Android

Programmeertechnieken, Tim Cocx

Universiteit Leiden
The Netherlands

Android

- What is Android?
- Dalvik virtual machine
- Android SDK
- Anatomy of an Android application
- Excercises

What is Android ?

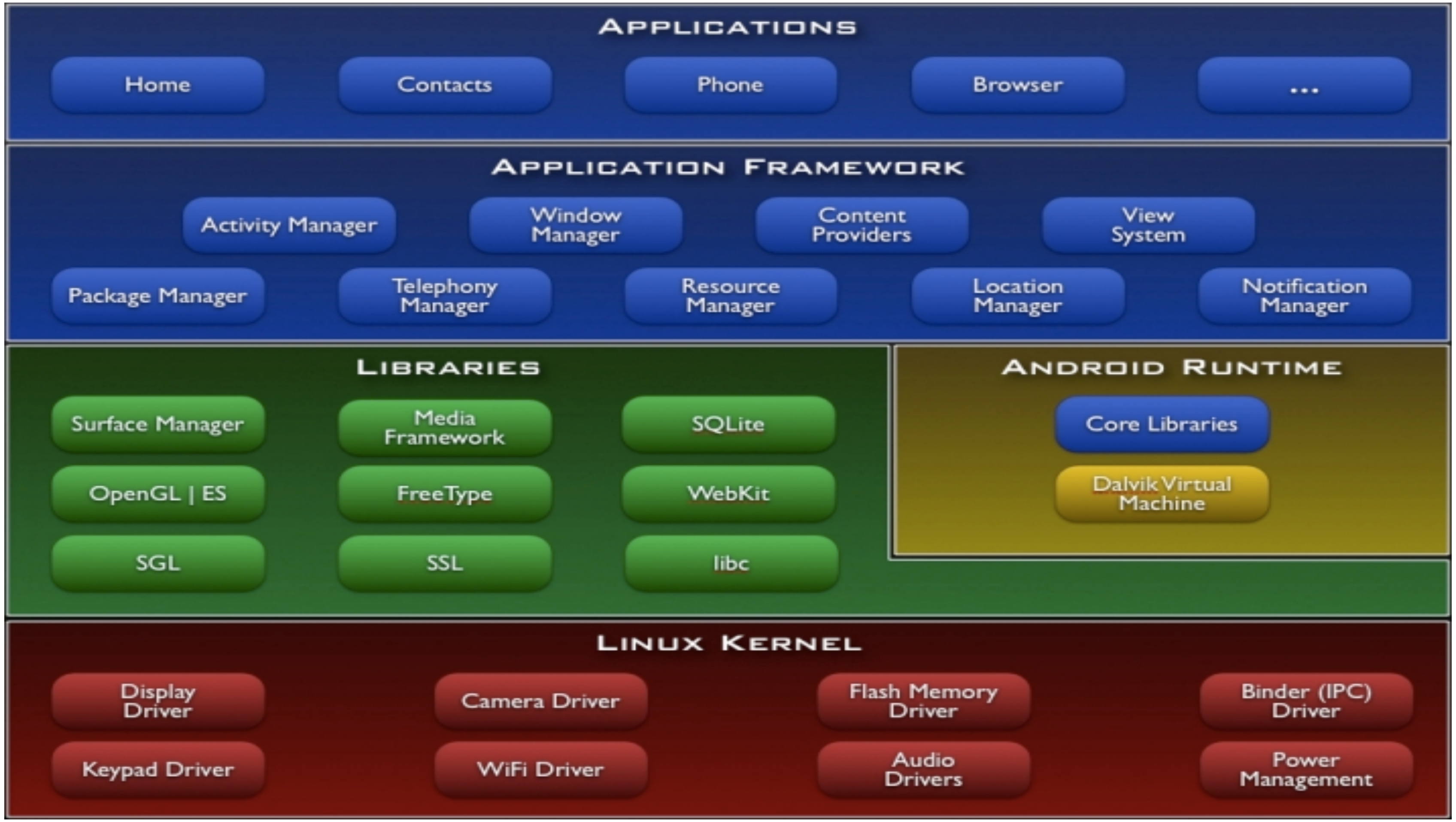
- Android is a mobile operating system that is based on a modified version of Linux.
- It was originally developed by a company with the same name (android, Inc.).
- Google purchased this company in 2005 and took over its development work and development team.
- Since then Android has become the most popular mobile operating system.
- Android is an open and free mobile operating system.
- As a result it can be used on different mobile devices of different vendors.

What is Android (2)

- Android is based on a Linux 2.6 kernel.
- It thus includes basic operating system capabilities such as memory management and process management.
- Some enhancements are made in order to let the operating system function optimally on a mobile device.
- On top of the adjusted kernel runs a Dalvik virtual machine.
- As a result Java programs can be run on android devices.

Architecture of Android

- As with most other operating systems Android can be roughly divided into layers:



- Notice the position of the Dalvik virtual machine:

Dalvik virtual machine

- Written by one person, Dan Bornstein
- Dalvik is a small village in Iceland where ancestors of Bornstein lived.

- To run the Java code, it is first compiled to special byte code that can be interpreted by the Dalvik virtual machine.

- Dalvik is optimized for the android platform. It can execute byte code faster and with less power consumption, because it contains some hardware dependent optimizations.
- Third party (some preinstalled) applications are also interpreted by the Dalvik virtual machine.

Android versions

- There hav ses. As a result there are a lot o

(date of figure: July 2012)

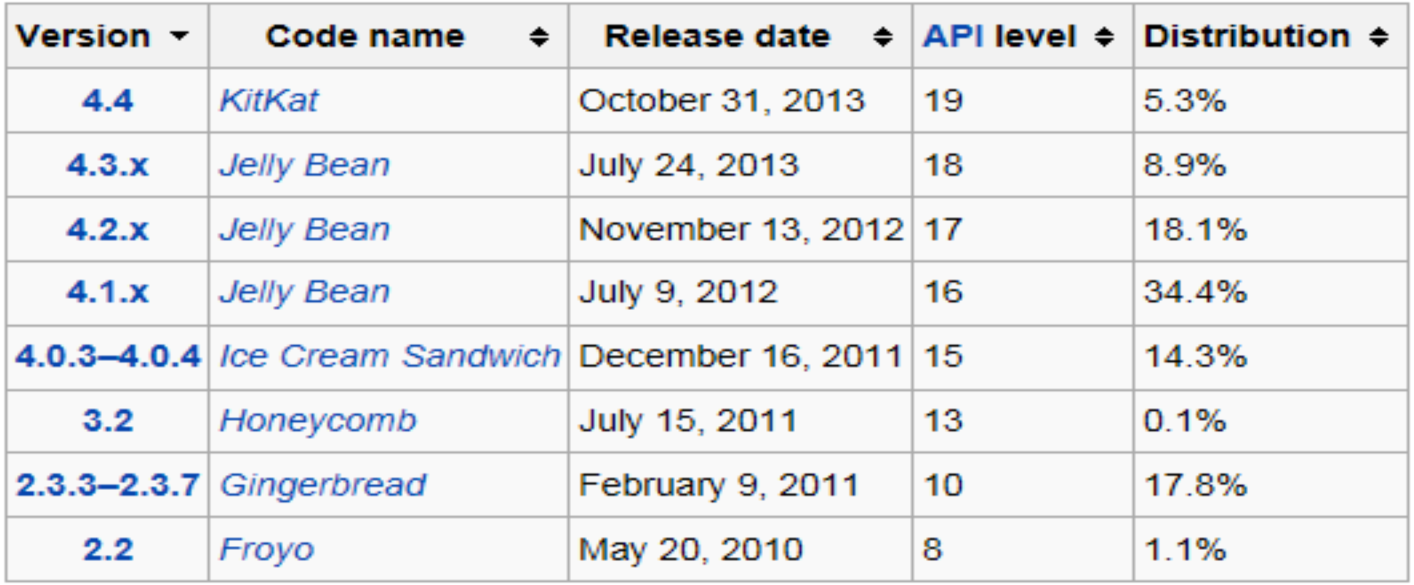
- Each new version contains bug fixes and new functionality, but more importantly also support for new devices such as tablets.

API's

- Third-party programs need to interact with the android operating system.
- Solution: Use API's (Application Programmers Interface) which are provided by the Android operating system. These are just methods from classes or objects which you can call from your Java program.
- In every android release these API's change because of new functionalities and improvements.
- Consequence: Every android release has a different set of API's
- You need to account for this when developing software!

API Levels

- To distinguish the different sets of API's each set is given a number. This is called an API level.
- Often each new Android version has got changes in the API, so the API level is increased.
- Overview of some different API levels:



Version	Code name	Release date	API level	Distribution
4.4	*KitKat*	October 31, 2013	19	5.3%
4.3.x	*Jelly Bean*	July 24, 2013	18	8.9%
4.2.x	*Jelly Bean*	November 13, 2012	17	18.1%
4.1.x	*Jelly Bean*	July 9, 2012	16	34.4%
4.0.3–4.0.4	*Ice Cream Sandwich*	December 16, 2011	15	14.3%
3.2	*Honeycomb*	July 15, 2011	13	0.1%
2.3.3–2.3.7	*Gingerbread*	February 9, 2011	10	17.8%
2.2	*Froyo*	May 20, 2010	8	1.1%

Android SDK

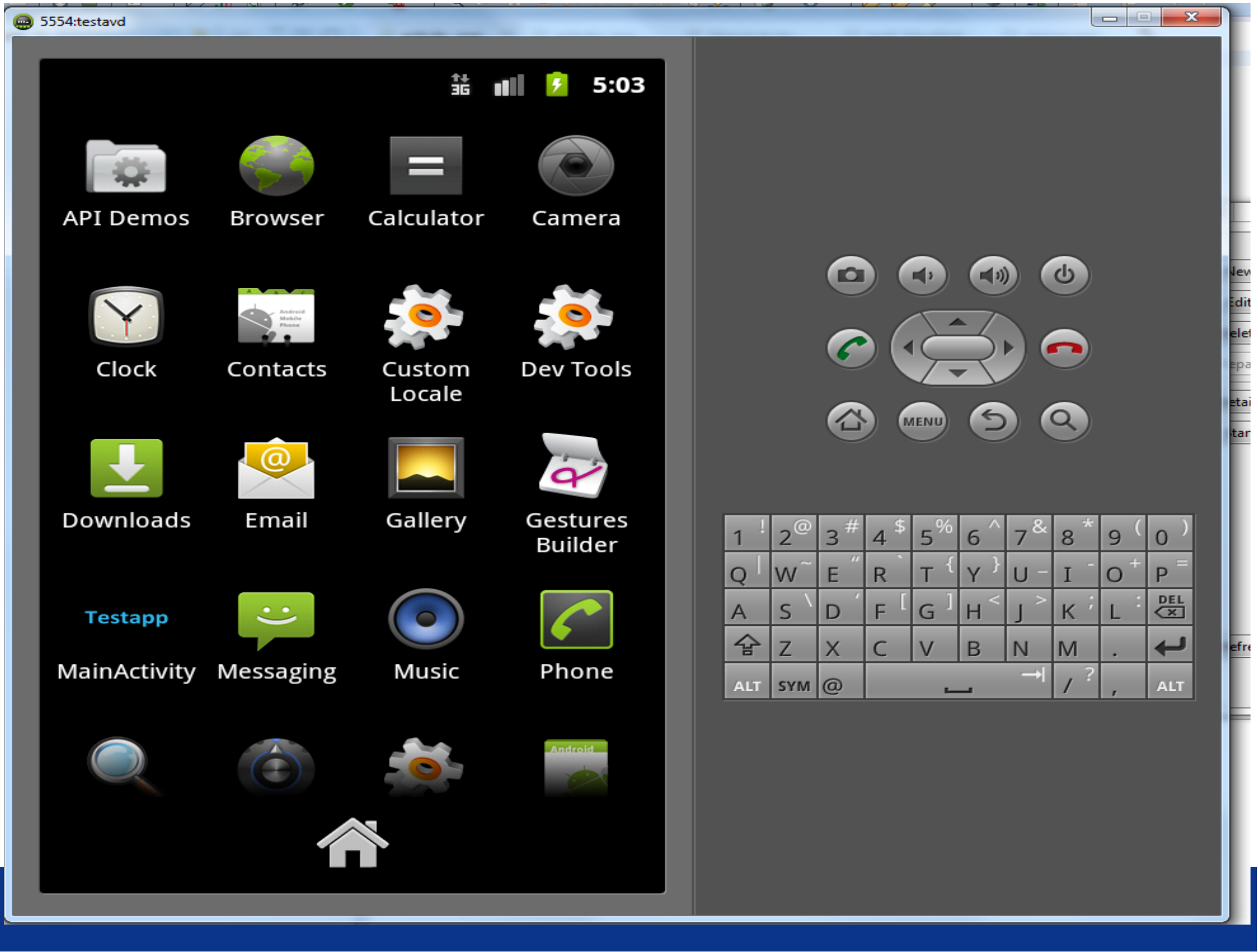
- In order for you to make android Java applications you need to use the android SDK.
- The SDK includes for example:
  - Software for compiling Dalvik virtual machine byte code
  - Method and class signatures of the API's.
  - Emulators for simulating your android programs
- The SDK can be integrated into a number of IDE’s

AVD (1)

- An Android Virtual Device (AVD) can be created once the SDK is installed
- An AVD is a simulation (actually emulation) of a real device and android operating system.
- You can use an AVD to test your own applications on different android versions and hardware.
- Your application may behave different on various devices due to the large number of android versions (API levels) and different hardware.
- It is therefore a wise idea to test your application on different android versions and hardware. An AVD allows you to do so.

AVD (2)

- The most important options for your AVD are:
  - Android version (this means also the API level)
  - Device RAM size
  - Resolution and size of the screen of the device
  - Existence and type of other hardware such as a camera, accelerometer keyboard etc.
- Options that are perhaps less important:
  - SD-card size



Anatomy of an android application

- The most important parts of an android application project are:
  - *src* : contains Java source files
  - *gen*: contains compiler generated files such as R.java
  - *res:* contains all the resources using your application. This includes for example images and layouts.
  - *bin*: this folder contains the generated .apk file. An apk file is the executable file which can be run by an android device. (comparable with a jar file)
  - *Androidmanifest.xml* : This is a file where important settings for your application are defined.

Androidmanifest.xml (1)

- Androidmanifest.xml is an XML file where important settings for your application are defined.
- It is mostly generated by the Android SDK.
- It contains detailed information about the application such as:
  - The permissions the application needs to run. You can notice the effects of this when you are installing an application and you need to allow the application to use for example the camera.
  - Which screens (activities) are in the application and which one to start with.
  - Application name and icons

Androidmanifest.xml (2)

- Example of an Androidmanifest.xml file

```
<?xml version="1.0" encoding="utf-8"?
<manifest xmlns:android="http://scher
  package="net.learn2develop.HelloWo
  android:versionCode="1"
  android:versionName="1.0" >

  <uses-sdk android:minSdkVersion="13" />

  <application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name" >
    <activity
      android:label="@string/app_name"
      android:name=".HelloWorldActivity" >
      <intent-filter >
          <action android:name="andr

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

</manifest>
```

-This means that this application needs API level 13.
-Android versions which only support API levels lower than 13
will not be able to run this application.

-The file res/values/strings.xml contains constants
which have values.
-*@string* is actually a reference to this particular
XML file and *app_name* is the name of the constant
within the XML file.
-These kind of constructions are often used in android Java
applications!

A simple activity

- A piece of Java code which will present a layout with for example buttons and labels is as follows:

```
public class HelloWorldActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInsta      te) {
        super.onC  ate(savedInstanceState
        setConte  ew(R.layou
    }
}
```

The Activity class is inherited, because we are creating a screen of our own and we want to use all the standard properties of the Activity class.

For now it is enough to assume that the *onCreate* method is called by the android OS when the activity (your program) must be started.

R.java

- The statement which will show the buttons, labels etc. is:

```
setContentView(R.layout.main);
```

- R (R.java) is a class which is automatically generated and contains constants with memory addresses.
- These addresses are ultimately used by *setContentView* to build the user interface with buttons and labels.
- How is R composed? It is generated by interpreting all the XML files of the project, so that Java code (which you can write) can reach the values that are originally in these files.
- *main* is a constant in R which contains an address. On this address all the information regarding the layout is stored.
- The data on this address is generated from an XML file.

Layout in XML

- The layout of an activity is thus defined in an XML file (in res/layout)
- Example of an XML file defining the layout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="This is my first Android Application.
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="And this is a clickable button!" />
</LinearLayout>
```

This is a GUI container which contains our components. (can be compared with a Jpanel in Java)

Defines a label with some text

The width of the component is adjusted to the width of the component where it is put in

Defines a button

- Why would the layout be defined in XML and not straight in Java code?

Building the GUI

- GUI components (such as text lines and buttons) can easily be created by introducing the corresponding XML tag in the GUI XML file.
- With properties of the XML tag you can adjust the component. (for example the width or color)
- There are of course other GUI components (such as radio buttons, spinners, progress bars etc.) which can also be shown by using the corresponding tag.
- All these components are shown and explained at: http://developer.android.com/guide/topics/ui/index.html
- Use this to see what a component does and what kind of properties it has.

Activities

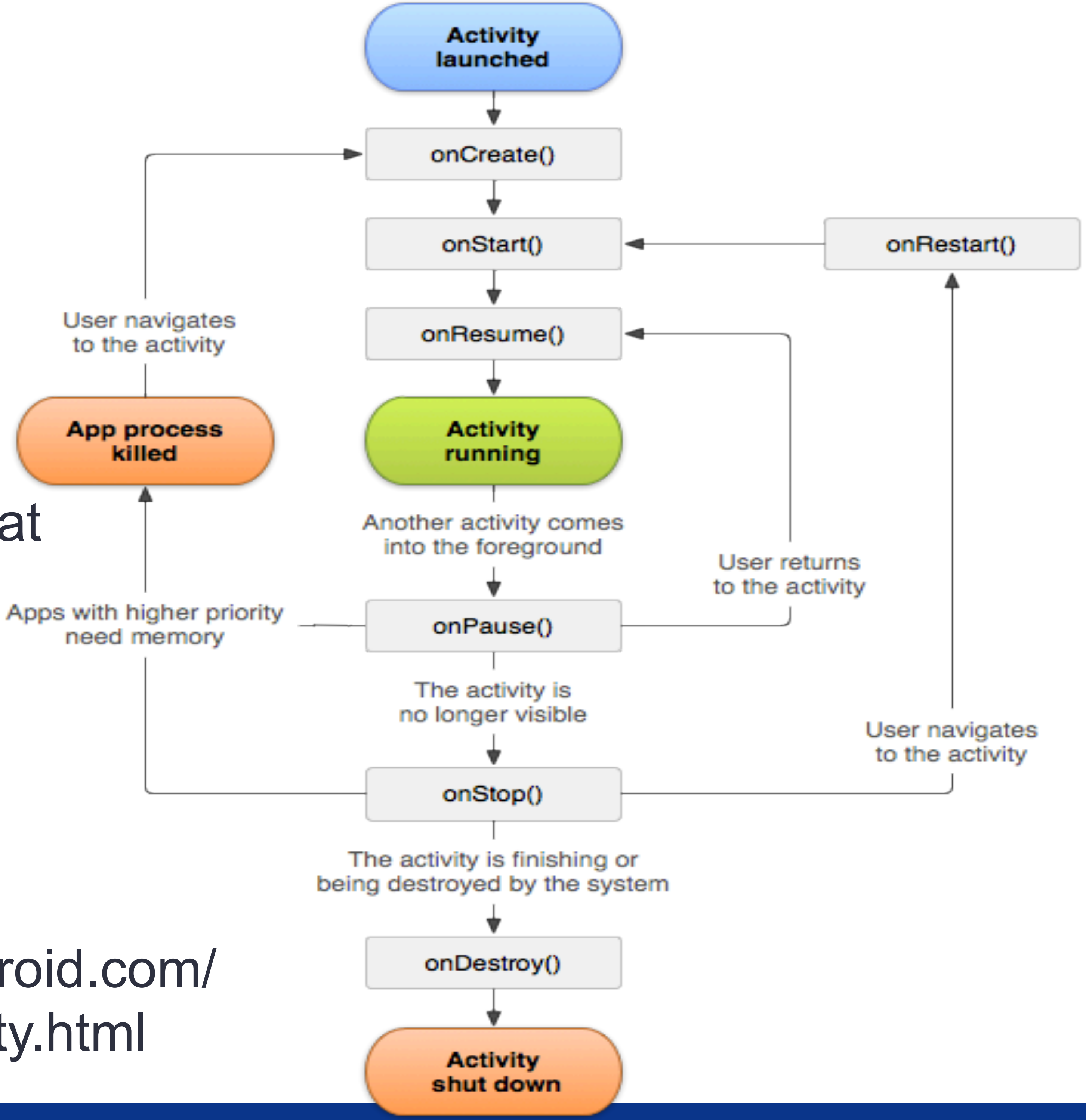
- Every activity you have in your application must be declared in the AndroidManifest.xml.
- The reasons for this will be clear later on.

```
<application
  ......
    <activity
     android:label="@string/app_name"
     android:name=".Activity101Activity" >
        <intent-filter >
            <action android:name="android.intent.action.MAIN" />
            <category android:name=
               "android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

This line will cause the android operating system to start with this activity.

Activity lifecycle (1)

- An activity has got various stages it goes through.
- Shown are the various stages and the corresponding methods that are called in a stage.


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Source: http://developer.android.com/reference/android/app/Activity.html

Activity lifecycle (2)

- From the previous image the following guidelines for the use of the different methods can be derived:
  - *onCreate():* Use this method to create and instantiate the objects you will be using in your application.
  - *onResume()*: Use this one to start any services or code that needs to run while your activity is in the foreground.
  - *onPause()*: Use this one to stop any services or code that does not need to run when your activity is not in the foreground.
  - *onDestroy()*: This method can be used to free up resources before your activity is destroyed.
- Note that you don't have any control on when these methods are called! This is all determined
  by the framework

How to start

1. Start a new Android Studio Project
2. Choose an Application name
3. Choose a company domain
    1. Needed for official purposes like play store, rights to publish, financial matters, etc.
4. Choose an SDK version
5. Choose ‘Add Empty Activity’
6. You now have:
    1. MainActivity.java ←code
    2. Activity_main.xml ← layout with ‘hello world’ TextView
    3. AndroidManifest.xml

To run on real device

1. Connect device ;-)
2. Enable USB debugging on your device by going to Settings > Developer options.
    1. Note: On Android 4.2 and newer, Developer options is hidden by default. To make it available, go to Settings > About phone and tap Build number seven times. Return to the previous screen to find Developer options
3. Click the ‘Play” icon
    1. Select deployment target→ select the device

To Run on emulator

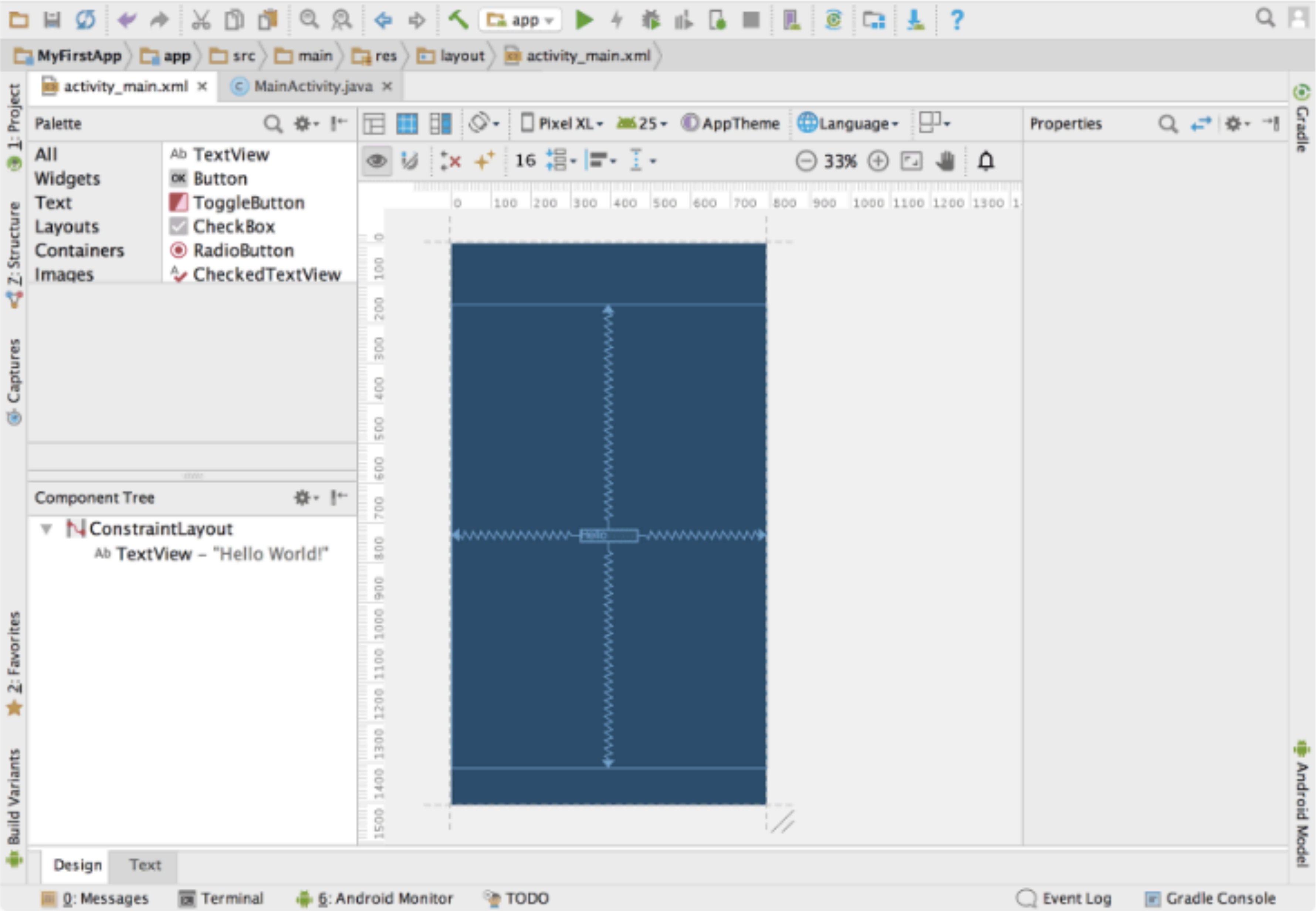
1. Start the AVD manager by clicking
2. Virtual Devices → create virtual device
3. Select Hardware → select a phone device
4. System Image → download an image
5. Finish
6. Virtual Devices→ Launch this AVD in the emulator
7. Click the ‘Play’ icon
8. Select Deployment target →emulator

Build an interface

[from Android Studio 2.3]

1. Open app/res/layout/activity_main.xml
2. If XML is showed click the design tab
3. Click show blueprint ▦
4. Click device in editor ▯ and select the required device

Editor


app
MyFirstApp
app
src
main
res
layout
activity_main.xml
activity_main.xml
MainActivity.java
Palette
All
Widgets
Text
Layouts
Containers
Images
TextView
Button
ToggleButton
CheckBox
RadioButton
CheckedTextView
Pixel XL
25
AppTheme
Language
16
33%
Properties
Component Tree
ConstraintLayout
TextView - "Hello World!"
Design
Text
1: Project
Z: Structure
Captures
2: Favorites
Build Variants
Gradle
Android Model
0: Messages
Terminal
6: Android Monitor
TODO
Event Log
Gradle Console

Add controls

1. Click Tekst in the Palette and select plain Text
2. Click it in the design editor
3. If you snap the arrow to the rulers it will stay at the top / right / left / bottom

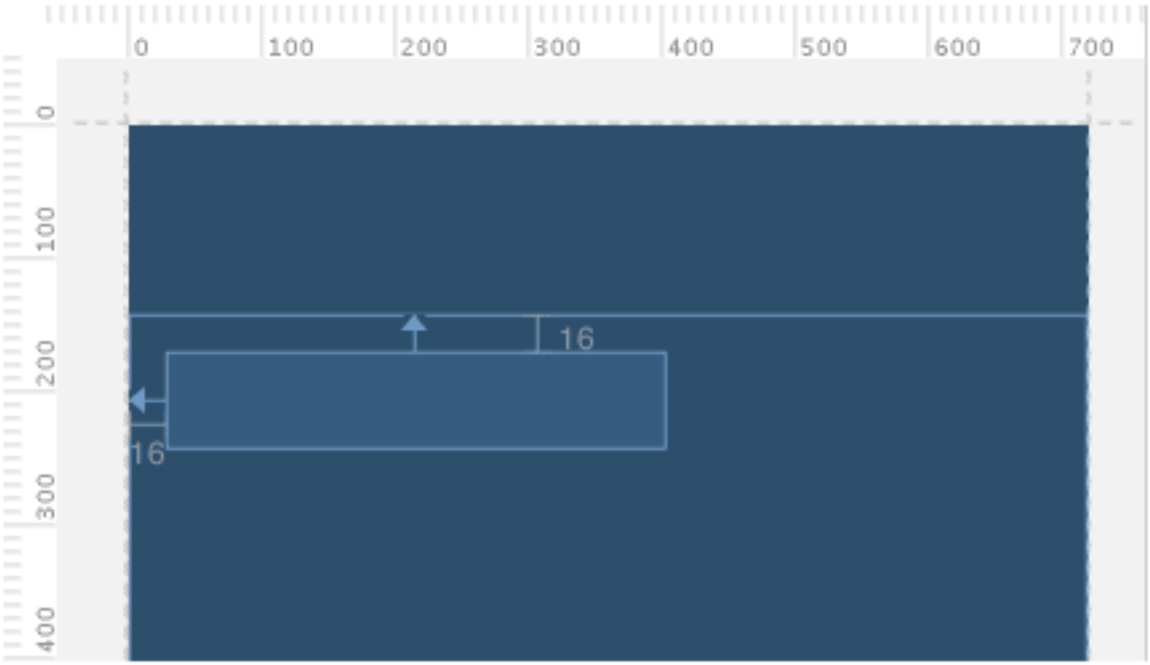
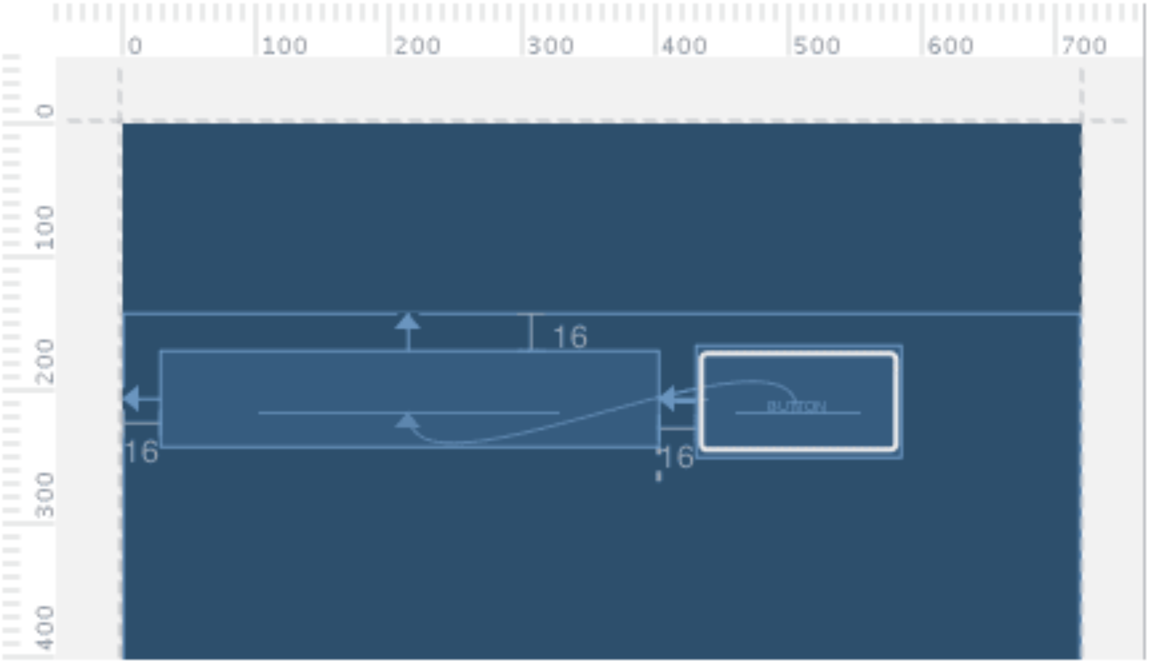


Figure 5. The text box is constrained to the top and left of the parent layout

Add a button

- Do the same for a button
- Snap it to the TextView
- Align the tekst base using : ab.



Respond to button

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    /** Called when the user taps the Send button */
    public void sendMessage(View view) {
        // Do something in response to button
    }
}
```

In the Properties window, locate the onClick property and select sendMessage [MainActivity] from the drop-down list.