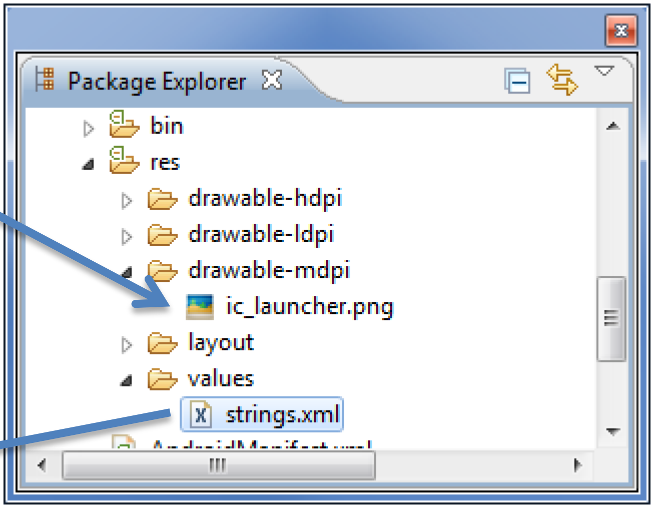
If your Java code refers to `R.string.title;`, here is what will happen at runtime:

- If the device is set to any language other than French, Android will load title from the res/values/strings.xml file.
- If the device is set to French, Android will load title from the res/values-fr/strings.xml file.

Accessing resources

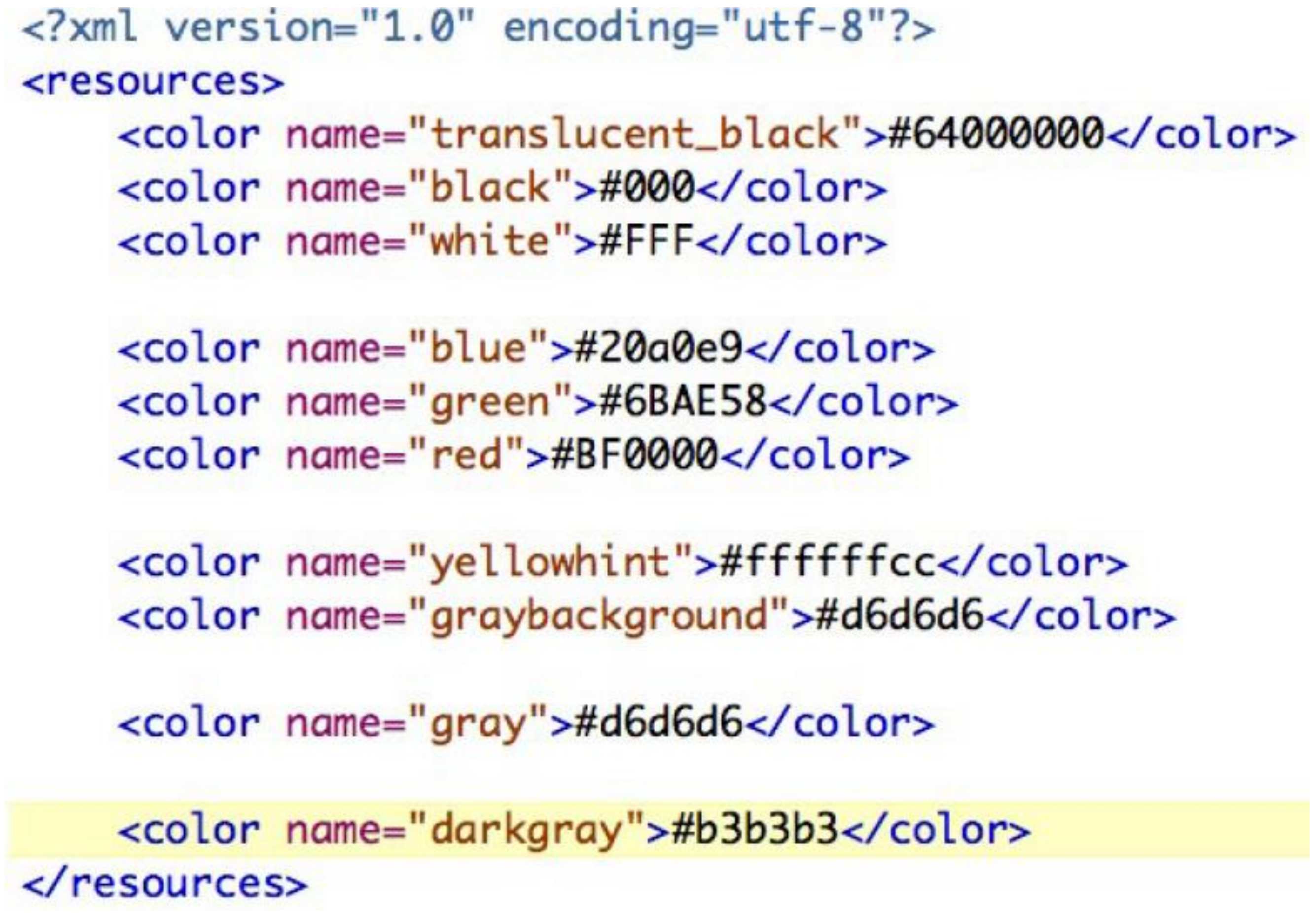
From AndroidManifest.xml

```
……
 <application android:icon="@drawable/ic_launcher"
         android:label="@string/app_name">
        <activity android:label="@string/app_name"
……
```



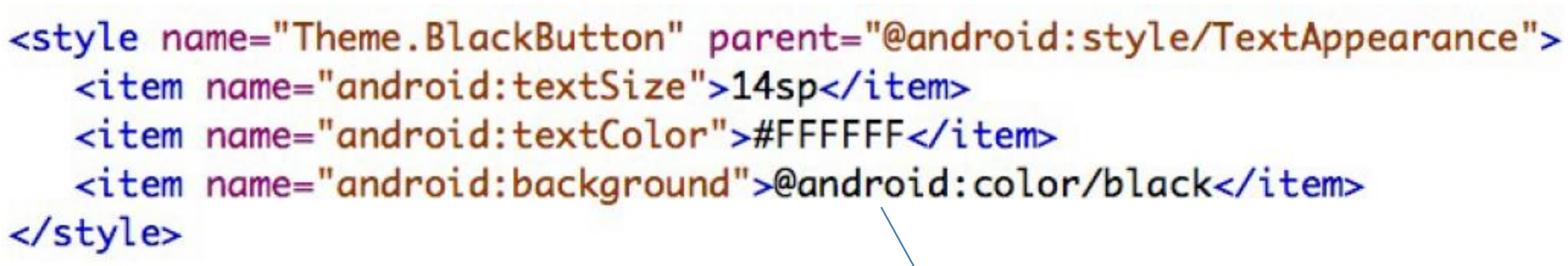
```
?xml version="1.0" encoding="UTF-8" standalone="no"?>
<resources>
   <string name="hello">Hello World</string>
   <string name="app_name">User Application</string>
</resources>
```

XML Resources

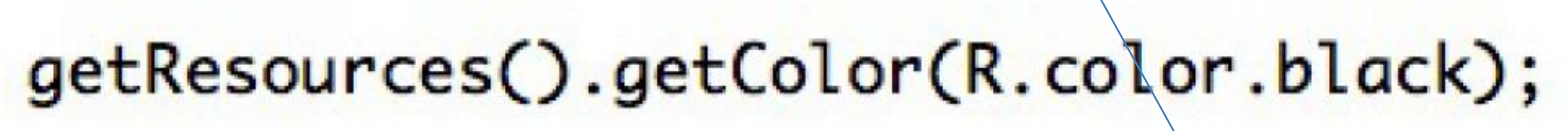


```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="translucent_black">#64000000</color>
    <color name="black">#000</color>
    <color name="white">#FFF</color>

    <color name="blue">#20a0e9</color>
    <color name="green">#6BAE58</color>
    <color name="red">#BF0000</color>

    <color name="yellowhint">#ffffffcc</color>
    <color name="graybackground">#d6d6d6</color>

    <color name="gray">#d6d6d6</color>

    <color name="darkgray">#b3b3b3</color>
</resources>
```

Accessing Resources



```
<style name="Theme.BlackButton" parent="@android:style/TextAppearance">
   <item name="android:textSize">14sp</item>
   <item name="android:textColor">#FFFFFF</item>
   <item name="android:background">@android:color/black</item>
</style>
```



```
getResources().getColor(R.color.black);
```

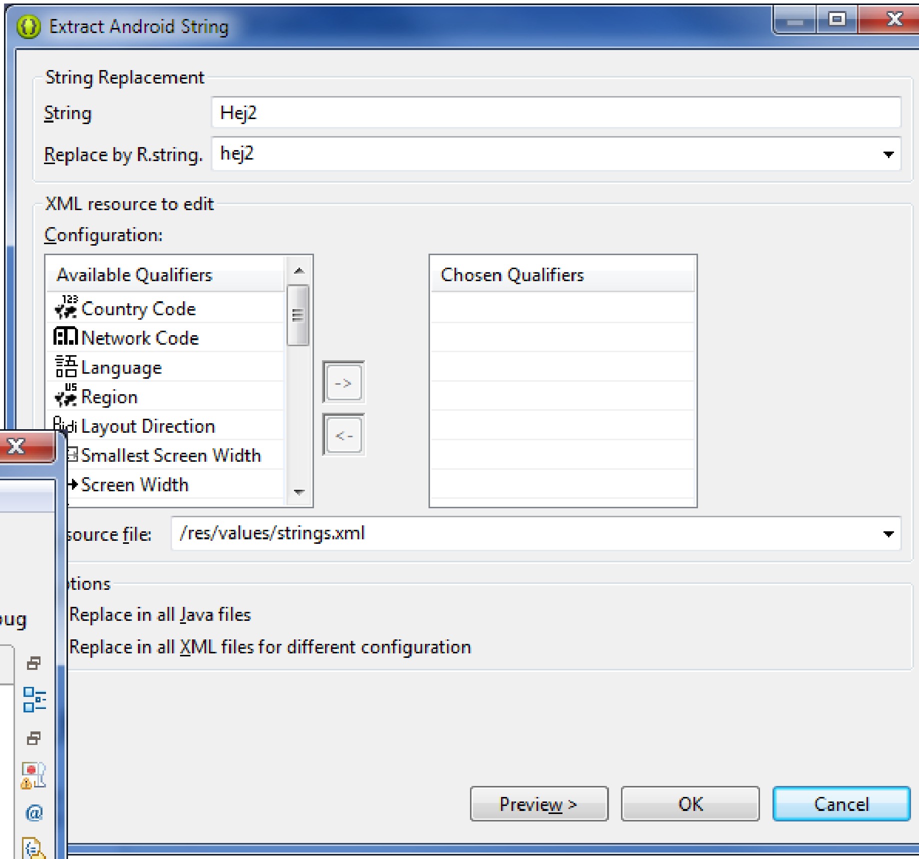
From Android Library

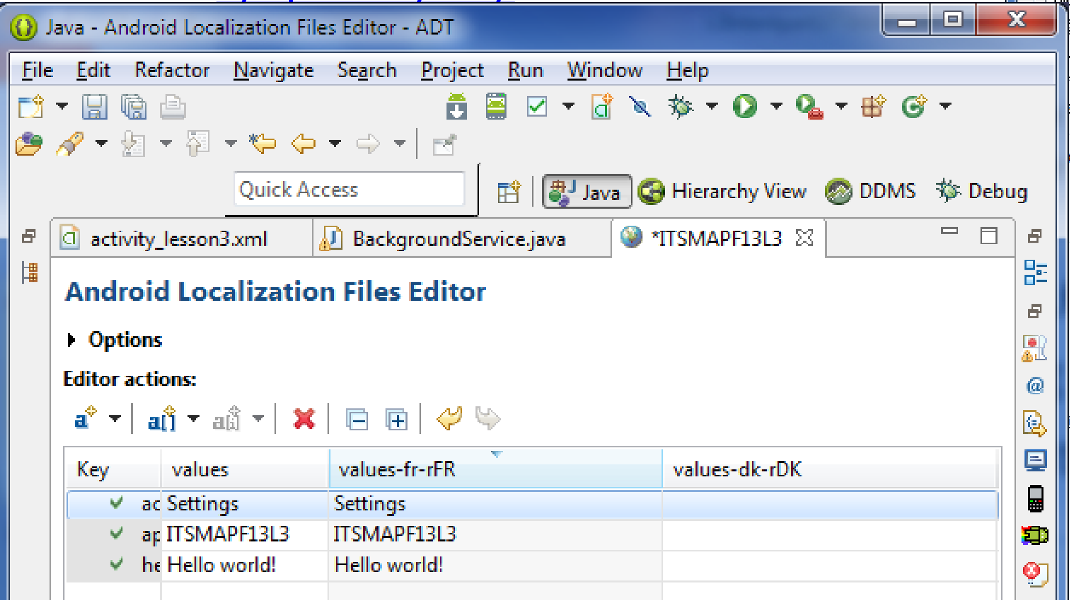
Lesson 3 Exercise 1

- Read string resources
- Create an Italian version of your app !

Hints:

- Use *Eclipse->Refactor->Android->Extract Android String*.
- Install *Sequoyah Android Localization Files Editor* from
  - http://download.eclipse.org/sequoyah/updates/2.1/




Java - Android Localization Files Editor - ADT
File Edit Refactor Navigate Search Project Run Window Help
Quick Access
Java
Hierarchy View
DDMS
Debug
activity_lesson3.xml
BackgroundService.java
*ITSMAPF13L3
Android Localization Files Editor
Options
Editor actions:

Key	values	values-fr-rFR	values-dk-rDK
ac	Settings	Settings	
ap	ITSMAPF13L3	ITSMAPF13L3	
he	Hello world!	Hello world!	

A little about Android Intent and Intent filters

Intent

The Android FW "Messengers"

- Android has a facility for **late run-time binding** between components in the same or different applications, using Intents.
- Intents object holds an abstract description of:
  - An operation to be performed **(start a component)**
  - A broadcast, a description of something that has happened and is being announced. **(sent a messages from a component)**
- Android FW takes care of Intent control

An Intent object is passed to (1)

- Context.startActivity() or Activity.startActivityForResult() to launch an activity or get an existing activity to do something new.
  - It can also be passed to Activity.setResult() to return information to the activity that called startActivityForResult()
- Context.startService() to initiate a service or deliver new instructions to an ongoing service.
  - Similarly, an intent can be passed to Context.bindService() to establish a connection between the calling component and a target service. It can optionally initiate the service if it's not already running.

An Intent object is passed to (2)

- To the broadcast methods
  - Context.sendBroadcast()
  - Context.sendOrderedBroadcast()
  - Context.sendStickyBroadcast()

  and are delivered to all interested broadcast receivers. Many kinds of broadcasts originate in system code.

http://www.vogella.com/articles/AndroidBroadcastReceiver/article.html
Examples using Global and Local Broadcasting.

LocalBroadCastManager

Only broadcasting Intents inside your App Process

Helper to register for and send broadcasts of Intents to local objects within your process. This is has a number of advantages over sending global broadcasts with sendBroadcast(Intent):

- You know that the data you are broadcasting won't leave your app, so don't need to worry about leaking private data.
- It is not possible for other applications to send these broadcasts to your app, so you don't need to worry about having security holes they can exploit.
- It is more efficient than sending a global broadcast through the system.

- http://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html API
- http://www.intertech.com/Blog/Post/Using-LocalBroadcastManager-in-Service-to-Activity-Communications.aspx Example of use

Intent + filters in manifest

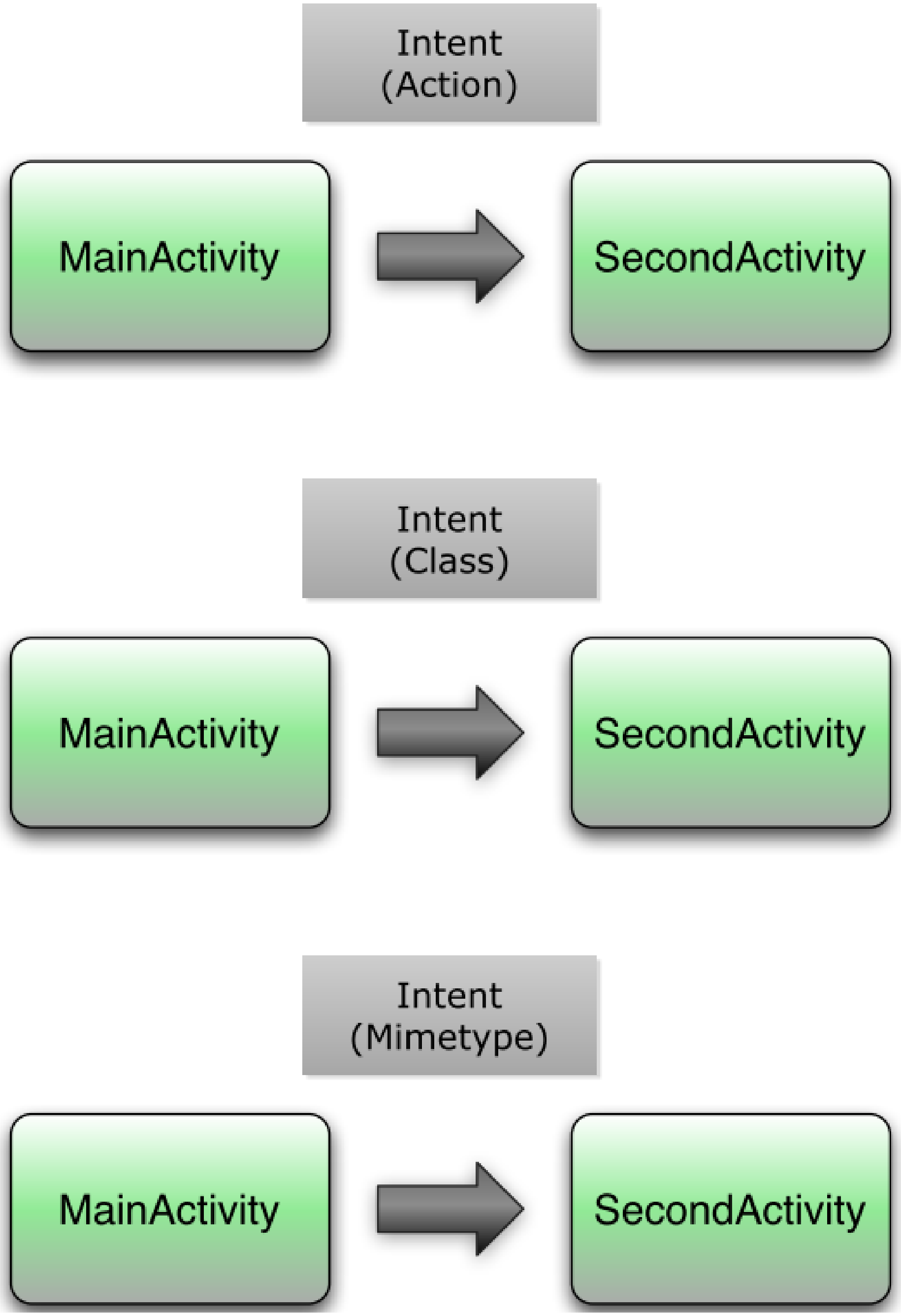
```
<activity android:label="@string/app_name"
        android:name="MainActivity"> <!-- Class name "MainActivty.class" -->
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                 <!-- Description what can be done, there may be
                      several actions to respond/filter on -->
                <category android:name="android.intent.category.LAUNCHER"/>
                 <!- Additional description what to do, there may be several
                     categories to respond/filter on -->
            </intent-filter>
        </activity>
```

- The Android FW uses the intent filters
  - to find the right component to start
  - or to delivers a broadcast
- http://developer.android.com/guide/topics/intents/intents-filters.html

Intent: a little more

Intent may contain a little more information

- Extras: additional named parameters using Keyed values
- Data: URI to data to act on and MIME type of that data.
- Flags: Information to Android FW how to handle and control the Intent


Intent
(Action)
MainActivity
SecondActivity
Intent
(Class)
MainActivity
SecondActivity
Intent
(Mimetype)
MainActivity
SecondActivity

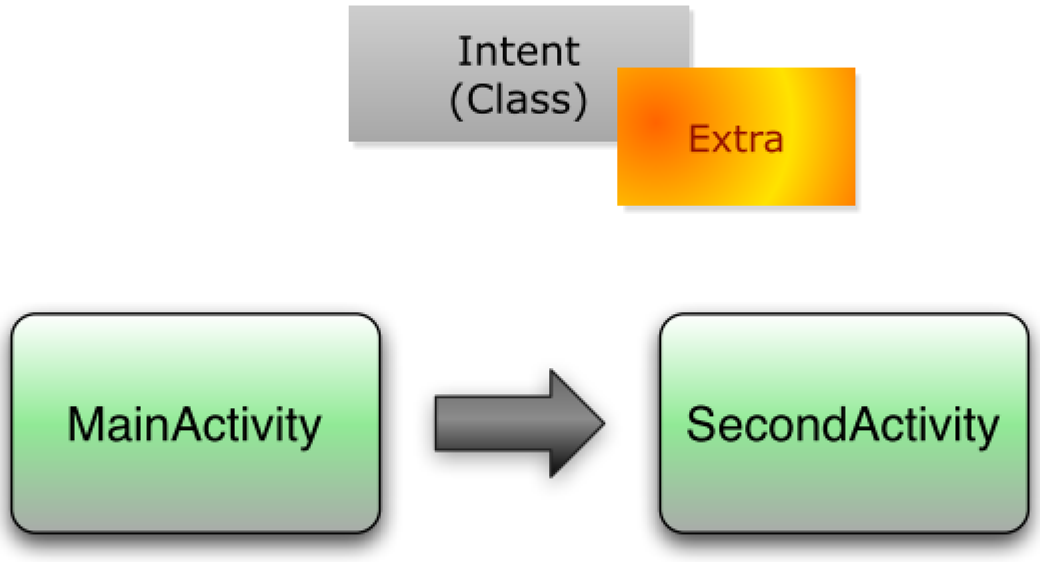
Intent with action
Implicitly call

```
public void Test() {
    Intent intent = new
    Intent(Intent.ACTION_CALL);
    intent.setData(
        Uri.parse("tel:41893263"));
    startActivity(intent);
}
```

Intent with class Explicitly

```
public void Test() {
    Intent intent = new Intent();
    intent.setClass(this,
    MainActivity.class);
    startActivity(intent);
}
```

Extras



Sending Extra

- Passing: (From “sender”)

```
public void Test() {
Intent intent = new Intent();
intent.putExtra("value","Test");
intent.setClass(this, MainActivity.class);
startActivity(intent)
}
```

- Retrieving: (In “receiver”)

```
getIntent().getStringExtra("value");
```

Getting Extra

```
 @Override
public void onCreate(Bundle
savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     String message =
     (String)getIntent().getStringExtra(
     "value");
………
}
```

Activities with results

- Starting activities Initializing activity

```
……..  Put these lines where need
Intent intent = new Intent(this, SecondActivity.class);
startActivityForResult(intent, “ChildID");
……
// Listen for results. Method to implement in activity to subscribe for a result
protected void onActivityResult(int requestCode, int resultCode, Intent data){
    // See which child activity is calling us back.
    switch (requestCode) {
        case: “ChildID" …….
        default:
            break;
    }
} // end method
```

- Inside SecondActivity put these lines where needed

```
setResult(RESULT_OK); // + more option for returning data, se SDK doc
finish(); // Finish an return to calling Activity
```

Lesson 3 Exercise 2

- Create a Android Project
- Create new activity, SecondActivity
- Create a new layout for SecondActivity
- in onCreate() of first activity, switch to SecondActivity
- Pass a value and display a Toast in SecondActivity with the Extra value that was passed.

Lesson 3 Exercise 3

- Continue with Lesson 3 Exercise 2
- Now add a two way switch between the two activities using the Activity.startActivityForResult()
- When returning to the first activity a notification must be set in the Status Bar using the Notification Manager
  - Then try using an Ongoing and Insistent notifications
  - Hint use [Meier] p402-423
  - (And if possible try using light, sounds and/or vibration)