Android Programing

Gavrilut Dragos

Administrative

Final grade for the Android course is computed using Gauss over the total points accumulated.

- Maximum 100 points from the laboratory examination
- Maximum 60 points at the final examination (course)

The minimum number of points that one needs to pass this exam:

- Minimum 50 points accumulated from the laboratory examination
- Minimum 20 points from the final examination (course)

- Course page: https://sites.google.com/site/fiiandroidprogramming/home

History

- 2003 October – Android Inc, is founded
- 2005 August – Google Inc. buys Android Inc for 50.000.000 USD
- 2007 – Android becomes an open-source project under Apache license
- 2008 October – first device with Android OS is release

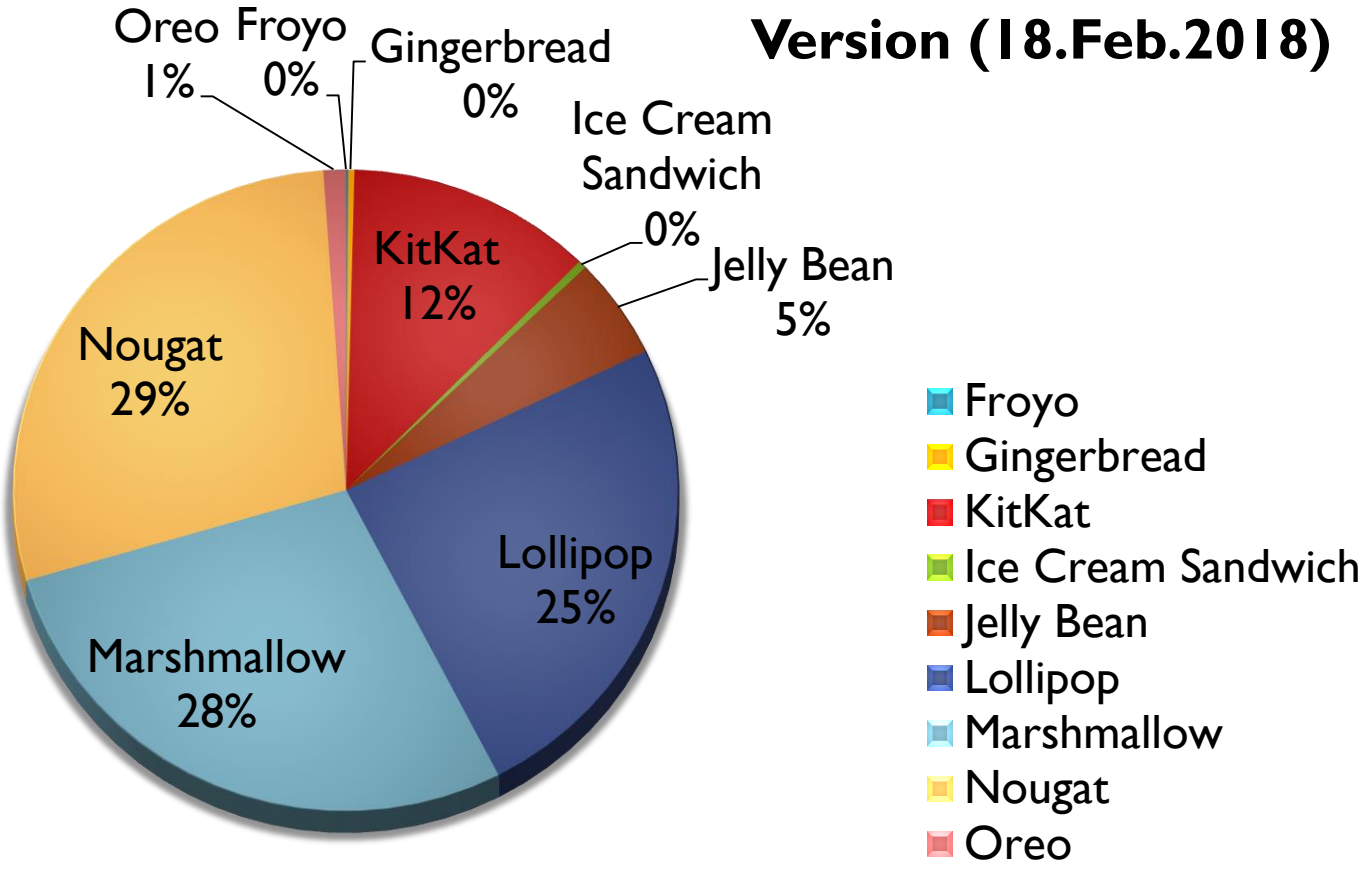
Android versions

Release date	Version	Code Name	API Level
2008.Sep	1.0	-	1
2009.Feb	1.3	Petite Four	2
2009.Apr	1.5	Cupcake	3
2009.Sep	1.6	Donut	4
2009.Oct	2.0	Eclair	5
2009.Dec	2.0.1	Eclair	6
2010.Ian	2.1	Eclair	7
2010.May	2.2	Froyo	8
2010.Dec	2.3 – 2.3.2	Gingerbread	9
2011.Feb	2.3.3 – 2.3.7	Gingerbread	10
2011.Feb	3.0	Honeycomb	11
2011.May	3.1	Honeycomb	12

Android versions

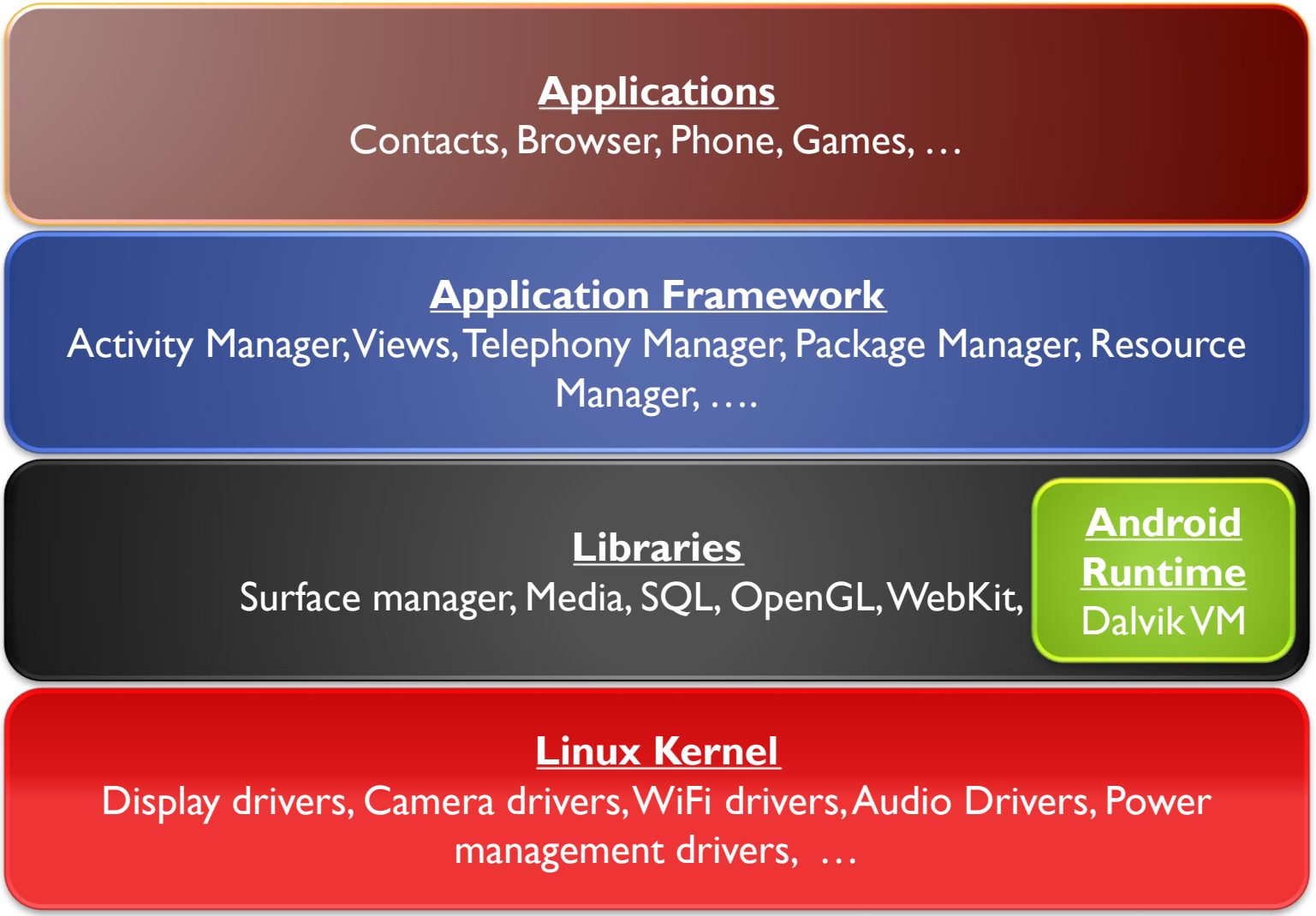
Release date	Version	Code Name	API Level
2011.July	3.2	Honeycomb	13
2011.Oct	4.0-4.0.2	Ice Cream Sandwich	14
2011.Dec	4.0.3 – 4.0.4	Ice Cream Sandwich	15
2012.Jul	4.1	Jelly Bean	16
2012.Nov	4.2	Jelly Bean	17
2013.Jul	4.3	Jelly Bean	18
2013.Sep	4.4	Kit Kat	19
2014.July	4.4W	Kit Kat	20
2014.Nov	5.0	Lollipop	21
2015.Mar	5.1	Lollipop	22
2015.Oct	6.0	Marshmallow	23
2016.Aug	7.0 – 7.1	Nougat	24/25
2017,Aug	8.0 – 8.1	Oreo	26/27

Android Distribution



http://developer.android.com/about/dashboards/index.html

Android architecture



Android architecture

- Hardware platform
  - ARM
  - X86 (Google TV, …)
  - i.MX
  - Intel

Mobile architecture

- Touch Screens
- SMS & Phone
- GPS
- Flash Drives
- NFC
- WiFi
- DLNA
- ….

Legal issues

- Google vs Oracle
  - Java structure
- Apple vs Samsung
  - Design
- Apple & Microsoft vs HTC & Samsung
- Google buys Motorola

File Hierarchy

- Different file systems (YAFFS, EXTx, proprietary (Samsung RFS), F2FS, F2FS, JFFS2, …)
- Partitions:
  - /cache
  - /system
  - /sdcard
  - /mnt
  - /sys
  - /data
  - /root
  - /dev
  - …

File Hierarchy

- /cache
  - Cached files and data
- /sdcard
  - Contains application data, pictures, etc
  - Some applications use this partition as a way to record different data regarding their instalation (install date, etc)
- /system
  - Includes Android OS files (except for kernel) such as libraries, fonts, default applictions (email, browser, phone, …), system sounds, linux executables for different commands (ls, rm, su, …), …

File Hierarchy

- /data
  - User specific data (contacts, messages, settings, …)
  - Private data and libraries for every application installed (by package)
  - Installed applications

APK format

- APK = **A**pplication **P**ackage **F**ile
- ZIP archive
  - **\META-INF**
    - MANIFEST.MF
    - CERT.RSA
    - CERT.SF (SHA-1 digest for MANIFEST.MF)
  - **\lib**
    - \armeabi
    - \armeabi-v7a
    - \x86
    - \mips
  - **\res**
    - drawable-{xxx}
    - raw-{xxx}
    - layout –{xxx}
    - menu
  - **\assets**
  - classes.dex
  - resources.arsc
  - AndroidManifest.xml

Dalvik

- Register-based VM
- Sandbox
- A **DEX** file contains all the information required for the Dalvik VM to execute the code (libraries, endianess, …)
- Can run native code.
- "Every Android application runs in its own process, with its own instance of the Dalvik virtual machine."

DEX files

- Header (CheckSum, SHA, …)
- String Indexes
- Type Indexes
- Prototype Indexes
- Field Indexes
- Method Indexes
- Class Definitions
- Data (dex code , strings, classes, …)

Zygote process

- Use to increase the start time of a dalvik VM process
- Shares constant data (libraries) between instances of VM processes

```
def ZygoteStart:
        while (true)
                if (new app is requested)
                        fork()
                endif
        endwhile
enddef
```

- Uses “copy-on-write” to copy modified memory to a spawn child

Aplications

- Each Android Application runs as a linux process
- Each Android Application has multiple components:
  - Activities
  - Services
  - Content Providers
  - Broadcast Receivers
- Each Android Application can start another Android Application components (use an activity from email application to send email). This can be done using **Intent** object.

Aplications

- Each Android Application has its own process. Each process has a rank (importance) in Android. The more important a process is, the less is the chance that it will be killed by the system.
- There are 5 ranks for processes:
  1. Forenground process
     - Has an Activity that users interacts with
     - Has a Service that interacts with a Forenground process
     - Has a Service that runs in forenground
     - Has an active Broadcast Receiver
  2. Visible process
     - Has an Activity that is in background (paused)
     - Has a Service that is linked to an background activity
  3. Service process
     - Has a Service
  4. Background process
     - Has an Activity with a process that was stopped
  5. Empty process
     - Does not have any components. It is maintain for caching purposes.

Aplications

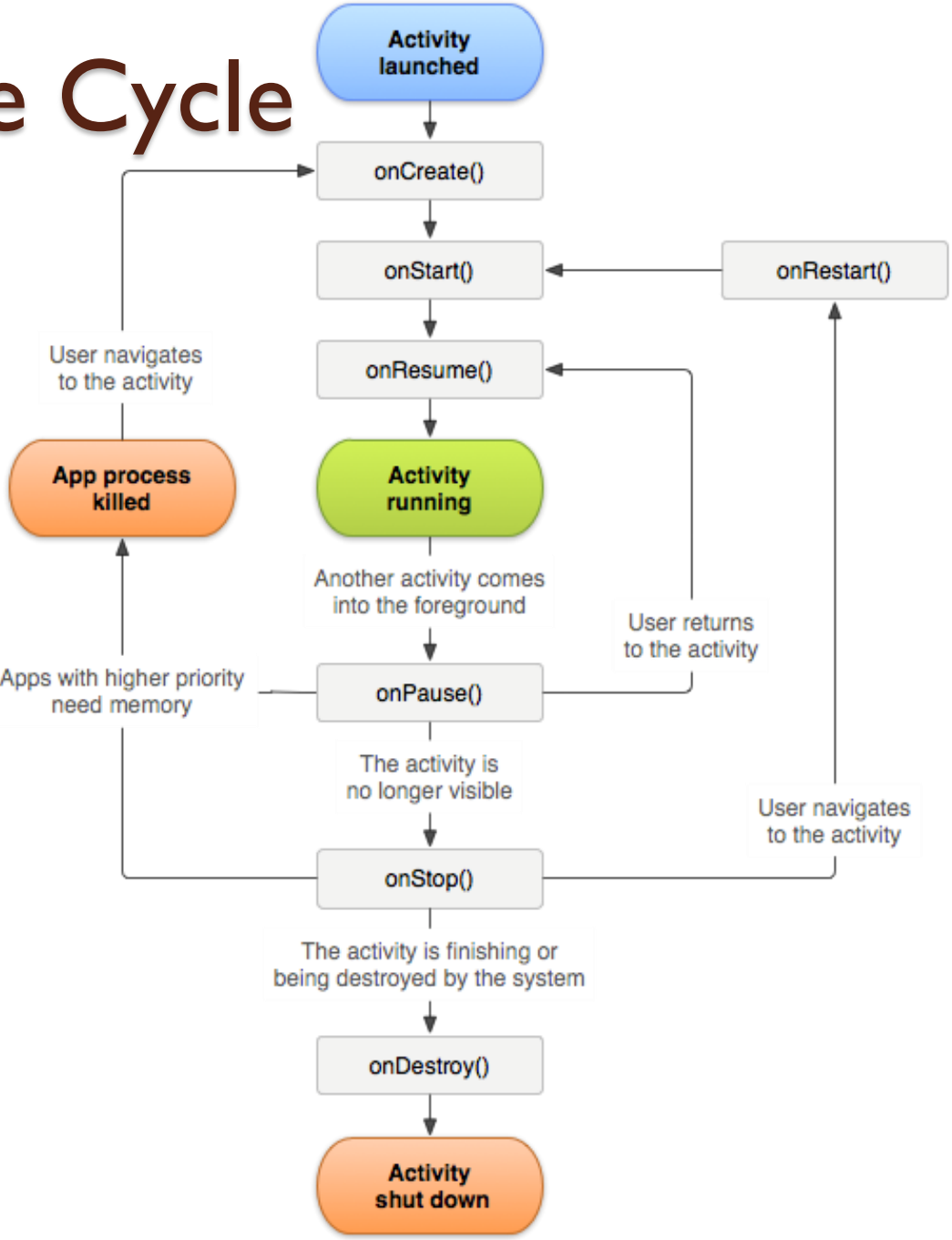
- To execute an Android Application the system checks **AndroidManifest.xml** file
- AndroidManifest file contains:
  - Permissions
  - List of activities
  - List of services
  - List of receivers
  - List of providers

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png">
        <activity android:name="com.myapp.myActivity"
                  android:label="@string/HelloWorld">
        </activity>
        ...
    </application>
</manifest>
```

Aplications

```
<application android:allowTaskReparenting=["true" | "false"]
        android:description="string resource"
        android:hasCode=["true" | "false"]
        android:hardwareAccelerated=["true" | "false"]
        android:icon="drawable resource"
        android:label="string resource"
        android:name="string"
        android:permission="string"
        android:process="string"
        android:theme="resource or theme"
. . .
</application>
```

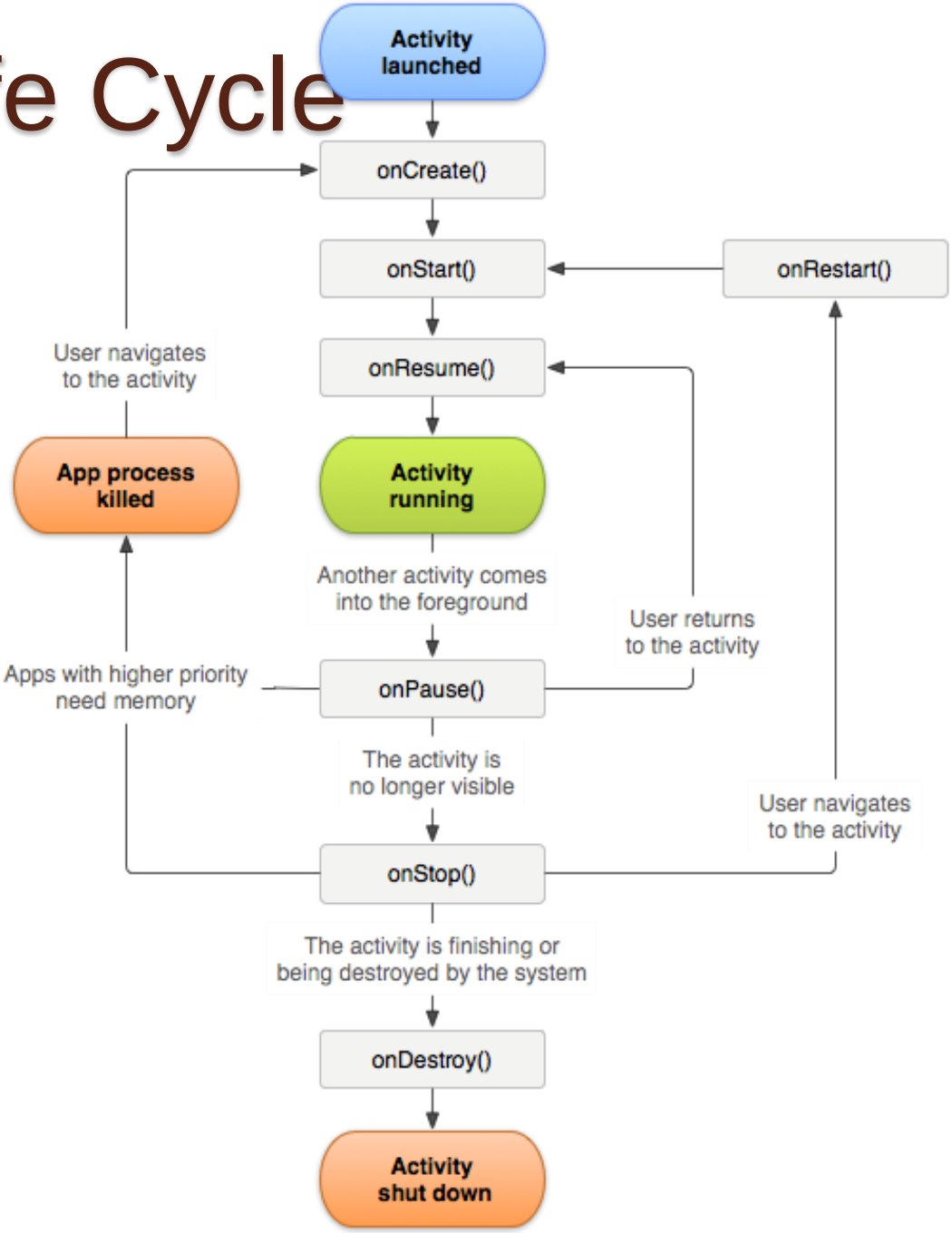
Activity Life Cycle



Android Programing

Gavrilut Dragos

Activity Life Cycle

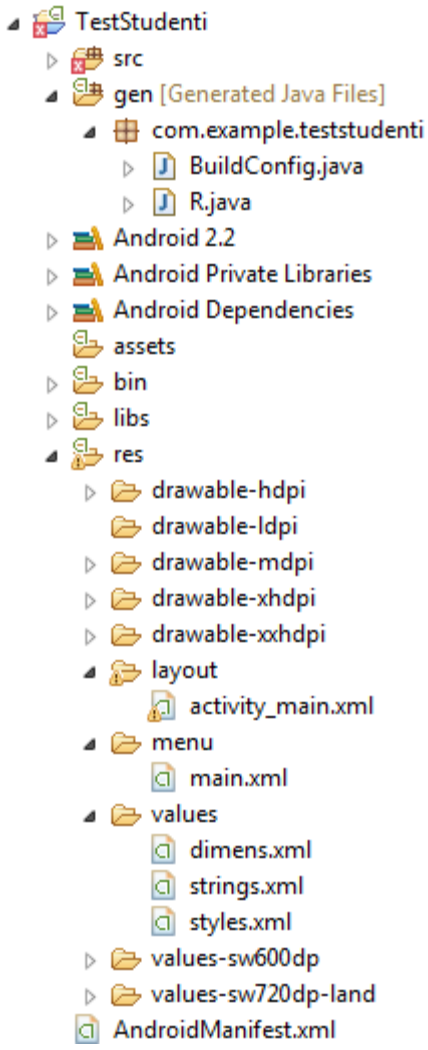


Activities - onCreate

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

- savedInstanceState – if different than null it represents a object with saved informations about the last state of the activity.
- It is set by **onSaveInstanceState** (Bundle outState)
- setContentView – sets the view object that will be displayed in the current activity
  - **public void** setContentView (**int** layoutResID)
  - **public void** setContentView (**View** view)
  - **public void** setContentView (**View** view, ViewGroup.LayoutParams params)

“R” file




```
package com.example.teststudenti;

public final class R {
    public static final class attr {
    }
    public static final class dimen {
        public static final int activity_horizontal_margin=0x7f040000;
        public static final int activity_vertical_margin=0x7f040001;
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int action_settings=0x7f080004;
        public static final int button1=0x7f080000;
        public static final int button2=0x7f080002;
        public static final int button3=0x7f080003;
        public static final int button4=0x7f080001;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    public static final class menu {
        public static final int main=0x7f070000;
    }
    public static final class string {
        public static final int action_settings=0x7f050001;
        public static final int app_name=0x7f050000;
        public static final int hello_world=0x7f050002;
    }
    public static final class style {
        public static final int AppBaseTheme=0x7f060000;
        public static final int AppTheme=0x7f060001;
    }
}
```

View

- The superclass for every UI design object in Android OS
- The following functions can be overwritten in a

Function	Action
onMeasure(int, int)	Called to compute the size of the view
onLayout(boolean, int, int, int, int)	Called to compute its layout position
onSizeChanged(int, int, int, int)	Called when the size of the view changed
onDraw(android.graphics.Canvas)	Called to draw the content of the view
onKeyDown(int, KeyEvent)	Called when a key is presses
onKeyUp(int, KeyEvent)	Called when a key is release
onTrackballEvent(MotionEvent)	Called when a track ball event occurs
onTouchEvent(MotionEvent)	Calles when a touch event occurs

View

- UI Objects derived from view
  - ImageView
    - ImageButton
      - ZoomButton
  - ProgressBar
  - SurfaceView
  - TextView
    - Button
      - CheckBox
      - RadioBox
      - ToggleButton

View

- A view and its children can be design using 2 methods: XML design or direct code
- Using XML design means using different layouts to arrange the UI objects in a View.

Layout object	API	Description
AbsoluteLayout	1	Arrange its children after their absolute position (x,y). **Deprecated since API Level 3**
FrameLayout	1	Displays one child (in a frame)
LinearLayout	1	Arrange its childern in one row or one column
RelativeLayout	1	Arrange its children in relation one to another
TableLayout	1	Arrange its childern in a Table (similar to a table in a HTML view)
GridLayout	14	Arrange its children in a Grid (with columns and rows)

View layout XML parameters

- android:layout_width
- android:layout_height
  - fill_parent → as big as its parent (minus the padding). Deprecated since API Level 8 and replaced with math_parent
  - match_parent
  - wrap_content → as big as its childs require plus the padding. In case of Buttons, TextView,… it is as big as the text it contains.
  - or a unit (in pixels (px), density independent pixels (dp), inches (in) , millimeters (mm)

View layout XML parameters

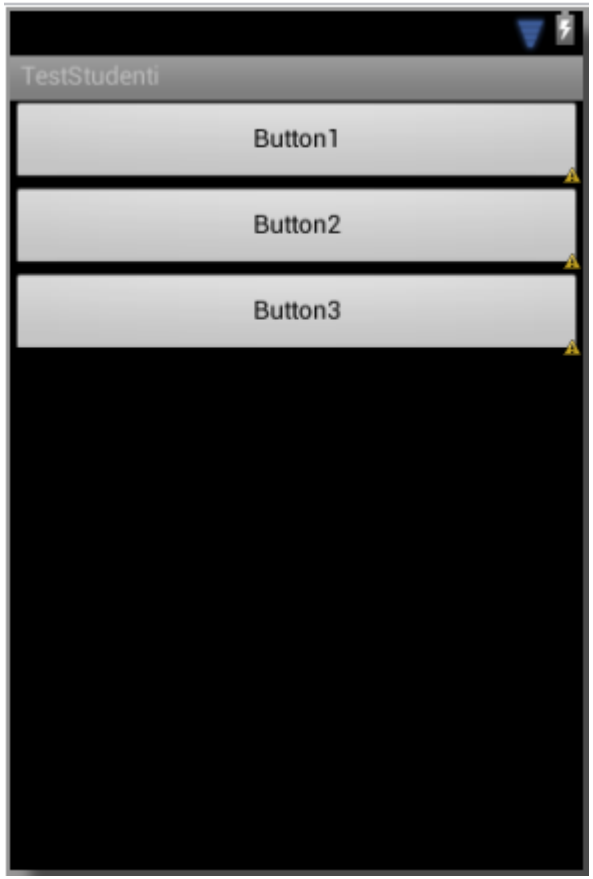
- android:id → object id (will be used in findViewById function).
  Exemple: android:id="@+id/button2"
- android:enabled → True/False
- android:onClick → name of the function to process the onClick event for the View

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    …
}
public void OnButton1Click(View button)
{
    …
}
```

```
<Button
    android:id="@+id/button1“
    android:onClick="OnButton1Click"
    android:text="Button" />
```

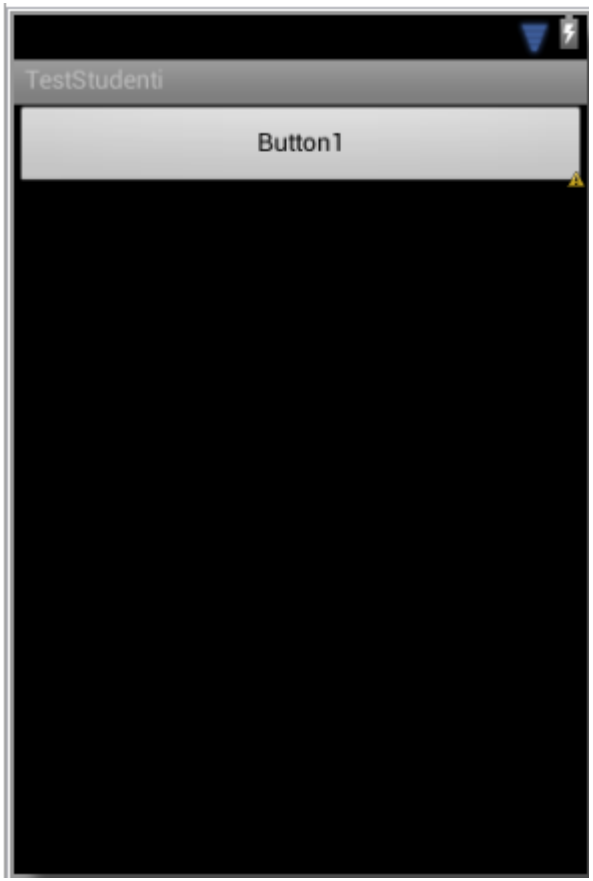
Linear Layout

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button1" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button2" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button3" />

</LinearLayout>
```



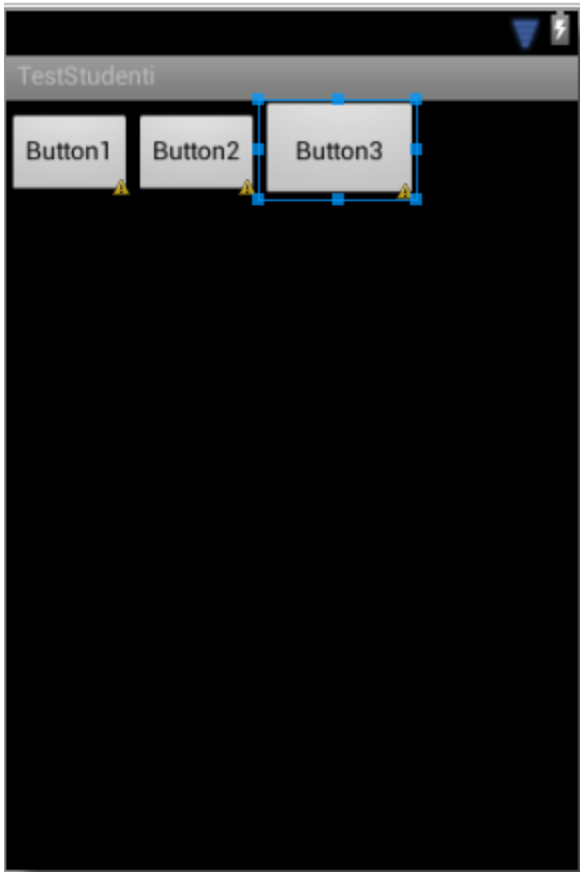
Linear Layout

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button1" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button2" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button3" />

</LinearLayout>
```



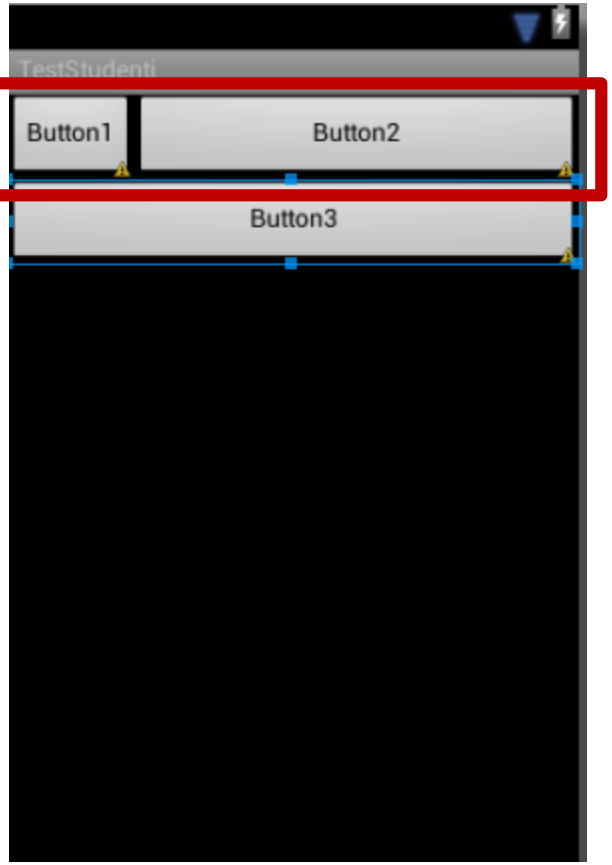
Linear Layout

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button2" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button3" />

</LinearLayout>
```



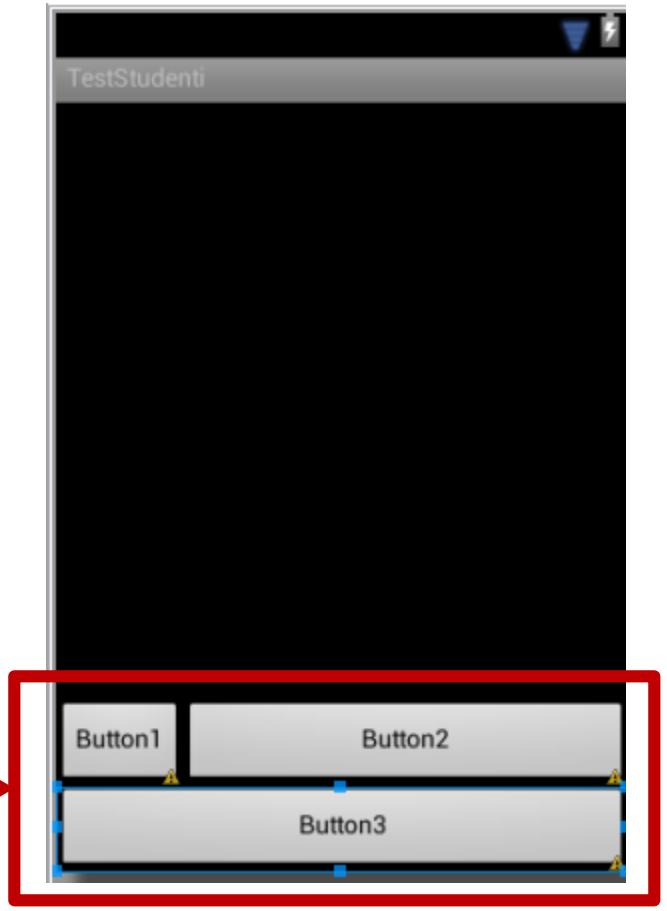
Linear Layout

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button1" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button2" />
    </LinearLayout>
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button3" />
</LinearLayout>
```



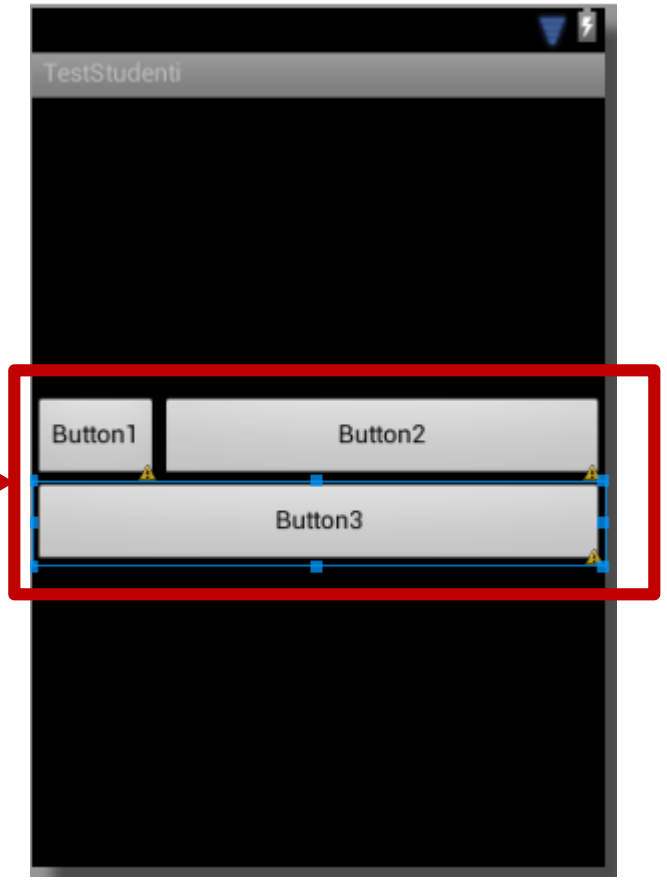
Linear Layout

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent“
    android:gravity="bottom"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button1" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button2" />
    </LinearLayout>
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button3" />
</LinearLayout>
```



Linear Layout

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent“
    android:gravity=“center"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button1" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button2" />
    </LinearLayout>
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button3" />
</LinearLayout>
```



Linear Layout

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent“
    android:gravity="center"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <Button
            android:layout_width=“150px"
            android:layout_height="wrap_content"
            android:text="Button1" />
        <Button
            android:layout_width=“wrap_content"
            android:layout_height="wrap_content"
            android:text="Button2" />
    </LinearLayout>
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button3" />
</LinearLayout>
```

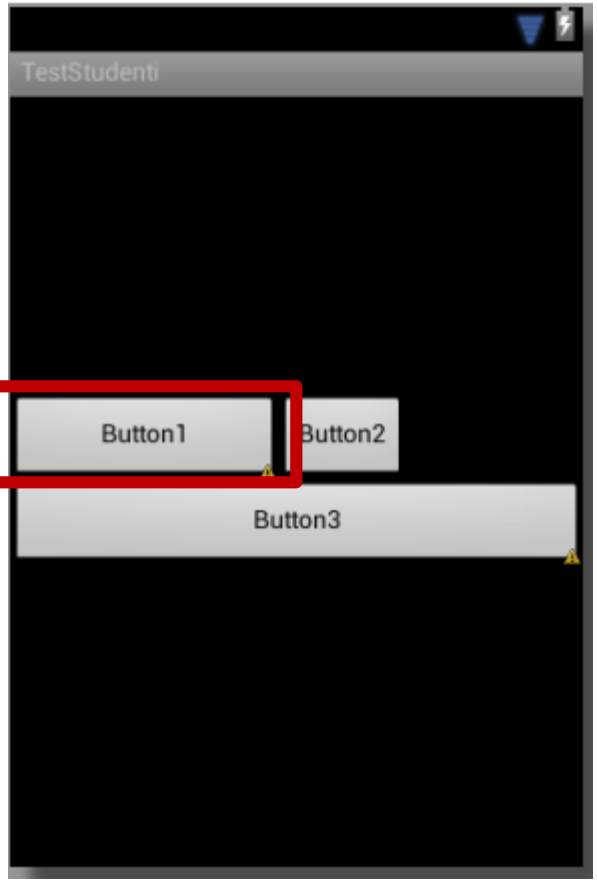
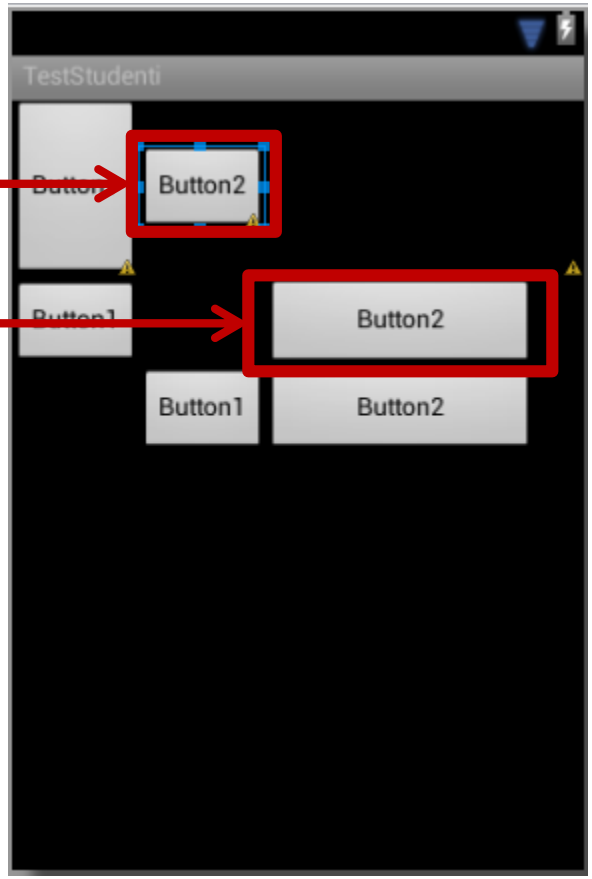


Table Layout

```
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TableRow>
      <Button android:layout_width="wrap_content"
              android:layout_height="100px"
              android:text="Button1" />
      <Buttor android:layout_width="10px"
              android:text="Button2" />
    </TableRow>
    <TableRow>
      <Button android:text="Button1" />
      <Button android:layout_width="150px"
              android:layout_height="50px"
              android:layout_column="2"
              android:text="Button2" />
    </TableRow>
    <TableRow>
      <Button android:layout_column="1"
              android:text="Button1" />
      <Button android:text="Button2" />
    </TableRow>
</TableLayout>
```



Column size is set as the biggest width from all of the elements/View from that column

Table Layout

```
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TableRow>
      <Button android:layout_width="wrap_content"
              android:layout_height="100px"
              android:text="Button1" />
      <Button android:layout_width="10px"
              android:text="Button2" />
    </TableRow>
    <TableRow android:layout_gravity="right" >
      <Button android:text="Button1" />
      <Button android:layout_width="150px"
              android:layout_height="50px"
              android:layout_column="2"
              android:text="Button2" />
    </TableRow>
    <TableRow>
      <Button android:layout_column="1"
              android:text="Button1" />
      <Button android:text="Button2" />
    </TableRow>
</TableLayout>
```

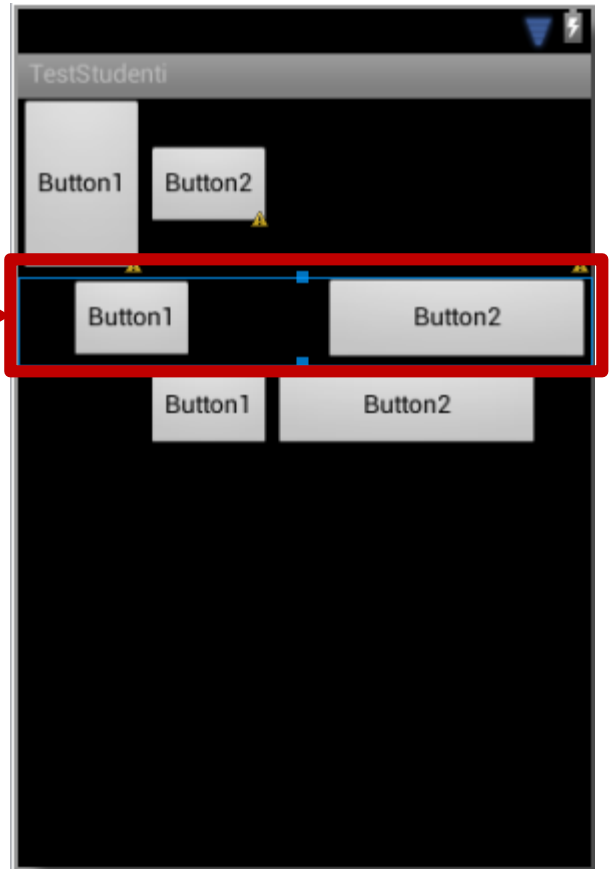


Table Layout

```
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:shrinkColumns="*"
    android:stretchColumns="*" >

    <TableRow android:layout_width="fill_parent"
              android:gravity="center_horizontal">
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button1" />
    </TableRow>

    <TableRow android:gravity="right" >
        <Button android:text="Button1" />
        <Button android:text="Button2" />
    </TableRow>

</TableLayout>
```

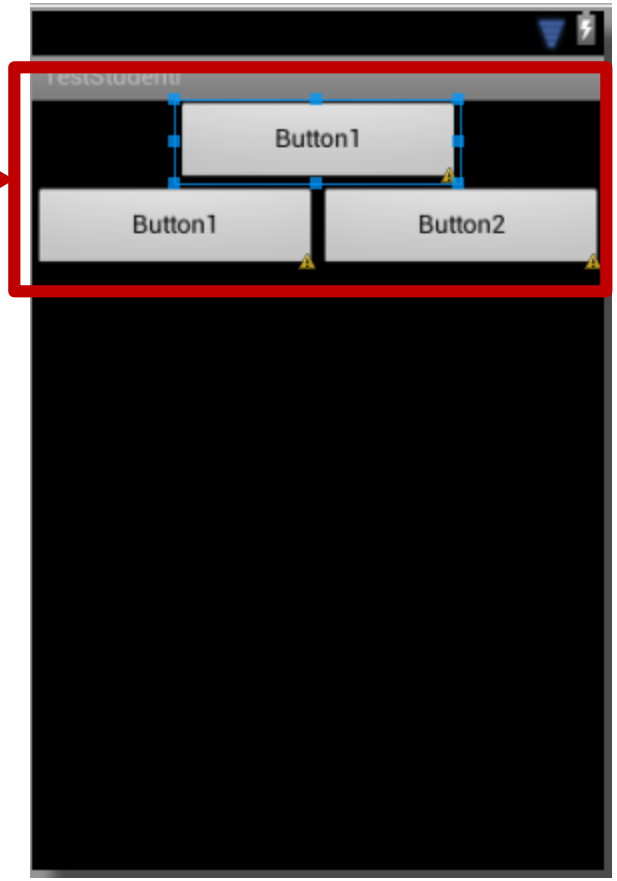


Table Layout

```
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TableRow android:layout_width="fill_parent">
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_span="2"
            android:text="Button1" />
        <Button android:text="Button2" />
    </TableRow>

    <TableRow android:gravity="right" >
        <Button android:text="Button3" />
        <Button android:text="Button4" />
        <Button android:text="Button5" />
    </TableRow>

</TableLayout>
```

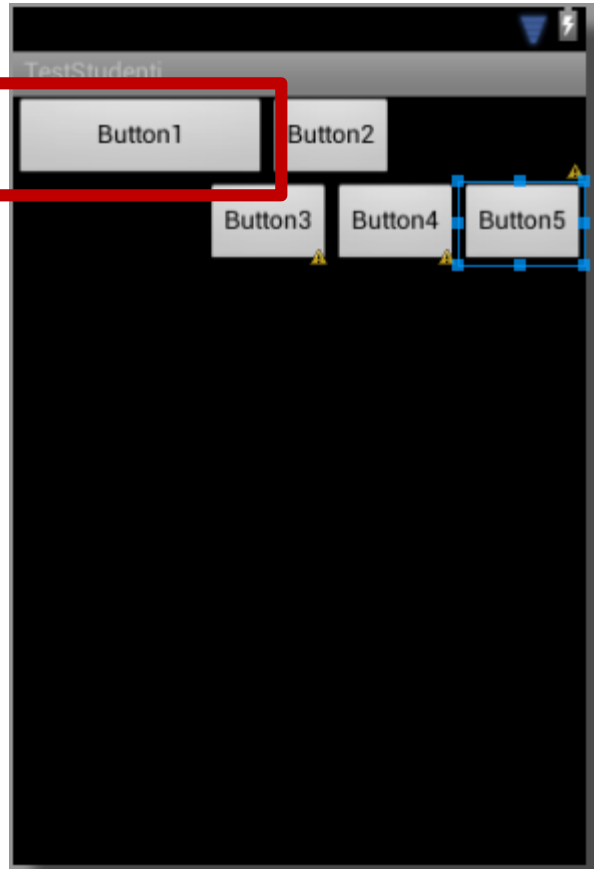


Table Layout

```
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent“
    android:shrinkColumns=“*“
    android:stretchColumns=“*“
>

    <TableRow android:layout_width="fill_parent">
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_span="2"
            android:text="Button1" />
        <Button android:text="Button2" />
    </TableRow>

    <TableRow android:gravity="right" >
        <Button android:text="Button3" />
        <Button android:text="Button4" />
        <Button android:text="Button5" />
    </TableRow>
 </TableLayout>
```

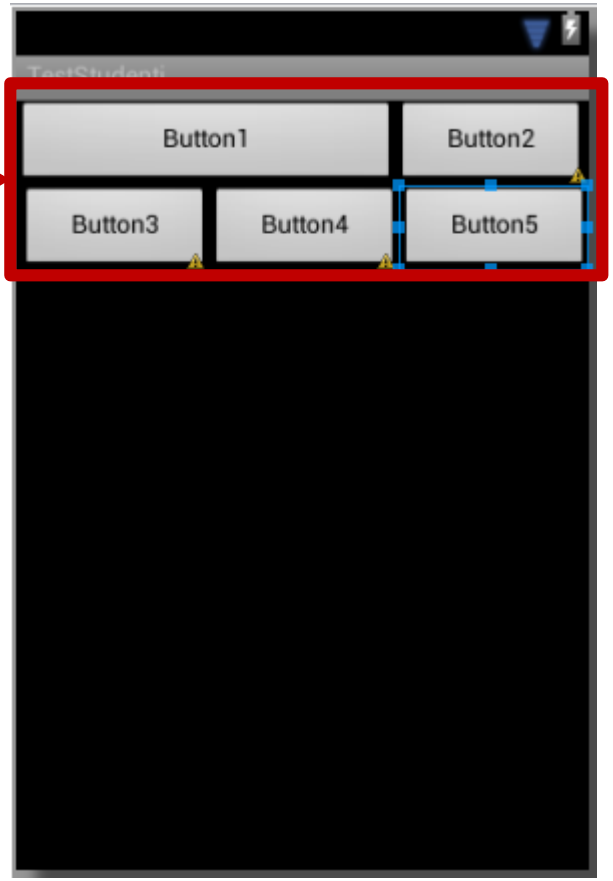
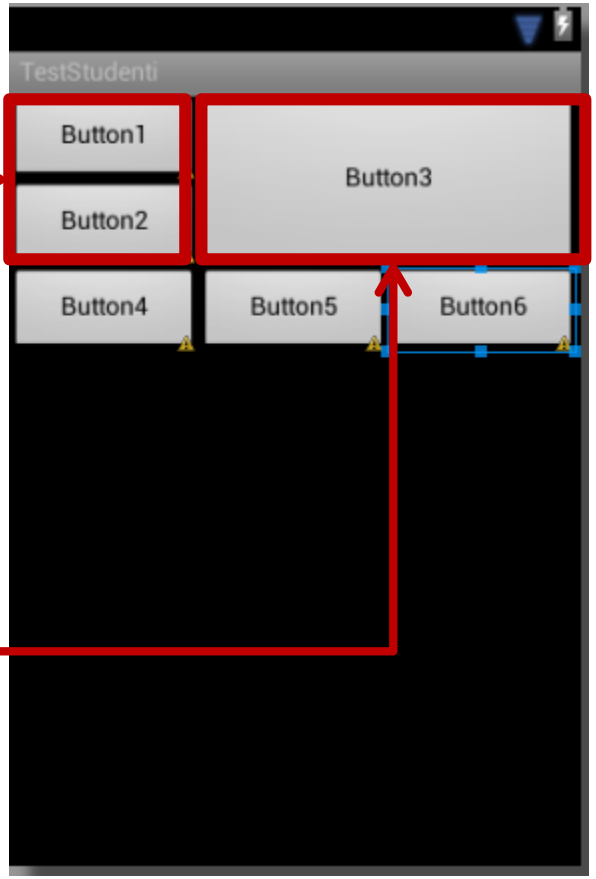


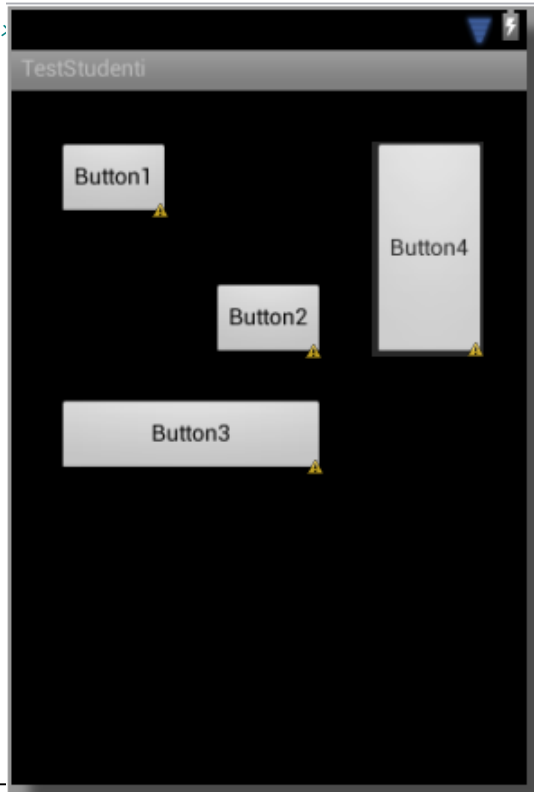
Table Layout

```
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="*" >
    <TableRow android:layout_width="fill_parent">
        <LinearLayout android:orientation="vertical"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
            <Button android:text="Button1"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content" />
            <Button android:text="Button2"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content" />
        </LinearLayout>
        <Button android:text="Button3“ android:layout_span="2"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"/>
     </TableRow>
    <TableRow >
        <Button android:text="Button4" />
        <Button android:text="Button5" />
        <Button android:text="Button6" />
    </TableRow>
</TableLayout>
```



Relative Layout

```
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="32dp"
        android:text="Button1" />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/button2"
        android:layout_alignTop="@+id/button1"
        android:layout_marginLeft="29dp"
        android:layout_toRightOf="@+id/button2"
        android:text="Button4" />
     <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="39dp"
        android:text="Button2" />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/button1"
        android:layout_centerVertical="true"
        android:layout_toLeftOf="@+id/button4"
        android:text="Button3" />
</RelativeLayout>
```



Relative Layout

- **XML Attributes**
  - layout_toRightOf
  - layout_toLeftOf
  - layout_above
  - layout_below
  - layout_alignLeft
  - layout_alignTop
  - layout_alignRight
  - layout_alignBottom
  - layout_alignParentLeft
  - layout_alignParentRight
  - layout_alignParentTop
  - layout_alignParentBottom
  - layout_marginLeft
  - layout_marginRight
  - layout_marginTop
  - layout_marginBottom

Android Programing

Gavrilut Dragos

Style

- Used to define a templete for other XML define objects
- Located in "**res\values-xxx\styles.xml**"

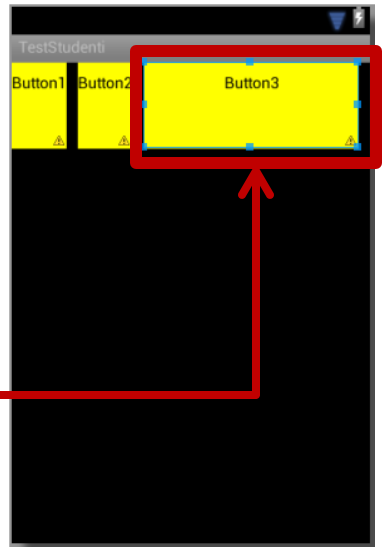
```
<resources xmlns:android="http://schemas.android.com/apk/res/android">
<style name="myStyle" parent="">
    <item name="android:background">#RGB</item>
    <item name="property_name">property_value</item>
    <item name="property_name">property_value</item>
    …
    <item name="property_name">property_value</item>
</style>
<style name="another style">
    …
</style>
…
</resources>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <Button
        style="@style/myStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1" />
</LinearLayout>
```

Style

```
<resources xmlns:android="http://schemas.android.com/apk/res/android">
<style name="myStyle">
    <item name="android:background">#FFFF00</item>
    <item name="android:foreground">#000000</item>
    <item name="android:layout_width">wrap_content</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_marginRight">10px</item>
    <item name="android:paddingTop">10px</item>
    <item name="android:paddingBottom">50px</item>
</style>
</resources>
```

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <Button style="@style/myStyle"
            android:text="Button1" />
    <Button style="@style/myStyle"
            android:text="Button2" />
    <Button style="@style/myStyle"
            android:layout_width="fill_parent"
            android:text="Button3" />

</LinearLayout>
```

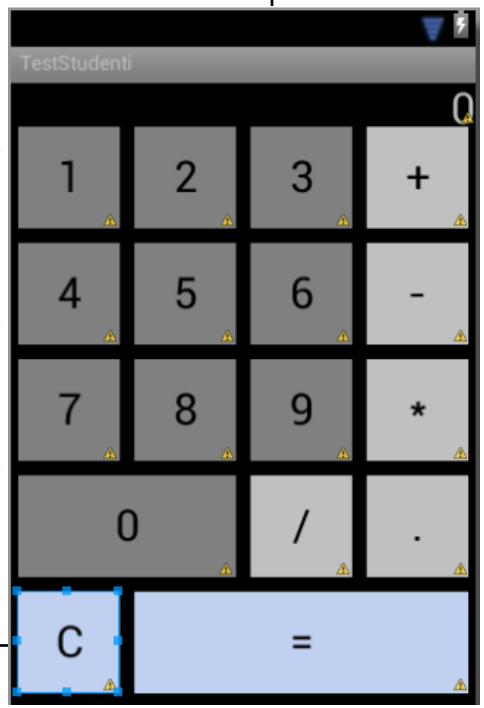


Style - Calculator

```
<resources xmlns:android="http://schemas.android.com/apk/res/android">

<style name="number">
    <item name="android:background">#808080</item>
    <item name="android:foreground">#000000</item>
    <item name="android:layout_width">70dp</item>
    <item name="android:layout_height">70dp</item>
    <item name="android:layout_marginRight">5dp</item>
    <item name="android:layout_marginLeft">5dp</item>
    <item name="android:layout_marginBottom">10dp</item>
    <item name="android:textSize">30sp</item>
</style>

<style name="operatii" parent="number">
    <item name="android:background">#C0C0C0</item>
</style>

<style name="misc" parent="number">
    <item name="android:background">#C0D0F0</item>
</style>

</resources>
```

Style - Calculator

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView android:text = "0" android:gravity="right"  android:layout_width="fill_parent"
        android:textSize="30sp"
        android:layout_height="30dp"/>

    <TableLayout android:layout_width="match_parent" android:layout_height="match_parent" >
        <TableRow android:layout_width="wrap_content" android:gravity="center_horizontal">
            <Button style="@style/number"  android:text="1" />
            <Button style="@style/number"  android:text="2" />
            <Button style="@style/number"  android:text="3" />
            <Button style="@style/operatii"  android:text="+" />
        </TableRow>
        <TableRow android:layout_width="wrap_content" android:gravity="center_horizontal">
            <Button style="@style/number"  android:text="4" />
            <Button style="@style/number"  android:text="5" />
            <Button style="@style/number"  android:text="6" />
            <Button style="@style/operatii"  android:text="-" />
        </TableRow>
        <TableRow android:layout_width="wrap_content" android:gravity="center_horizontal">
            <Button style="@style/number"  android:text="7" />
            <Button style="@style/number"  android:text="8" />
            <Button style="@style/number"  android:text="9" />
            <Button style="@style/operatii"  android:text="*" />
        </TableRow>
        <TableRow android:layout_width="wrap_content" android:gravity="center_horizontal">
            <Button style="@style/number"  android:text="0" android:layout_span="2"/>
            <Button style="@style/operatii"  android:text="/" />
            <Button style="@style/operatii"  android:text="." />
        </TableRow>
        <TableRow android:layout_width="wrap_content" android:gravity="center_horizontal">
            <Button style="@style/misc"  android:text="C"/>
            <Button style="@style/misc"  android:text="=" android:layout_span="3"/>
        </TableRow>
    </TableLayout>
</LinearLayout>
```



Theme

- A style apply to the entire activity and not just some View derived objects
- Added directly in AndroidManifest.xml when declaring an Activity

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

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

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/myStyle" >
        <activity
            android:name="com.example.teststudenti.MainActivity"
            android:label="@string/app_name" >
        </activity>
    </application>
</manifest>
```

Menus

- There are 3 types of menus:
  - Options menus. These menus are associated with the applications (Triggered by the pressed of the "Menu" key (up to version 2.3) or the Menu icon from the action bar starting from 3.0)
  - Contextual menus. Usually asociated with a long-click over an View derived object
  - Popup menus

Menus

- Can be define using XML (res\menu folder)
- Usually it has one of the following items:
  - <menu> → identifies a menu containter
  - <item> → identifies a item within a menu container
  - <group> → used to define a group of several items. All of the items in a group share some properties (visibility, active state, …)

Menus

- Every item in a menu has the following XML attributes:

XML Attribute	Descripion
android:id	Item id (used by **findViewById** function)
android:onClick	Name of public method "**public void <method> (MenuItem)**"
android:orderInCategory	Specifies the position/order of the item in a menu/group
android:title	Menu text
android:titleCondensed	If menu text is too long, titleCondensed is used
android:icon	Menu icon
android:alphabeticShortcut	Alphabetic shortcut (char)
android:numericShortcut	Numeric shortcut (integer)
android:checkable	Boolean. True if the item is checkable
android:checked	Boolean. True if the item is checkd
android:visible	Boolean. True if the item is visible
android:enabled	Boolean. True if the item is enabled

Options Menu

- Create

```
@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;
}
```

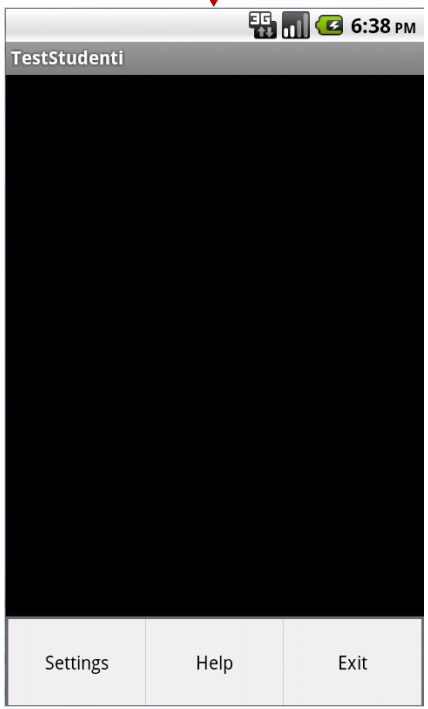
- Handle events

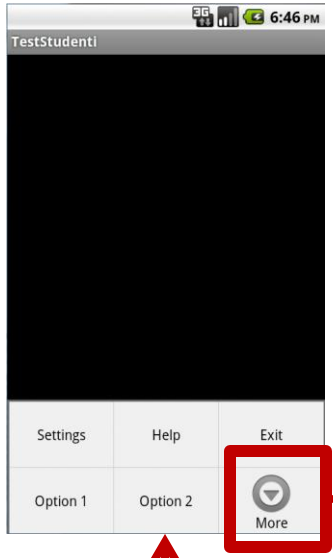
```
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    return true;
}
```

- Or specific event function for each item

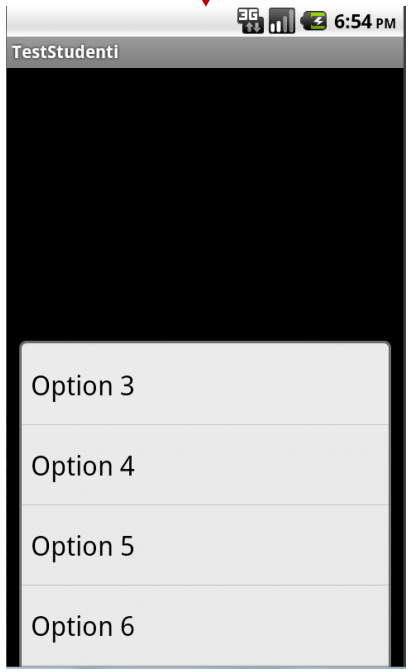
Options Menu

```
<menu>
    <item android:title="Settings"/>
    <item android:title="Help" />
    <item android:title="Exit" />
</menu>
```



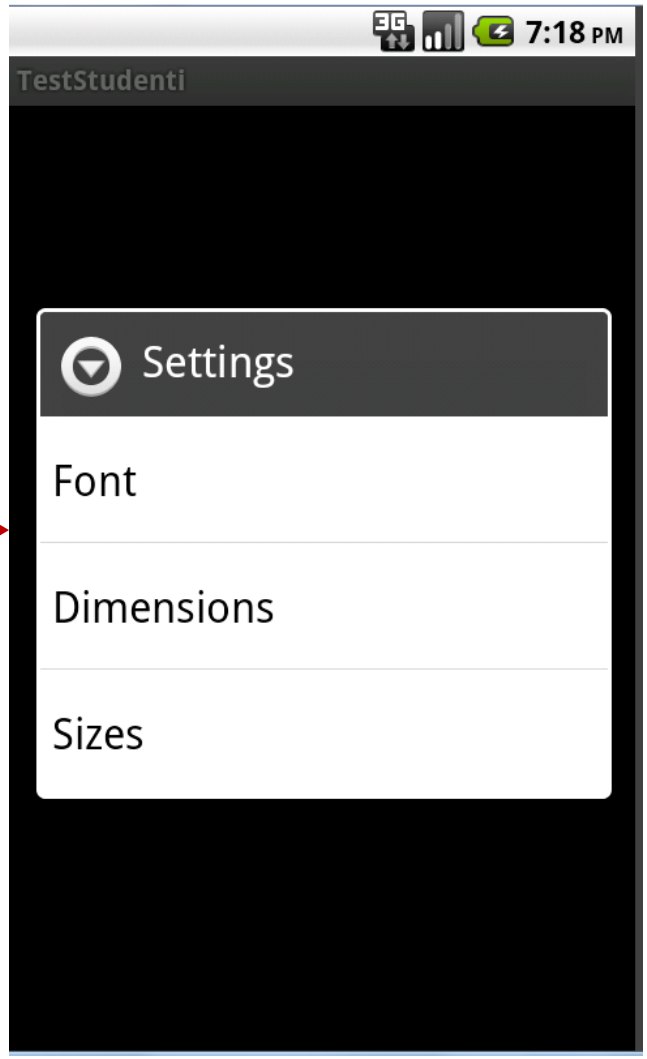



```
<menu>

    <item android:title="Settings"/>
    <item android:title="Help" />
    <item android:title="Exit" />
    <item android:title="Option 1" />
    <item android:title="Option 2" />
    <item android:title="Option 3" />
    <item android:title="Option 4" />
    <item android:title="Option 5" />
    <item android:title="Option 6" />
</menu
```



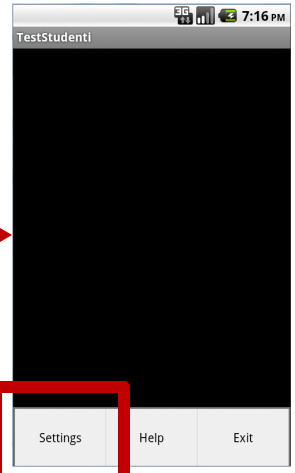
Options Menu (submenus)

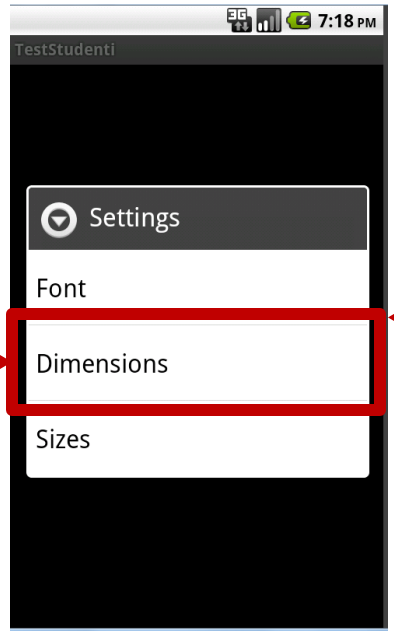
```
<menu>
    <item android:title="Settings">
        <menu >
            <item android:title="Font" />
            <item android:title="Dimensions" />
            <item android:title="Sizes" />
        </menu>
    </item>
    <item android:title="Help" />
    <item android:title="Exit" />
</menu>
```



Options Menu (submenus)

```
<menu>
    <item android:title="Settings">
        <menu>
            <item android:title="Font" />
            <item android:title="Dimensions">
                <menu>
                    <item android:title="Pixels" />
                    <item android:title="Inchs" />
                    <item android:title="Mm" />
                    <item android:title="Km" />
                    <item android:title="Wrap" />
                </menu>
            </item>
            <item android:title="Sizes" />
        </menu>
    </item>
    <item android:title="Help" />
    <item android:title="Exit" />
</menu>
```

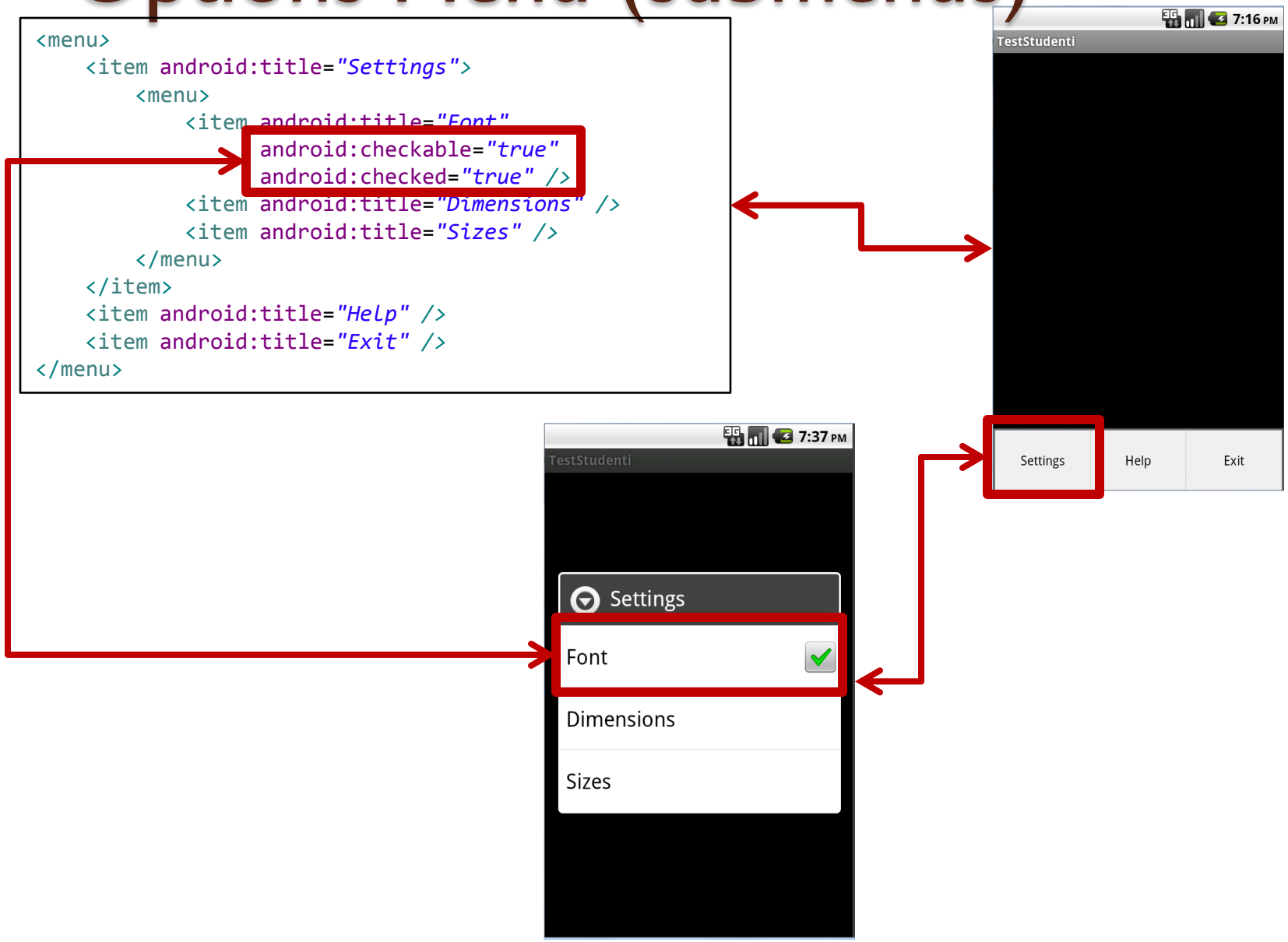




java.lang.UnsupportedOperationException:
Attempt to add a sub-menu to a sub-menu.

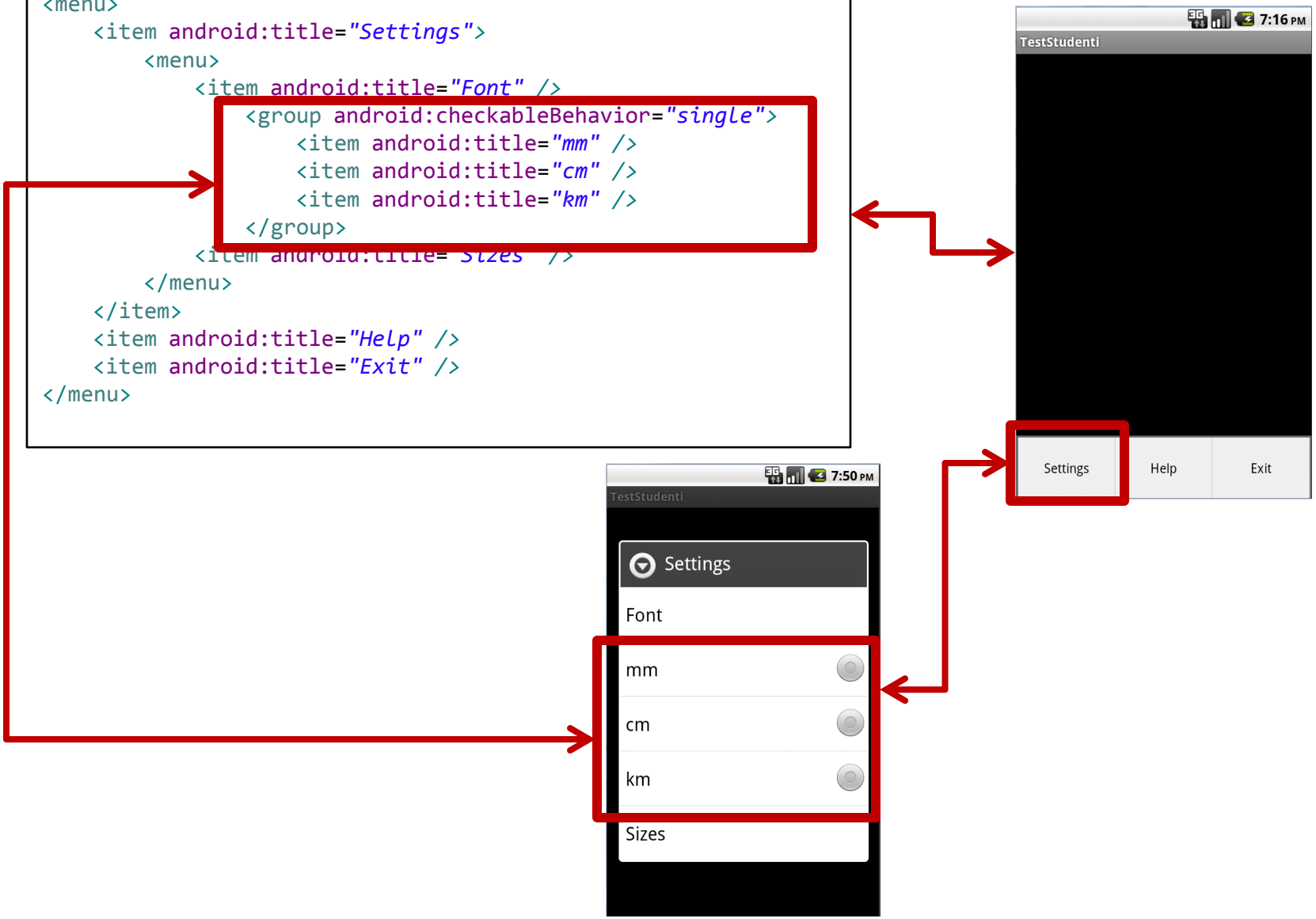
Options Menu (submenus)

```
<menu>
    <item android:title="Settings">
        <menu>
            <item android:title="Font"
                android:checkable="true"
                android:checked="true" />
            <item android:title="Dimensions" />
            <item android:title="Sizes" />
        </menu>
    </item>
    <item android:title="Help" />
    <item android:title="Exit" />
</menu>
```



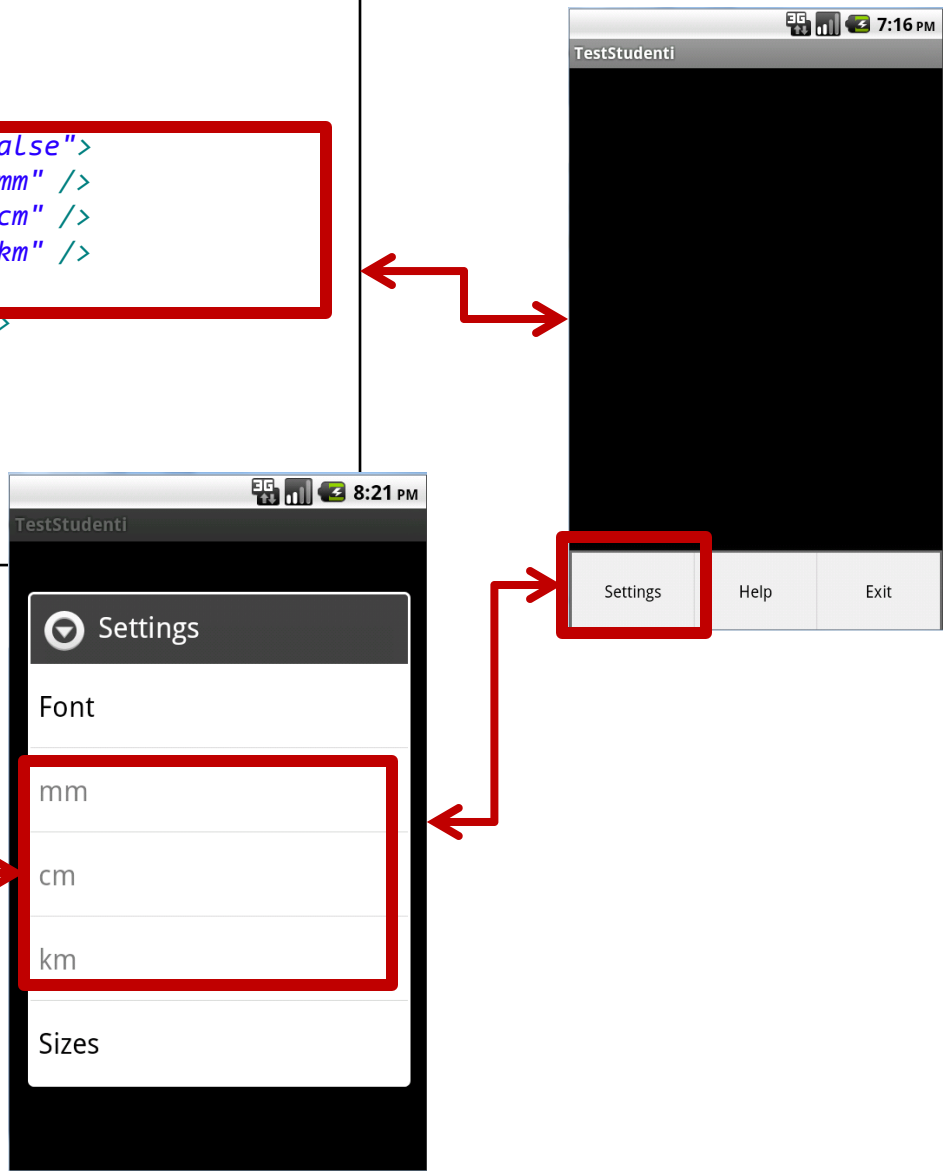
Options Menu (groups)

```
<menu>
    <item android:title="Settings">
        <menu>
            <item android:title="Font" />
                <group android:checkableBehavior="single">
                    <item android:title="mm" />
                    <item android:title="cm" />
                    <item android:title="km" />
                </group>
            <item android:title="Sizes" />
        </menu>
    </item>
    <item android:title="Help" />
    <item android:title="Exit" />
</menu>
```



Options Menu (groups)

```
<menu>
    <item android:title="Settings">
        <menu>
            <item android:title="Font" />
            <group android:enabled="false">
                <item android:title="mm" />
                <item android:title="cm" />
                <item android:title="km" />
            </group>
            <item android:title="Sizes" />
        </menu>
    </item>
    <item android:title="Help" />
    <item android:title="Exit" />
</menu>
```



Options Menu

- Handle menu events

```
<menu>
    <item
        android:id = "@+id/MenuSettings"
        android:title="Settings" />
    <item
        android:id = "@+id/MenuHelp"
        android:title="Help" />
    <item
        android:id = "@+id/MenuExit"
        android:title="Exit" />
</menu>
```

```
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) {
         getMenuInflater().inflate(R.menu.main, menu);
         return true;
     }

     @Override
     public boolean onOptionsItemSelected(MenuItem item)
     {
         switch (item.getItemId())
         {
             case R.id.MenuSettings:
                 Log.v("tag","settings pressed");
                 break;
             case R.id.MenuHelp:
                 Log.v("tag","help pressed");
                 break;
             case R.id.MenuExit:
                 Log.v("tag","exit pressed");
                 break;
         }
         return true;
     }
}
```

Options Menu

- Handle menu events

```
<menu>
    <item
        android:id = "@+id/MenuSettings"
        android:title="Settings" />
    <item
        android:id = "@+id/MenuHelp"
        android:title="Help" />
    <item
        android:id = "@+id/MenuExit"
        android:onClick="OnExit"
        android:title="Exit" />
</menu>
```

```
public class MainActivity extends Activity {

    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        switch (item.getItemId())
        {
            case R.id.MenuSettings:
                Log.v("tag","settings pressed");
                break;
            case R.id.MenuHelp:
                Log.v("tag","help pressed");
                break;
            case R.id.MenuExit:
                Log.v("tag","exit pressed");
                break;
        }
        return true;
    }
    public boolean OnExit(MenuItem item)
    {
        Log.v("tag","OnExit pressed");
        return true;
    }
}
```

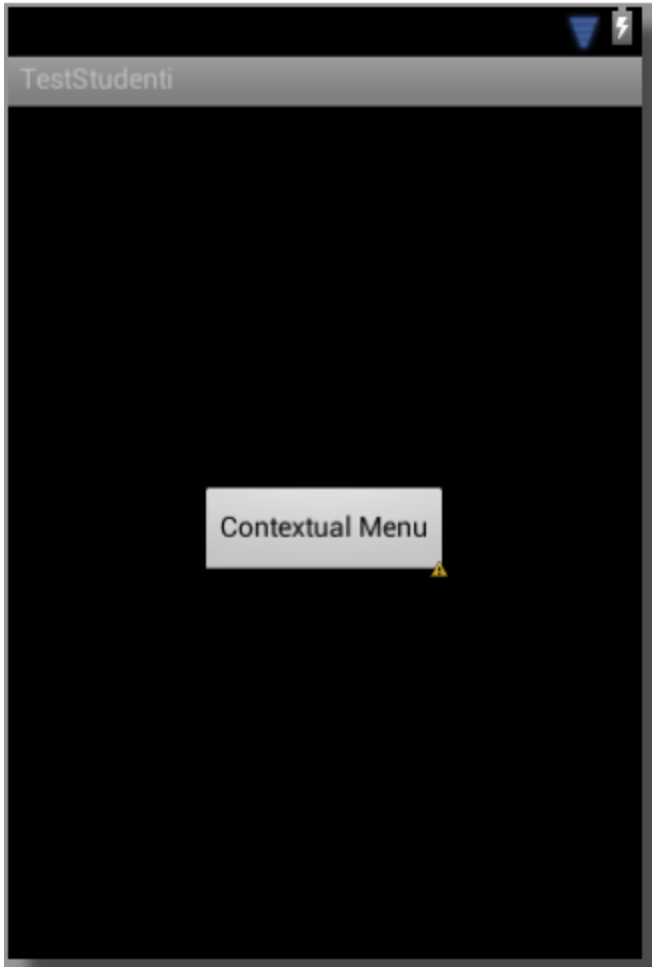
API Level ≥ 11

Contextual Menus

- Menus associated with a View derived object
- To create a contextual menu 3 steps must be performed
  1. Register a View derived object to receive a contextual menu (using registerForContextMenu api function)
  2. Overwrite onCreateContextMenu function (this function will be called for each registered View to set the Menu associated with it)
  3. Overwrite onContextItemSelected function to handle menu selection events

Contextual Menus - Example

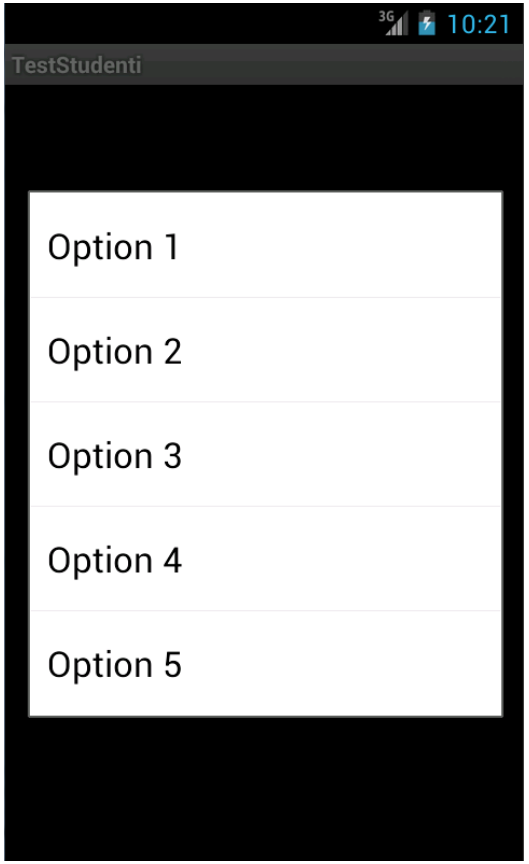
```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:gravity="center" >

    <Button
        android:id="@+id/MyButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Contextual Menu"
        android:onClick="ShowContextualMenu" />

</LinearLayout>
```



Contextual Menus - Example

- Create a new XML menu (res\menu\contextual_menu.xml)

```
<menu>
    <item
        android:id = "@+id/context_opt1"
        android:title="Option 1" />
    <item
        android:id = "@+id/context_opt2"
        android:title="Option 2" />
    <item
        android:id = "@+id/context_opt3"
        android:title="Option 3" />
    <item
        android:id = "@+id/context_opt4"
        android:title="Option 4" />
    <item
        android:id = "@+id/context_opt5"
        android:title="Option 5" />
</menu>
```



Contextual Menus - Example

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button b = (Button)findViewById(R.id.MyButton);
        registerForContextMenu(b);
    }
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);
        getMenuInflater().inflate(R.menu.contextual_menu, menu);
    }
    @Override
    public boolean onContextItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.context_opt1:
                Log.v("tag", "contextual menu 1 pressed");
                return true;
            default:
                return super.onContextItemSelected(item);
        }
    }
    public void ShowContextualMenu(View btn) {
        btn.showContextMenu();
    }
}
```

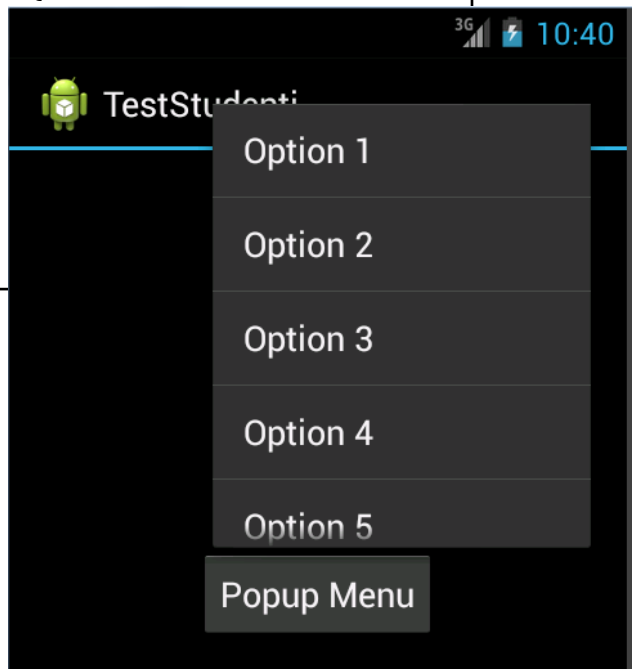
Popup menus

- Menus linked to a View (API Level ≥ 11)

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:gravity="center" >

    <Button
        android:id="@+id/MyButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Popup Menu"
        android:onClick="ShowPopupMenu" />

</LinearLayout>
```

Popup menus

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void ShowPopupMenu(View btn) {
        PopupMenu popup = new PopupMenu(this, btn);
        popup.getMenuInflater().inflate(R.menu.contextual_menu,popup.getMenu());
        popup.setOnMenuItemClickListener(new OnMenuItemClickListener() {
            @Override
            public boolean onMenuItemClick(MenuItem item) {
                // perform action for selected menu item
                return false;
            }
        });
        popup.show();
    }
}
```



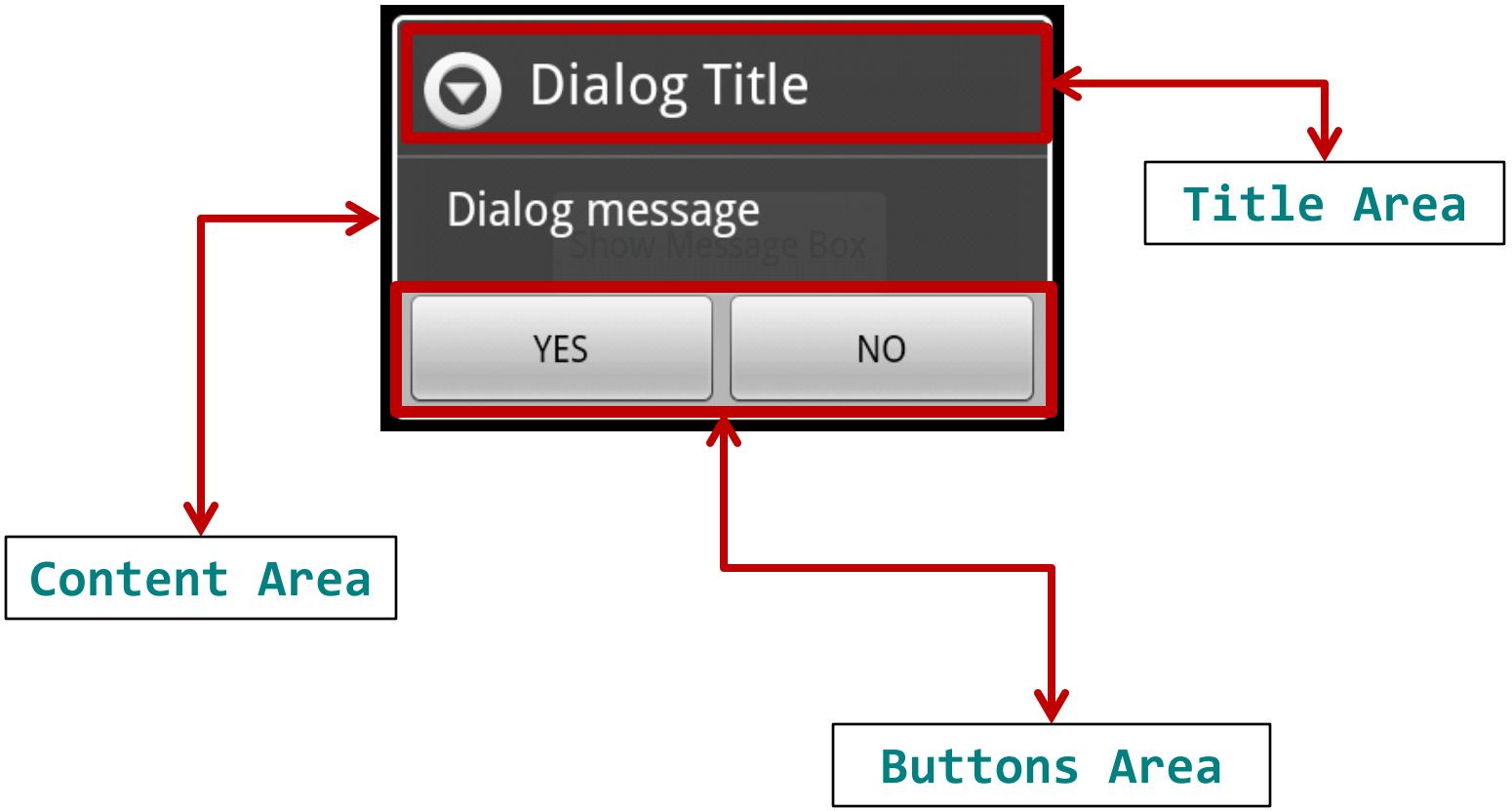
Android Programing

Gavrilut Dragos

AlertDialog

- This is the equivalent of a MessageBox in Desktop based systems
- Besides the normal behavior (show a popup message with some bottoms), an AlertDialog can be customed to look and behave like a modal window.
- Every AlertDialog can be canceled by pressing the Back key (this is the default behavior)

AlertDialog - layout



AlertDialog – builder functions

Function	Description
setTitle	Sets the title of the dialog
setMessage	Sets the message that is display in the dialog
setIcon	Sets the icon of the dialog
setPositiveButton	Sets the positive (YES / OK / AGREE) button of the dialog
setNegativeButton	Sets the negative (CANCEL / NO) button of the dialog
setNeutralButton	Sets the 3rd button of the dialog (usually called the neutral button) for the cases when a 3 options dialog is requaired
setCustomTitle	Sets the title using a custom view
setItems	Sets a list of items to be display in the dialog.
setMultiChoiceItems	Sets a list of items from where multiple items can be selected
setSingleChoiceItems	Sets a list of items from only one item can be selected
setView	Sets a custom view for the content of the dialog

AlertDialog - Example

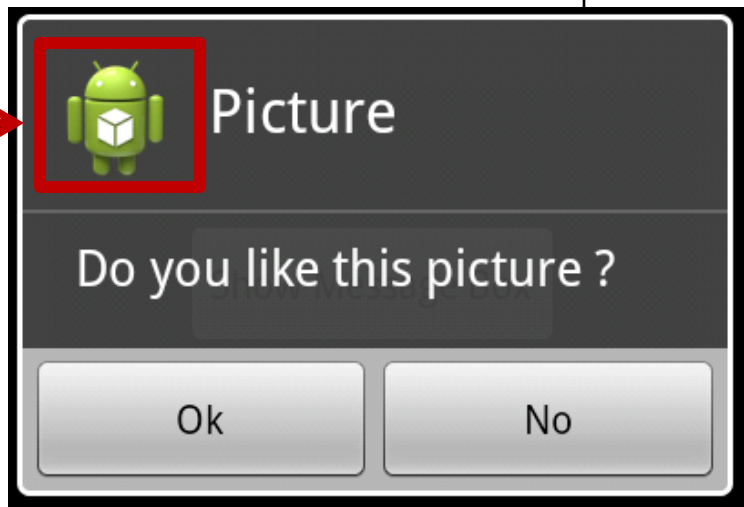
```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:gravity="center" >

    <Button
        android:id="@+id/MyButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" Show Message Box "
        android:onClick="ShowMessageBox" />

</LinearLayout>
```

AlertDialog - Example

```
public void ShowMessageBox(View btn)
{
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Dialog Title");
    builder.setMessage("Dialog message");
    builder.setPositiveButton("YES", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {

        }
    });
    builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {

        }
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

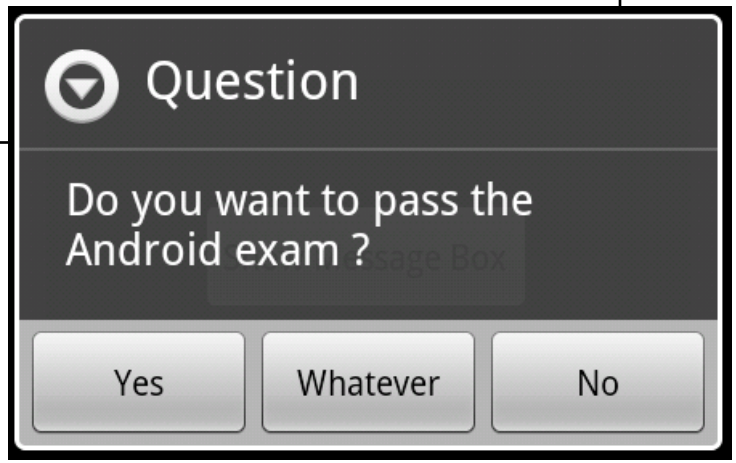
AlertDialog - Example

```
public void ShowMessageBox(View btn)
{
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Picture");
    builder.setMessage("Do you like this picture ?");

    builder.setIcon(getResources().getDrawable(R.drawable.ic_launcher));

    builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```



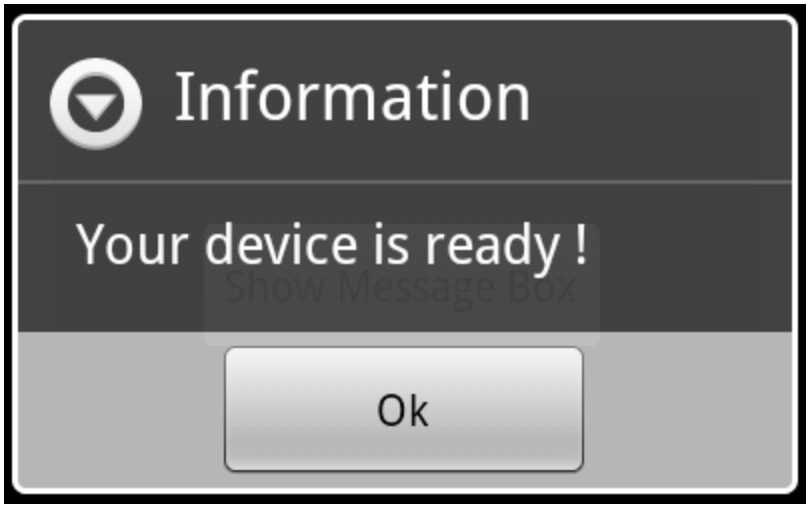
AlertDialog - Example

```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Question");
    builder.setMessage("Do you want to pass the Android exam ?");
    builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    builder.setNeutralButton("Whatever", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```



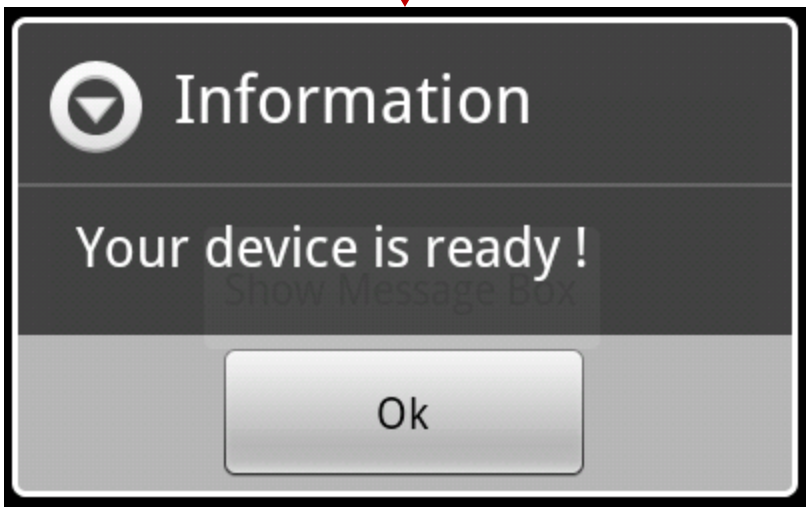
AlertDialog - Example

```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Information");
    builder.setMessage("Your device is ready !");
    builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```



AlertDialog - Example

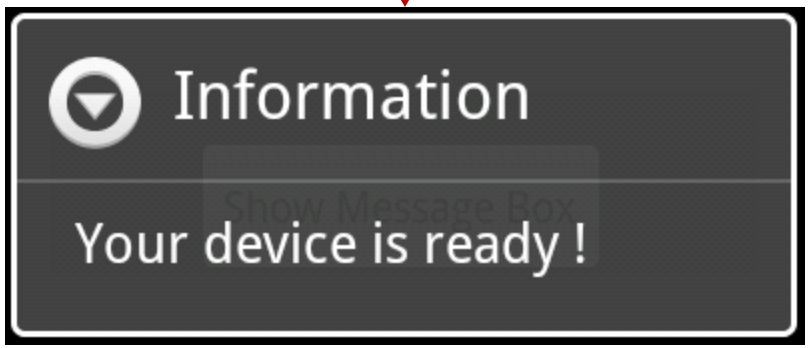
```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Information");
    builder.setMessage("Your device is ready !");
    builder.setNegativeButton("Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```



AlertDialog - Example

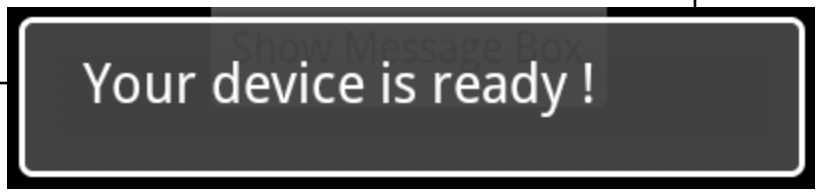
```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Information");
    builder.setMessage("Your device is ready !");
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

Hit "**BACK**" key to exit this dialog



AlertDialog - Example

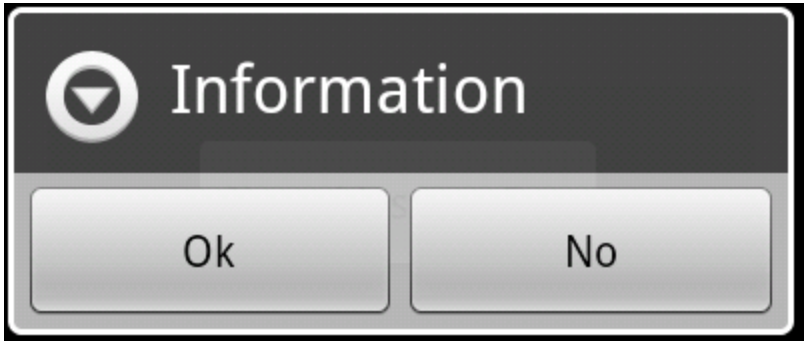
```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("Your device is ready !");
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```




```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Information");
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

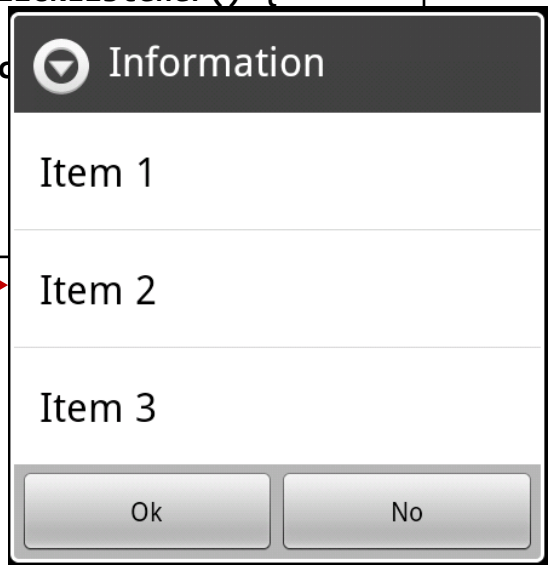
AlertDialog - Example

```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(“Information");
    builder.setPositiveButton(“Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```



AlertDialog – Example - Items

```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Information");
    builder.setItems(new CharSequence[] {"Item 1","Item 2","Item 3"},
                     new DialogInterface.OnClickListener() {
                         @Override
                         public void onClick(DialogInterface dialog, int which) {
                         }
    });
    builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int whic
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```



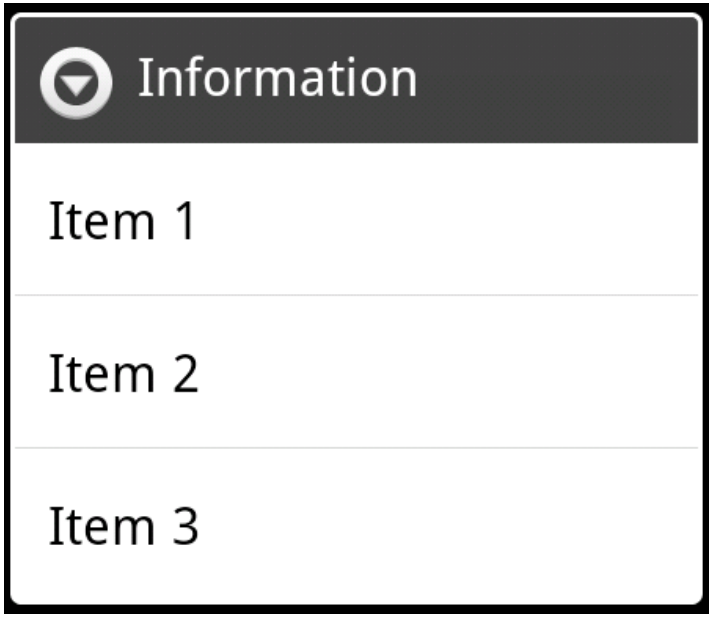
Clicking one item will close the alert dialog

IT IS NOT OK TO HAVE BUTTONS AND ITEMS AT THE SAME TIME

AlertDialog – Example - Items

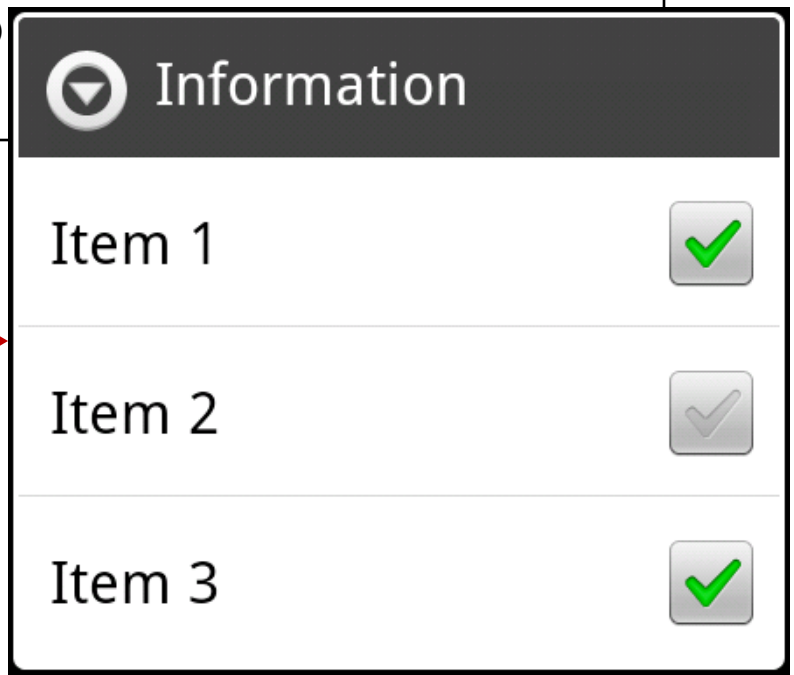
```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Information");
    builder.setItems(new CharSequence[] {"Item 1","Item 2","Item 3"},
                     new DialogInterface.OnClickListener() {
                         @Override
                         public void onClick(DialogInterface dialog, int which) {
                         }
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

The "**which**" parameters refers to the index of the item that was clicked. For example if you click on the second item ("Item 2") the "**which**" parameter will be 1 (as the list is 0-indexed).



AlertDialog – Example - Items

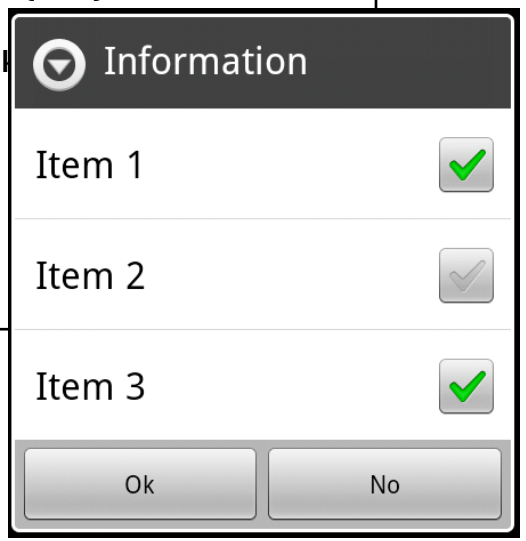
```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Information");
    builder.setMultiChoiceItems(new CharSequence[] {"Item 1","Item 2","Item 3"},
                                new boolean[] {true,false,true},
                                new DialogInterface.OnMultiChoiceClickListener() {
                                    @Override
                                    public void onClick(DialogInterface dialog,
                                                        int which,
                                                        boolean isChecked) { … }
                                }
    );
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```



IT IS NOT OK TO NOT HAVE BUTTONS FOR CHECKD ITEMS !

AlertDialog – Example - Items

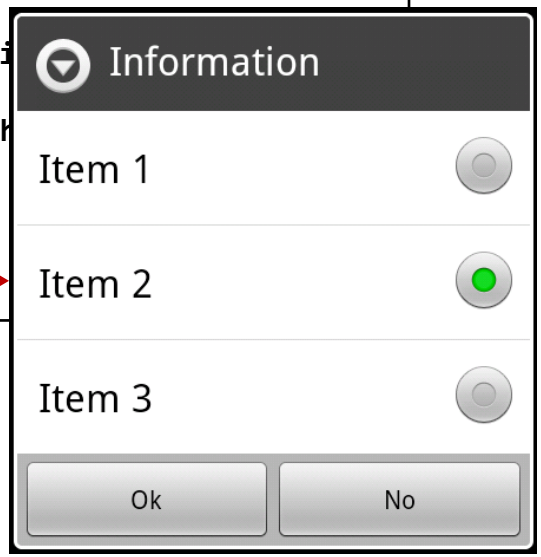
```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Information");
    builder.setMultiChoiceItems(new CharSequence[] {"Item 1","Item 2","Item 3"},
                                new boolean[] {true,false,true},
                                new DialogInterface.OnMultiChoiceClickListener() {
                                    @Override
                                    public void onClick(DialogInterface dialog,
                                                        int which,
                                                        boolean isChecked) { … }
                                }
    );
    builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClick
        @Override
        public void onClick(DialogInterface dialog, int which)
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```



AlertDialog – Example - Items

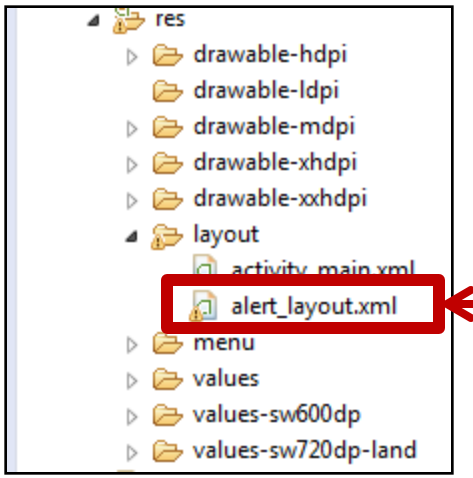
```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Information");
    builder.setSingleChoiceItems(new CharSequence[] {"Item 1","Item 2","Item 3"},
                                 1,
                                 new DialogInterface.OnClickListener() {
                                     @Override
                                     public void onClick(DialogInterface dialog,
                                                         int which) { … }
                                 }
    );
    builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    builder.setNegativeButton("No", new DialogInterface.OnCli
        @Override
        public void onClick(DialogInterface dialog, int which
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

The second parameter refers to the initial item that should be checked in the a 0-based list. A value of "-1" means that no item should be check by default.



Custom Alert Dialog

- Create a new layout with the content that will be used on the custom alert dialog

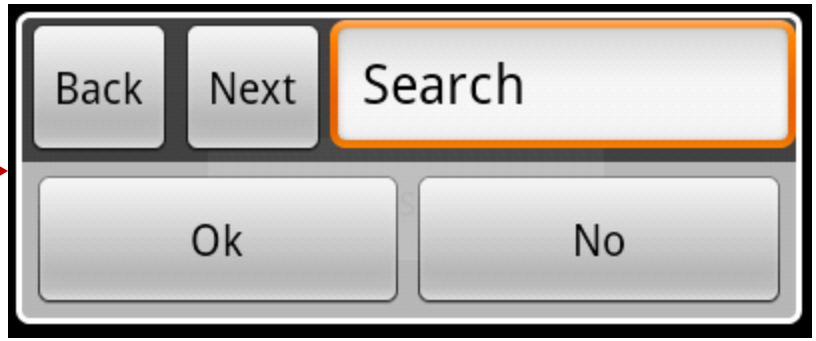



```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Back"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Next"/>
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Search" />

</LinearLayout>
```

Custom Alert Dialog

```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);

    LayoutInflater inflater = this.getLayoutInflater();
    builder.setCustomTitle(inflater.inflate(R.layout.alert_layout,null));

    builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

The **LayoutInflater** object can help you convert any layout into a **View** object and use it in a Alert Dialog



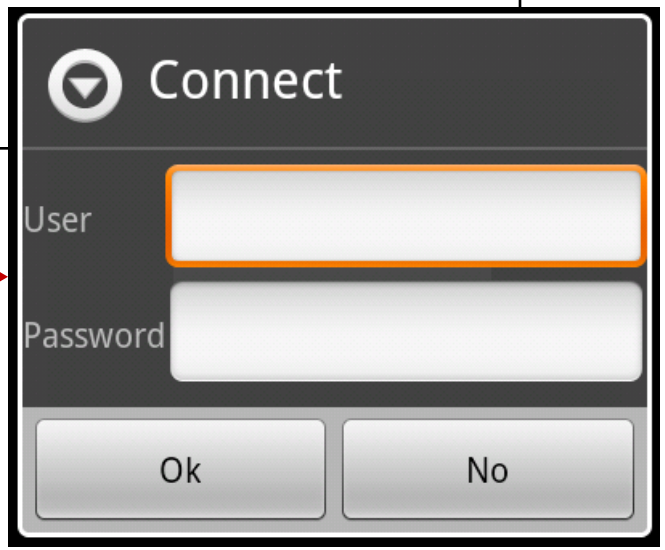
Custom Alert Dialog

```
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="match_parent" >

    <TableRow android:layout_width="fill_parent" android:layout_height="match_parent" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="User" />
        <EditText
            android:layout_width="200dp"
            android:layout_height="wrap_content" />
    </TableRow>

    <TableRow android:layout_width="fill_parent" android:layout_height="match_parent" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Password" />
        <EditText
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword" />
    </TableRow>

</TableLayout>
```

Custom Alert Dialog

```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Connect");

    LayoutInflater inflater = this.getLayoutInflater();
    builder.setView(inflater.inflate(R.layout.alert_layout,null));

    builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

Use **builder.setView** to set a new cutom view to the content area of an Alert Dialog



Custom Alert Dialog

```
public void ShowMessageBox(View btn) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Connect");

    builder.setCancelable(false);

    LayoutInflater inflater = this.getLayoutInflater();
    builder.setView(inflater.inflate(R.layout.alert_layout,null));

    builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { … }
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

By default, every Alert Dialog is cancelable. This means that hitting the "**Back**" key will close that dialog. Use **builder.setCancelable** to overwrite this behavior.

Toast

- A "Toast" is a short message that is shown to the user (an informative message).
- Create:

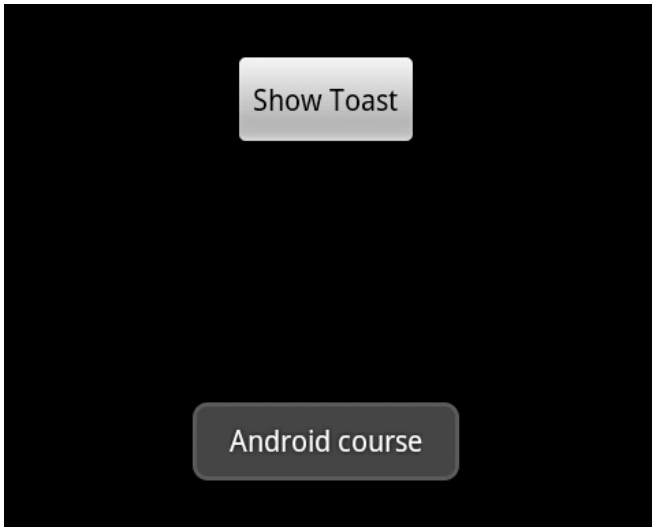
Toast toast = Toast.makeText(context, text, duration);

Where:

1. "context" is the context of the current Activity
2. "text" is the text that should be displays
3. "duration" is one of the following
   - Toast. LENGTH_LONG
   - Toast. LENGTH_SHORT

Toast - Example

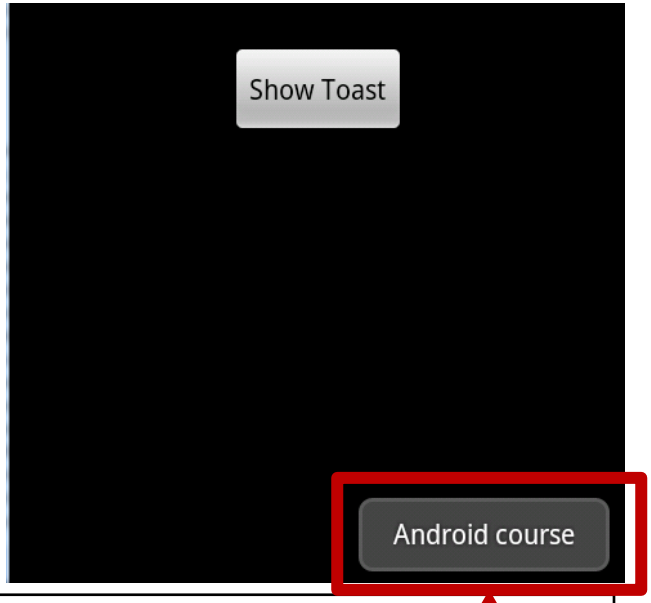
```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:gravity="center" >

    <Button
        android:id="@+id/MyButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" Show Toast"
        android:onClick="ShowToast" />

</LinearLayout>
```




```
public void ShowToast(View btn)
{
    Toast toast = Toast.makeText(this, "Android course", Toast.LENGTH_SHORT );
    toast.show();
}
```

Toast – Example (position)

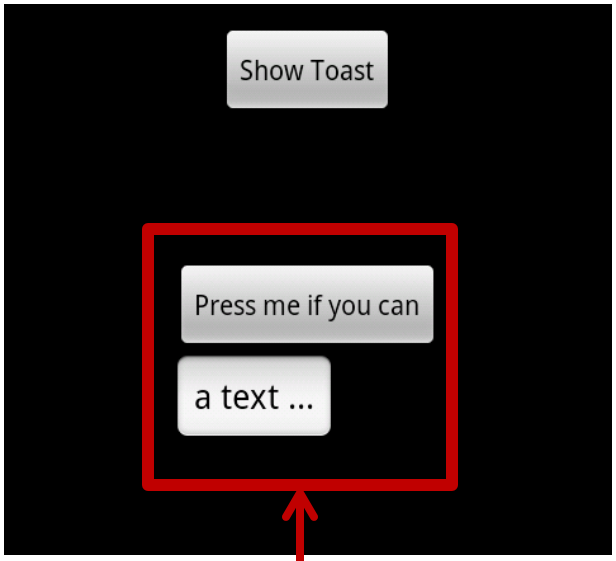
```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:gravity="center" >

    <Button
        android:id="@+id/MyButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" Show Toast"
        android:onClick="ShowToast" />

</LinearLayout>
```




```
public void ShowToast(View btn)
{
    Toast toast = Toast.makeText(this, "Android course", Toast.LENGTH_SHORT );

    toast.setGravity(Gravity.BOTTOM|Gravity.RIGHT, 0, 0);

    toast.show();
}
```

Custom Toast – Example

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Press me if you can"/>
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="a text ..."/>

</LinearLayout>
```




```
public void ShowToast(View btn)
{
    Toast toast = Toast.makeText(this, "Android course", Toast.LENGTH_SHORT );

    LayoutInflater inflater = this.getLayoutInflater();
    toast.setView(inflater.inflate(R.layout.toast_layout,null));

    toast.show();
}
```

Intents

- An "Intent" is a way to communicate between applications, activities, services, receivers, and so on
- An "Intent" is an object that contains the following informations:
  - Component name → the component that should receive this intent and process it. This field (if set) is a package name of the application that should process the intent
  - Action → a string that represents the action that needs to be performed. There are some predefined actions (such as phone calls, getting a picture, …). Every application can create its own actions
  - Data → URI for the data needed by the Intent and the MIME type of the data.
  - Category → the category of the intent. This can be set to specify what kind of components can process this intent (browsers, launchers, …)
  - Extra → pairs (key,value) of extra informations send to the component that processes the Intent
  - Flags → different flags for the intent

Intents

- Create
  - **Intent** intent = new **Intent**("**<action>**")
  - **Intent** intent = new **Intent**("**<action>**", Uri)
  - **Intent** intent = new **Intent**(Context, Class)
  - **Intent** intent = new **Intent**("**<action>**", Uri, Context, Class)
- Starting an intent
  - Context.**startActivity** (Intent intent)
  - Context.**startActivity** (Intent intent, Bundle options)
  - Context.**startActivityForResult** (Intent intent, int requestCode)
  - Context.**startActivityForResult** (Intent intent, int requestCode, Bundle options)
  - Context.**startService** (Intent intent)
  - Context.**sendBroadcast** (Intent intent)

Intents

- Start a new activity from another

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        …
    }

    public void OnButtonPressed() {
        Intent i = new Intent(this, SecondaryActivity.class);
        startActivity(i);
    }
}
```

Intents

- Share data between activities

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {…}

    public void OnButtonPressed() {
        Intent i = new Intent(this, SecondaryActivity.class);
        i.putExtra("Param1", "some_text");
        startActivityForResult(i, 123);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if ((resultCode == RESULT_OK) && (requestCode == 123)) {
            if (data.hasExtra("Return1")) {
                …
            }
        }
    }

}
```

Intents

- Share data between activities

```
public class SecondaryActivity extends Activity

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

        Bundle extraInfo = getIntent().getExtras();
        String someInfo = extraInfo.getString(“Param1");
    }

    @Override
    public void finish() {
        Intent data = new Intent();
        data.putExtra("Return1", "Return value");
        setResult(RESULT_OK, data);
        super.finish();
    }
}
```

Intents

- Using the camera to take a picture

```
public class MainActivity extends Activity

    public void OnButtonPressed() {
        Intent imageIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        imageIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(new File("…")));
        startActivityForResult(imageIntent,1278);
    }

}
```

- Or a video

```
public class MainActivity extends Activity

    public void OnButtonPressed() {
        Intent imageIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
        imageIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(new File("…")));
        startActivityForResult(imageIntent,1278);
    }

}
```

Intents

- Open the market to a specific application

```
public class MainActivity extends Activity

    public void OnButtonPressed() {
        String market="market://details?id="+"com.fii.app";
        Intent marketIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(market));
        marketIntent.addFlags(   Intent.FLAG_ACTIVITY_NO_HISTORY |
                                 Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
        startActivity(marketIntent);
    }

}
```

- Open the browser to a webpage

```
public class MainActivity extends Activity

    public void OnButtonPressed() {
        Intent browser = new Intent(Intent.ACTION_VIEW, Uri.parse("www.google.com"));
        startActivity(browser);
    }

}
```

Intents

- Send a SMS

```
public class MainActivity extends Activity

    public void OnButtonPressed() {
        Uri uri = Uri.parse("smsto:0740123456");
        Intent it = new Intent(Intent.ACTION_SENDTO, uri);
        it.putExtra("sms_body", "text to send");
        startActivity(it);
    }
}
```

- Send an email

```
public class MainActivity extends Activity

    public void OnButtonPressed() {
        Intent emailIntent = new Intent(Intent.ACTION_SEND);
        emailIntent.putExtra(Intent.EXTRA_TEXT, "Email text");
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Subject");
        emailIntent.setType("application/image");
        Uri attachament = Uri.parse("file://" + filePath);
        emailIntent.putExtra(Intent.EXTRA_STREAM, attachament);
        startActivity(emailIntent);
    }
}
```

Intents

- Make a phone call

```
public class MainActivity extends Activity

    public void OnButtonPressed() {
        Intent intent = new Intent(Intent.ACTION_CALL);
        //needs: android.permission.CALL_PHONE
        intent.setData(Uri.parse("tel:0740123456"));
        startActivity(intent);
    }
}
```

- Start the phone application

```
public class MainActivity extends Activity

    public void OnButtonPressed() {
        Intent intent = new Intent(Intent.ACTION_DIAL);
        intent.setData(Uri.parse("tel:0740123456"));
        startActivity(intent);
    }
}
```

Android Programing

Gavrilut Dragos

Storage

- 5 methods that can be used for storage on Android based systems:
  - **Shared Preferences** (stores informations as a Dictionary)
  - **Internal Storage** (store data into device internal memory)
  - **External Storage** (store data into device external memory – SD-Card)
  - **SQLLite DataBases**
  - Via **network**

Shared Preferences

- Use the SharedPreferences class
- It can be obtain from an Activity in the following way:
  - Call **getSharedPreferences**() with the name of the preferences you need
  - Call **getPreferences**() if you only need one preference file for each Activity
  - Does not require any special permisions

Shared Preferences

- Values that can be written in a SharedPreferences class

Getter	Setter
getBoolean	putBoolean
getFloat	putFloat
getInt	putInt
getLong	putLong
getString	putString
getStringSet	putStringSet
getAll	

Shared Preferences - Example

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        SharedPreferences pref = getSharedPreferences("my_prefs", 0);
        int someValue = pref.getInt("my_value", 123);
        // use someValue

        setContentView(…);
    }

    protected void SavePrefs() {
        SharedPreferences pref = getSharedPreferences("my_prefs", 0);
        SharedPreferences.Editor edit = pref.edit();
        edit.putInt("my_value", 123456);
        edit.commit();
    }
}
```

/data/data/com.example.surfaceviewtest/shared_prefs ➔ my_prefs.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <int name="my_value" value="123456" />
</map>
```

Shared Preferences - Example

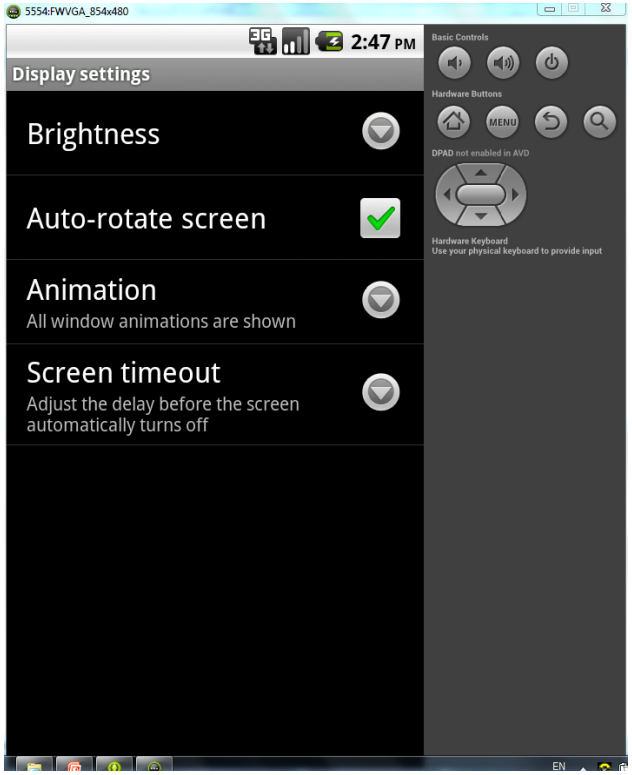
```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        SharedPreferences pref = getPreferences(0);
        int someValue = pref.getInt("my_value", 123);
        // use someValue

        setContentView(…);
    }

    protected void SavePrefs() {
        SharedPreferences pref = getPreferences(0);
        SharedPreferences.Editor edit = pref.edit();
        edit.putInt("my_value", 123456);
        edit.commit();
    }
}
```

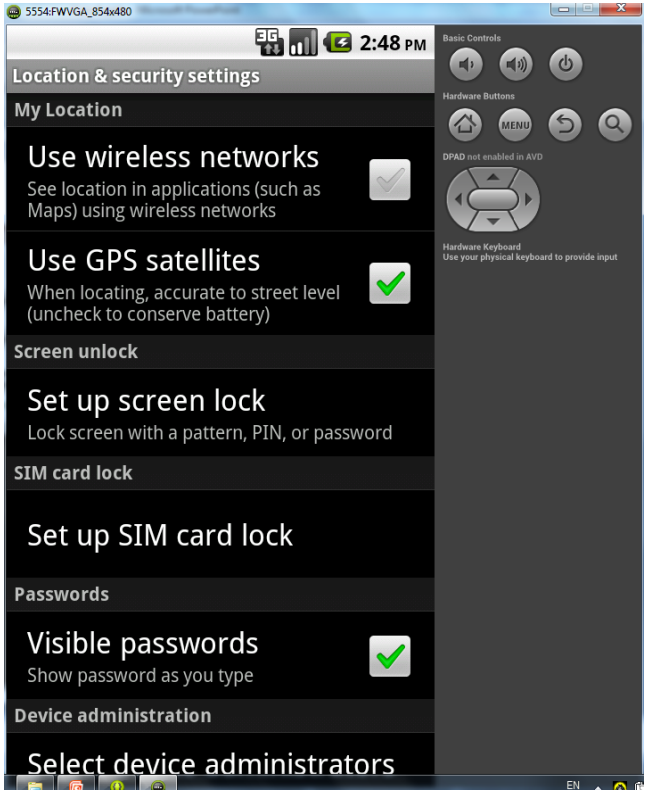
/data/data/com.example.surfaceviewtest/shared_prefs → **MainActivity.xml**

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <int name="my_value" value="123456" />
</map>
```

Shared Preferences

- Android has an option to create a special Activity desing for viewing and editing the preferences.





Shared Preferences

1. Create a new XML file that contains every item that should be modified. The XML will be located in res\xml\my_preferences.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <EditTextPreference
        android:key="my_value_2"
        android:defaultValue="123"
        android:title="Some value ..."/>


</PreferenceScreen>
```

Shared Preferences

2. Create a new activity and derive it from the PreferenceActivity class

```
public class MyPrefActivity extends PreferenceActivity{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getPreferenceManager().setSharedPreferencesName("my_prefs");
        addPreferencesFromResource(R.xml.my_preferences);
    }
}
```

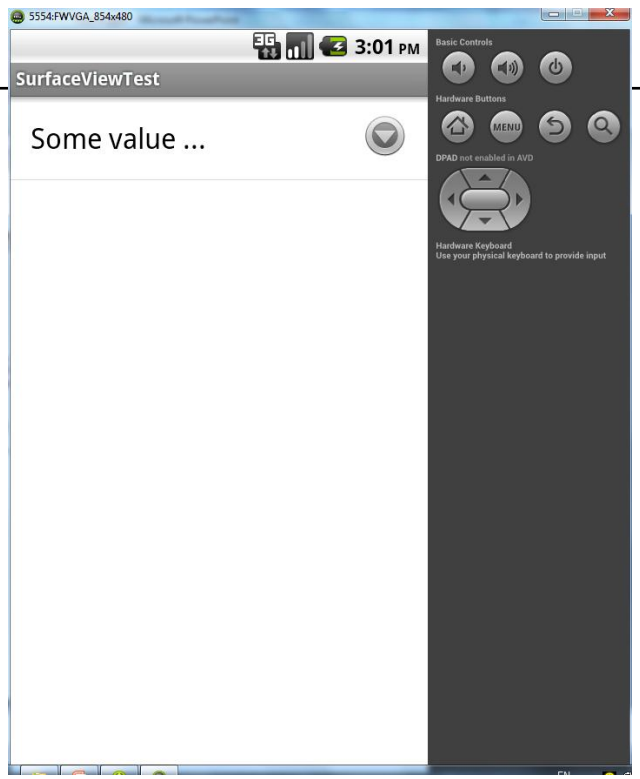
3. Add the new activity in the AndroidManifest.xml

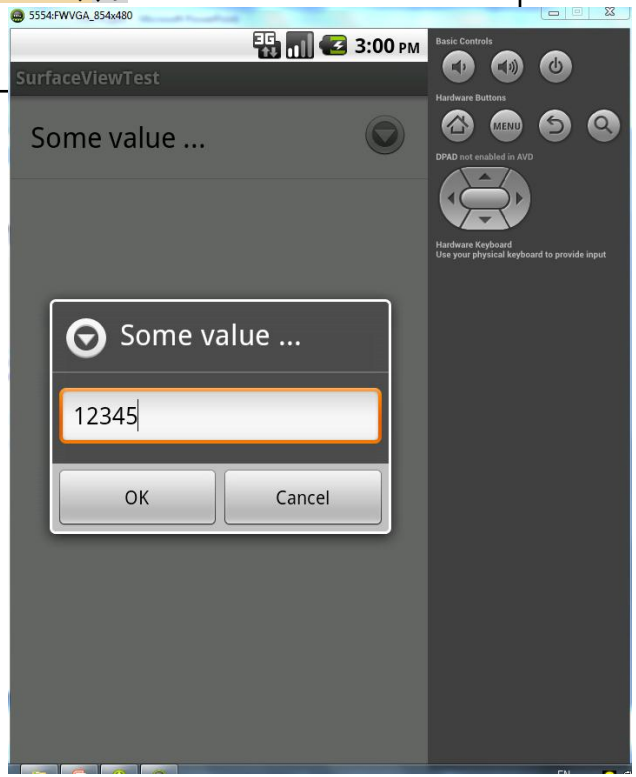
```
<activity android:name="com.example.studenti.MyPrefActivity" />
```

Shared Preferences

4. Start the Preference activity from one point in the main activity

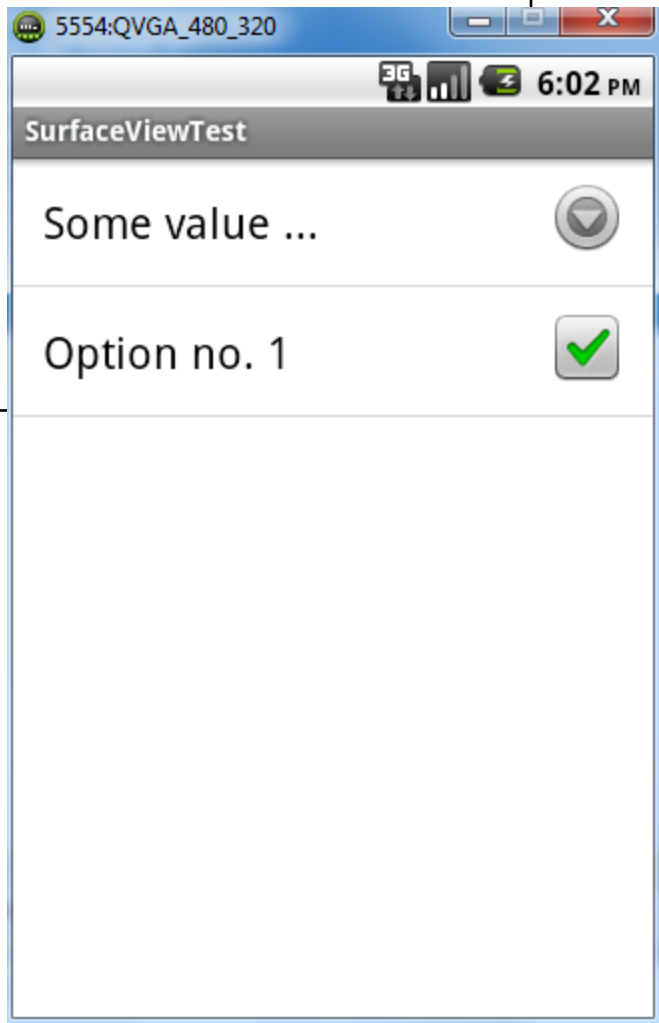
```
public class MainActivity extends Activity {
    protected void ShowPreferences() {
        startActivity(new Intent(this,MyPrefActivity.class));
    }
}
```





Shared Preferences - XML

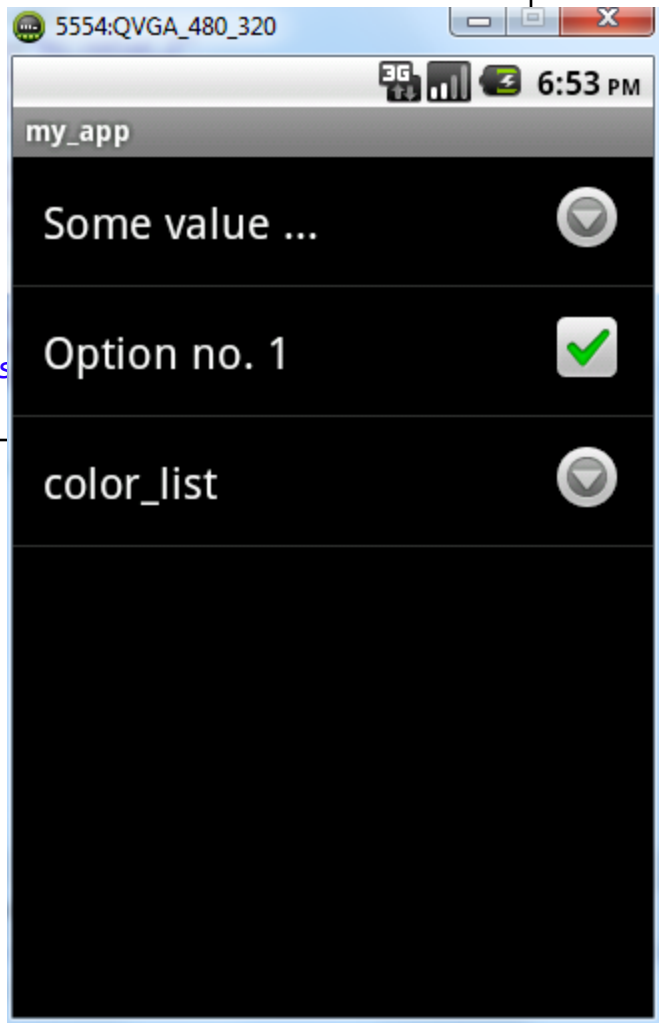
```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <EditTextPreference
        android:key="my_value_2"
        android:defaultValue="123"
        android:title="Some value ..."/>
    <CheckBoxPreference
        android:key="option_1"
        android:defaultValue="true"
        android:title="Option no. 1" />

</PreferenceScreen>
```



Shared Preferences - XML

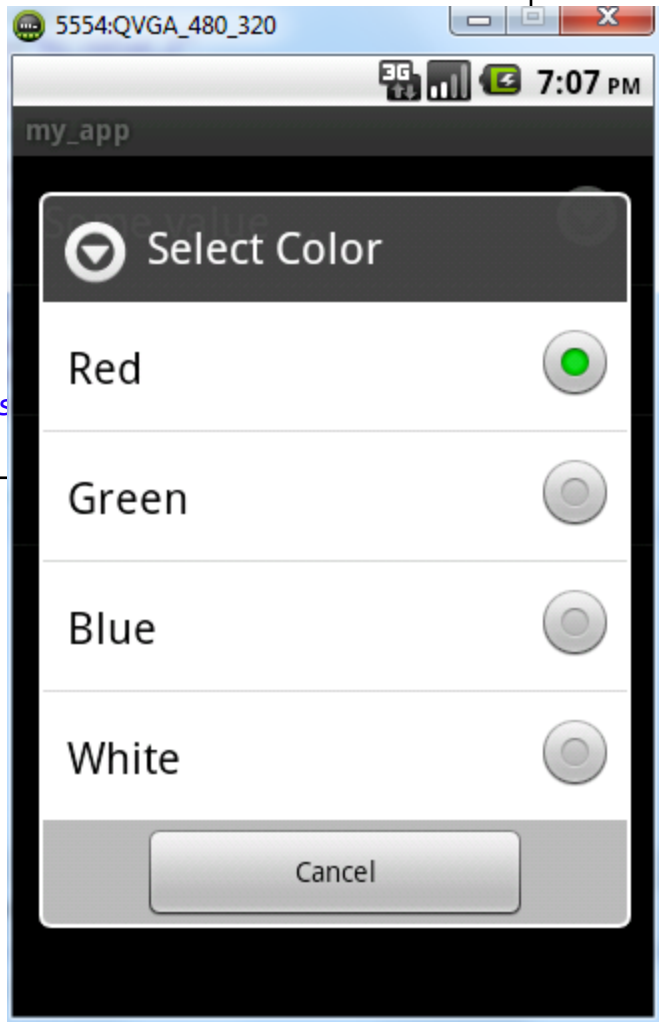
```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <EditTextPreference
        android:key="my_value_2"
        android:defaultValue="123"
        android:title="Some value ..."/>
    <CheckBoxPreference
        android:key="option_1"
        android:defaultValue="true"
        android:title="Option no. 1" />
    <ListPreference
        android:key="list_color"
        android:title="color_list"
        android:entries="@array/color_list"
        android:entryValues="@array/color_list_results
</PreferenceScreen>
```

```
<resources>
 <string-array name="color_list">
    <item>Red</item>
    <item>Green</item>
    <item>Blue</item>
    <item>White</item>
</string-array>
    <string-array name="color_list_results">
    <item>0</item>
    <item>1</item>
    <item>2</item>
    <item>3</item>
</string-array>
</resources>
```



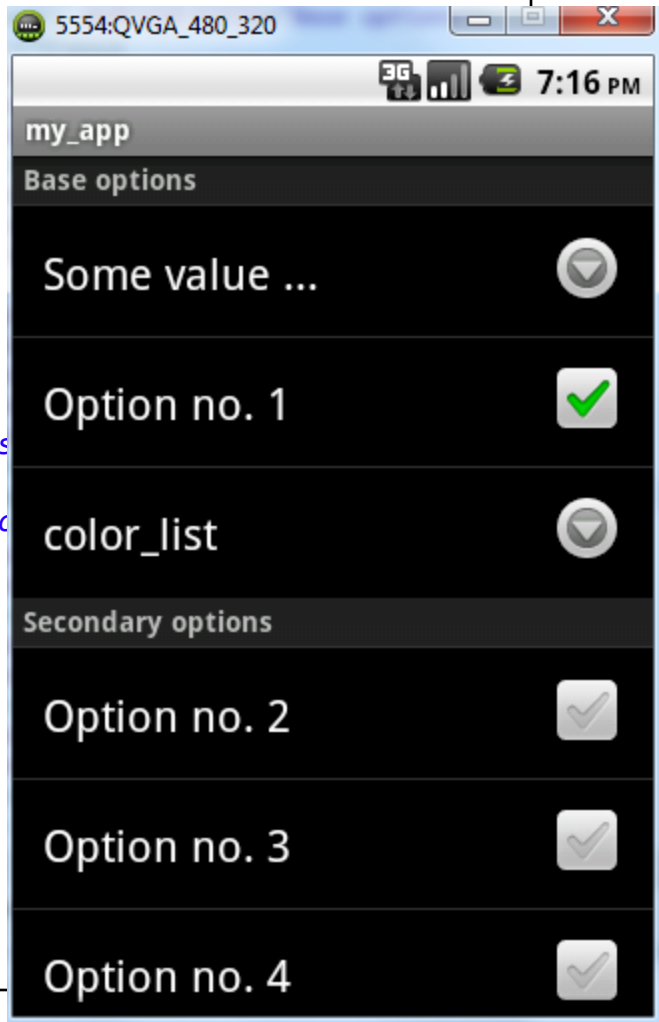
Shared Preferences - XML

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <EditTextPreference
        android:key="my_value_2"
        android:defaultValue="123"
        android:title="Some value ..."/>
    <CheckBoxPreference
        android:key="option_1"
        android:defaultValue="true"
        android:title="Option no. 1" />
    <ListPreference
        android:key="list_color"
        android:title="color_list“
        android:dialogTitle=“Select Color"
        android:entries="@array/color_list"
        android:entryValues="@array/color_list_results
</PreferenceScreen>
```



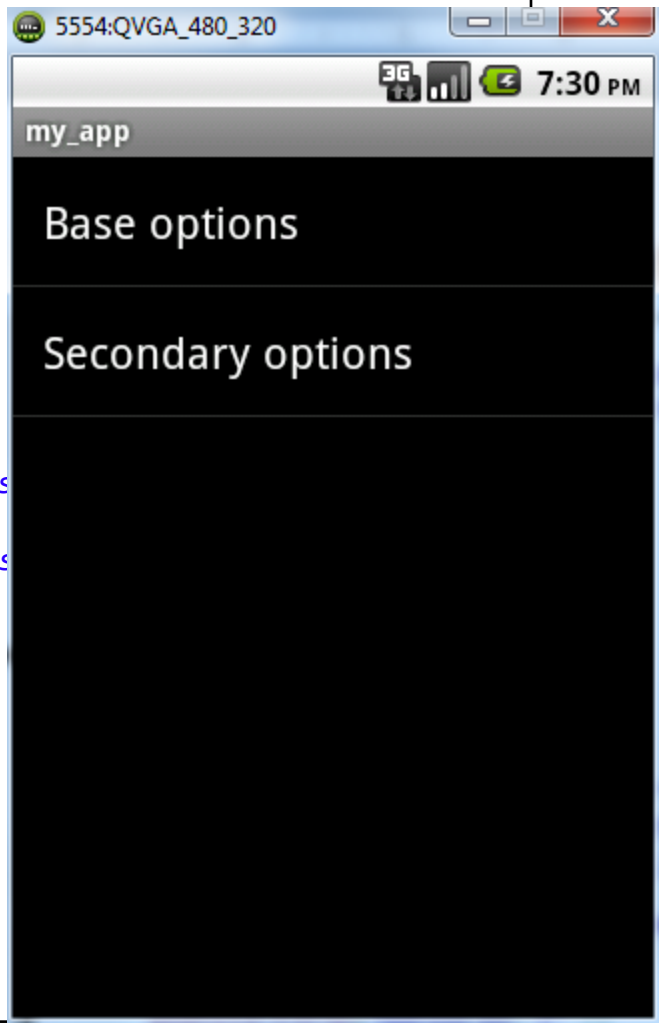
Shared Preferences - XML

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <PreferenceCategory android:title="Base options">
    <EditTextPreference
        android:key="my_value_2"
        android:defaultValue="123"
        android:title="Some value ..."/>
    <CheckBoxPreference
        android:key="option_1"
        android:defaultValue="true"
        android:title="Option no. 1" />
    <ListPreference
        android:key="list_color"
        android:title="color_list"
        android:dialogTitle="Select Color"
        android:entries="@array/color_list"
        android:entryValues="@array/color_list_results
    </PreferenceCategory>
    <PreferenceCategory android:title="Secondary optic
    <CheckBoxPreference
        android:key="option_2"
        android:title="Option no. 2" />
    <CheckBoxPreference
        android:key="option_3"
        android:title="Option no. 3" />
    <CheckBoxPreference
        android:key="option_4"
        android:title="Option no. 4" />
    </PreferenceCategory>
</PreferenceScreen>
```



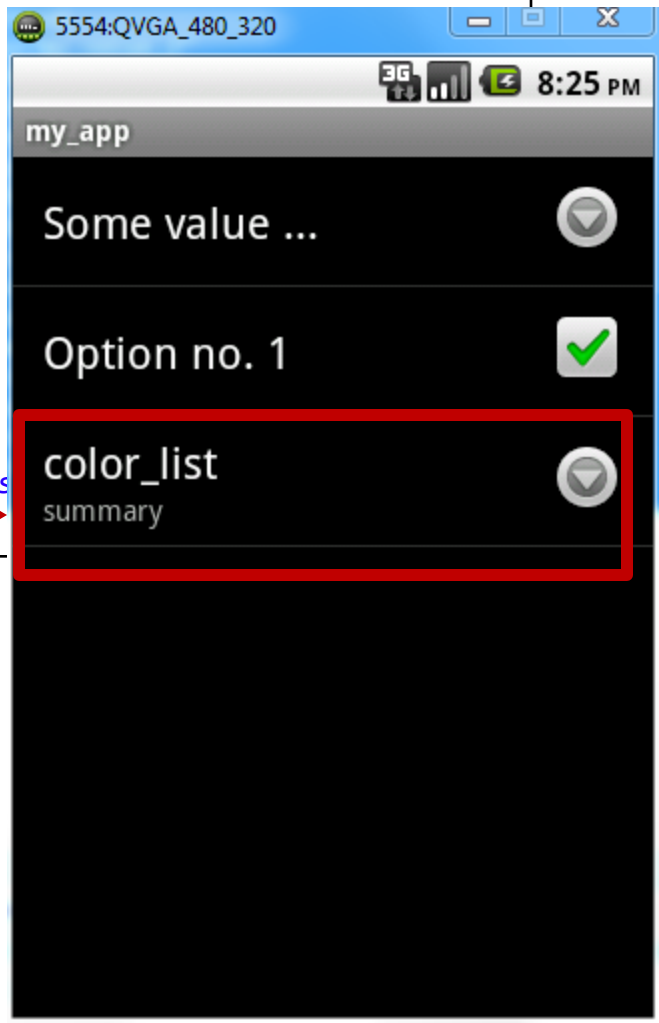
Shared Preferences - XML

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <PreferenceScreen android:title="Base options" >
    <EditTextPreference
        android:key="my_value_2"
        android:defaultValue="123"
        android:title="Some value ..."/>
    <CheckBoxPreference
        android:key="option_1"
        android:defaultValue="true"
        android:title="Option no. 1" />
    <ListPreference
        android:key="list_color"
        android:title="color_list"
        android:dialogTitle="Select Color"
        android:entries="@array/color_list"
        android:entryValues="@array/color_list_results
    </PreferenceScreen>
    <PreferenceScreen android:title="Secondary options
    <CheckBoxPreference
        android:key="option_2"
        android:title="Option no. 2" />
    <CheckBoxPreference
        android:key="option_3"
        android:title="Option no. 3" />
    <CheckBoxPreference
        android:key="option_4"
        android:title="Option no. 4" />
    </PreferenceScreen>
</PreferenceScreen>
```



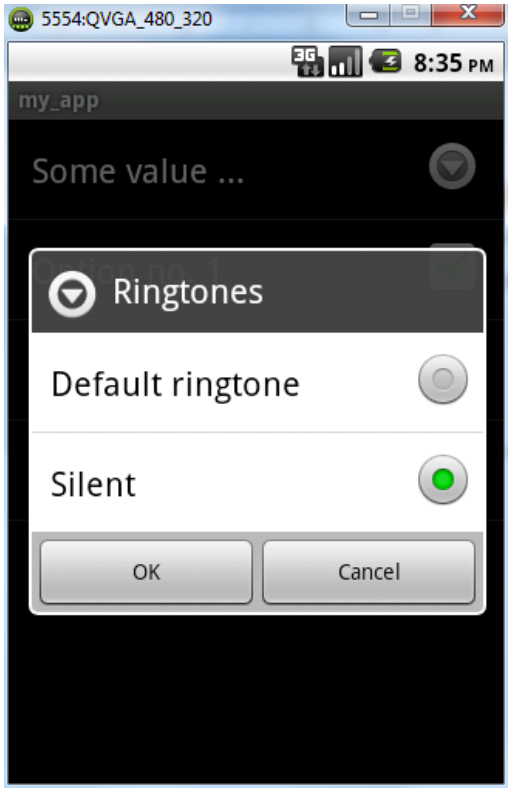
Shared Preferences - XML

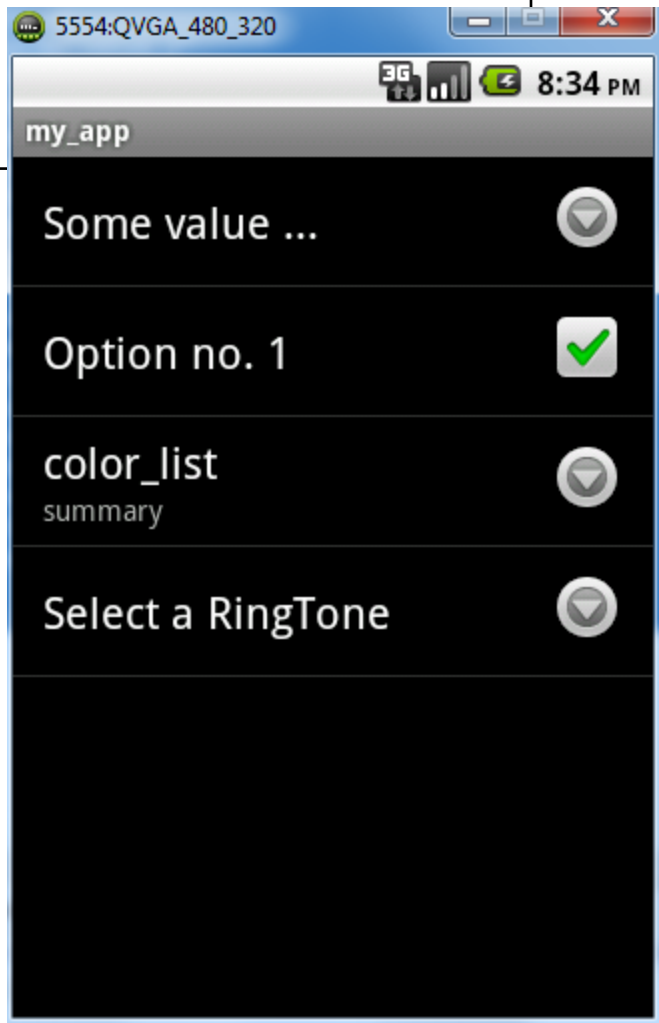
```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <EditTextPreference
        android:key="my_value_2"
        android:defaultValue="123"
        android:title="Some value ..."/>
    <CheckBoxPreference
        android:key="option_1"
        android:defaultValue="true"
        android:title="Option no. 1" />
    <ListPreference
        android:key="list_color"
        android:dialogMessage="Test message"
        android:summary="summary"
        android:title="color_list"
        android:entries="@array/color_list"
        android:entryValues="@array/color_list_results
</PreferenceScreen>
```



Shared Preferences - XML

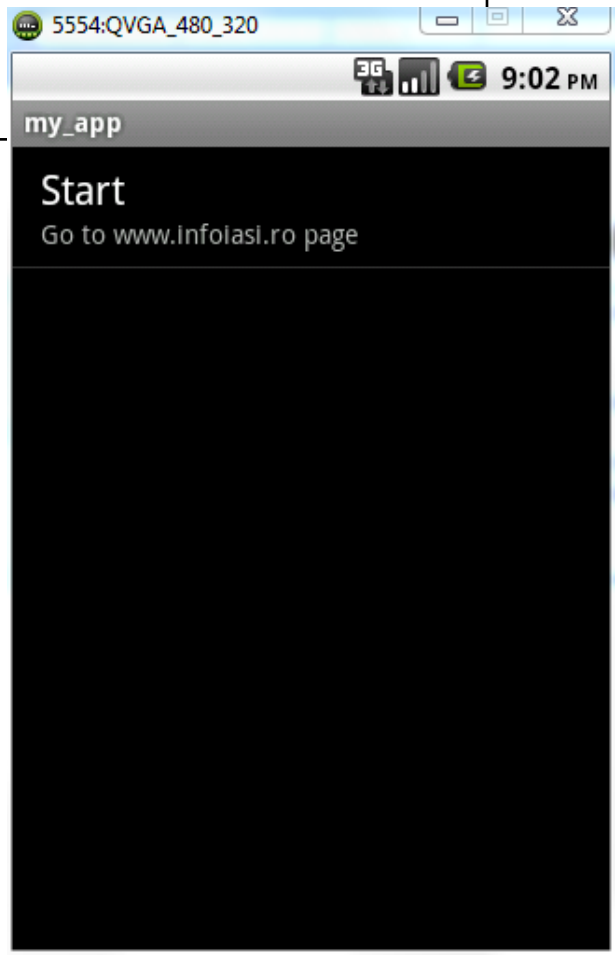
```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
…
<RingtonePreference
        android:title="Select a RingTone"
        android:ringtoneType="alarm" />

</PreferenceScreen>
```





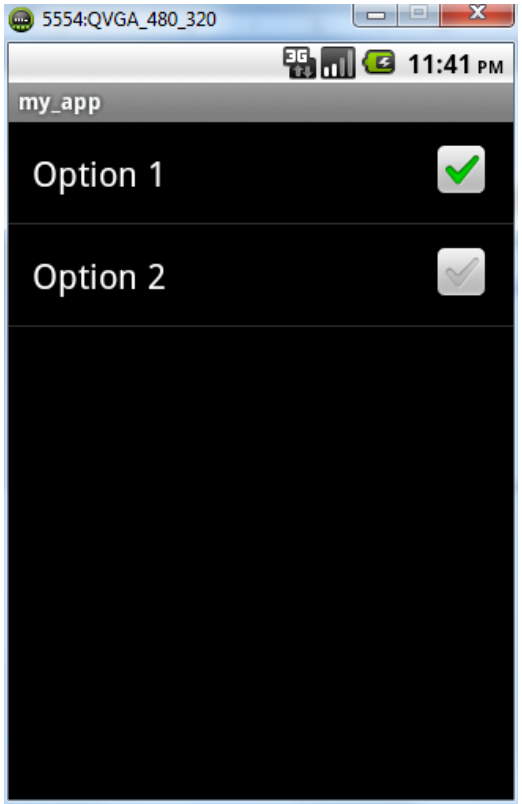
Shared Preferences - XML

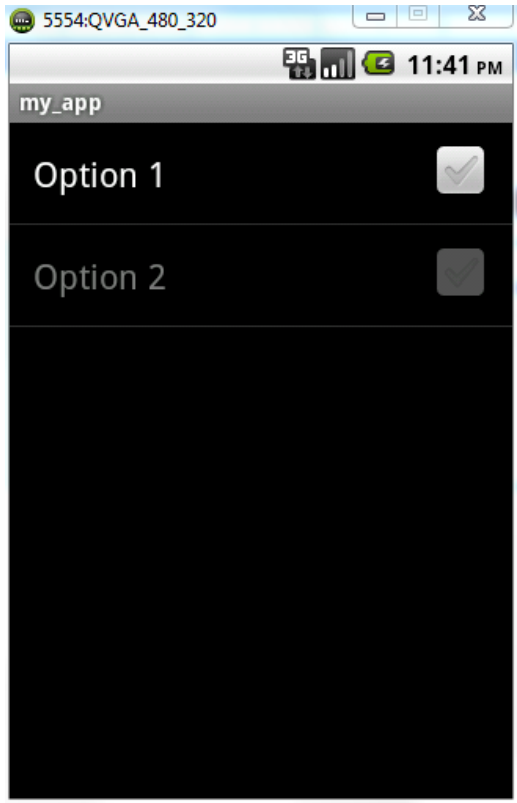
```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <Preference
        android:title="Start"
        android:summary="Go to www.infoiasi.ro page" >
        <intent
            android:action="android.intent.action.VIEW"
            android:data="http://www.infoiasi.ro" />
    </Preference>
</PreferenceScreen>
```



Shared Preferences - XML

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <CheckBoxPreference
        android:key="opt_1"
        android:title="Option 1" />
    <CheckBoxPreference
        android:title = "Option 2"
        android:dependency="opt_1" />
</PreferenceScreen>
```





Internal storage

- Represented by a location in the device internal memory
- The files saved into the internal storage can only be used by the application that saves them
- Once the application is uninstall, these files are deleted

Internal storage - Example

```
public class MainActivity extends Activity {

    protected void SaveFileToInternalStorage() {
        FileOutputStream fos;
        try {
            fos = openFileOutput("internal_file.txt", Context.MODE_PRIVATE);
            fos.write("test".getBytes());
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
/data/data/com.example.studenti/files➔ internal_file.xml
```

- Other functions associated with internal storage (members of the **Context** class)
  - openFileInput → opens a private file for reading
  - getFilesDir → returns the directory where the internal files are kept
  - getDir → to create/retrieve a directory
  - deleteFile → deletes an internal storage file

External storage

- SD-Card (requires WRITE_EXTERNAL_STORAGE permisions)
- Obtaind using: getExternalStorageDirectory, getExternalStoragePublicDirectory,
- getExternalStoragePublicDirectory can be used to obtain the location of some specials directoris such as:
  - Music
  - Podcasts
  - Ringtones
  - Alarms
  - Notifications
  - Pictures
  - Movies
  - Downloads

SQLLite databases

```
public class MainActivity extends Activity {
    private SQLiteDatabase myDB;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(…);
        OpenDB();
    }
    public void OpenDB() {
        SQLiteOpenHelper sql = new SQLiteOpenHelper(this,"my_db",null,1) {
            @Override
            public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            }
            @Override
            public void onCreate(SQLiteDatabase db) {
            }
         };
         myDB = sql.getWritableDatabase();
         myDB.execSQL("CREATE TABLE names (name TEXT)");
    }
}
```

/data/data/com.example.surfaceviewtest/databases ➔my_db

Android Programing

Gavrilut Dragos

Sensors

- Hardware components that can provide informations about motion, environment, position and so on.
- Not all of the sensors curently suported by the Android platform are availabe on all devices and not all versions of Android suports the software equavalent of the hardware components.
- A sensor can be either hardware (there is a phisical component that can measure and provide data) or software. A software sensor that gather its data from one or more hardware sensors.

Sensors

Sensor	Type	Common Uses
TYPE_ACCELEROMETER	Hardware	Motion detection (shake, tilt, etc.).
TYPE_AMBIENT_TEMPERATURE	Hardware	Monitoring air temperatures.
TYPE_GRAVITY	Software or Hardware	Motion detection (shake, tilt, etc.).
TYPE_GYROSCOPE	Hardware	Rotation detection (spin, turn, etc.).
TYPE_LIGHT	Hardware	Controlling screen brightness.
TYPE_LINEAR_ACCELERATION	Software or Hardware	Monitoring acceleration along a single axis.
TYPE_MAGNETIC_FIELD	Hardware	Creating a compass.
TYPE_ORIENTATION	Software	Determining device position.
TYPE_PRESSURE	Hardware	Monitoring air pressure changes.
TYPE_PROXIMITY	Hardware	Phone position during a call.
TYPE_RELATIVE_HUMIDITY	Hardware	Monitoring dewpoint, absolute, and relative humidity.
TYPE_ROTATION_VECTOR	Software or Hardware	Motion detection and rotation detection.
TYPE_TEMPERATURE	Hardware	Monitoring temperatures.

Sensors

- Objects that are related to sensors:
  - SensorManager
  - Sensor
  - SensorEvent
  - SensorEventListener

Sensors

- 2 type of sensors:
  - Streamimg sensors (return data at a specific time interval)
  - Non-streaming – it reports data when a change occurs in the its data.
- Use Sensor. getMinDelay() method to find out the type of sensor: 0 means non-streaming

Sensors

```
public class MainActivity extends Activity implements SensorEventListener {
    private Sensor sensorAccelerometer;
    private SensorManager sensorManager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
        sensorAccelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
         sensorManager.registerListener(this, sensorAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
    }
    protected void onResume() {
        if (sensorManager!=null)
            sensorManager.registerListener(this, sensorAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
    }
    protected void onPause() {
        if (sensorManager!=null)
            sensorManager.unregisterListener(this);
    }
    @Override
    public void onAccuracyChanged(Sensor s, int value) {
        Log.v("TAG","Accuracy Changed");
        Log.v("TAG","Sensor = "+s.getName());
        Log.v("TAG","Vendor = "+s.getVendor());
        Log.v("TAG","Version = "+String.valueOf(s.getVersion()));
        Log.v("TAG","Value = "+String.valueOf(value));
    }
    @Override
    public void onSensorChanged(SensorEvent s) {
       Log.v("TAG","Sensor Changed");
       Log.v("TAG","Sensor = "+s.sensor.getName());
       String values = "";
       for (int tr=0;tr<s.values.length;tr++)
           values+=String.format("%f,", s.values[tr]);
       Log.v("TAG","Values = "+values);
    }
```

Sensors

- Result (Samsung Galaxy Tablet 10.1)
  - Accuracy Changed
    - Sensor = MPL accel
    - Vendor = Invensense
    - Version = 1
  - Sensor Changed
    - Sensor = MPL accel
    - Values = 0.000000,-0.775986,9.886641,
- Values field holds the data that is specific for each sensor

Sensor values

- Sensor.TYPE_ACCELEROMETER
  - 3 values measured in SI units
- Sensor.TYPE_MAGNETIC_FIELD:
  - 3 values measured in micro-Tesla units
- Sensor.TYPE_GYROSCOPE
  - 3 values measured in radians/second
- Sensor.TYPE_LIGHT
  - One unit measured in SI lux unit
- Sensor.TYPE_PRESSURE
  - One unit measuring atmospheric presure in hPA
- Sensor.TYPE_PROXIMITY
  - One unit measured in centimeters
- More details on: http://developer.android.com/reference/android/hardware/SensorEvent.html

Sensors

```
public class MainActivity extends Activity implements SensorEventListener {
    private SensorManager sensorManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
        List<Sensor> sensorList = sensorManager.getSensorList(Sensor.TYPE_ALL);
        for (Sensor s: sensorList)
        {
          Log.v("TAG",String.format("Name:%s, Vendor:%s,
                                    Type:%d",s.getName(),s.getVendor(),s.getType()));
        }
    }
}
```

Samsung Galaxy Tablet 10.1

Name:BH1721FVC Light Sensor,Vendor:ROHM,Type:5
Name:MPL rotation vector,Vendor:Invensense,Type:11
Name:MPL linear accel,Vendor:Invensense,Type:10
Name:MPL gravity,Vendor:Invensense,Type:9
Name:MPL Gyro,Vendor:Invensense,Type:4
Name:MPL accel,Vendor:Invensense,Type:1
Name:AK8975 Magnetic field Sensor,Vendor:Asahi Kasei Microdevices,Type:2
Name:AK8975 Orientation Sensor,Vendor:Asahi Kasei Microdevices,Type:3

Location

- Location services enables you to find your location based on a GPS or a network.
- Requires the following permisions:
  - *android.permission.ACCESS_FINE_LOCATION*
  - *android.permission.INTERNET*
- *Use: LocationManager* to obtain informations about the current location

Location

```
public class MainActivity extends Activity implements LocationListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        LocationManager lm = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
        boolean gps = lm.isProviderEnabled(LocationManager.GPS_PROVIDER);
        boolean net = lm.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
        Log.v("TAG","GPS:"+String.valueOf(gps)+" Net:"+String.valueOf(net));
        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 3000, 5, this);
    }
    @Override
    public void onLocationChanged(Location l) {
        Log.v("TAG",String.format("Location changed: Long:%f,
                                   Lat:%f",(float)l.getLongitude(),(float)l.getLatitude()));
    }
    @Override
    public void onProviderDisabled(String txt) {
        Log.v("TAG",String.format("Provider disabled: %s",txt));
    }
    @Override
    public void onProviderEnabled(String txt) {
        Log.v("TAG",String.format("Provider enabled: %s",txt));
    }
    @Override
    public void onStatusChanged(String arg0, int arg1, Bundle arg2) {
        Log.v("TAG","Status Changed");
    }
}
```

Location

- Sometimes location can be cached. In this cases it is recommended that you have an update system that rejects some old locations (if you receive a location at an time interval that is higher than a specific rate (for example, more than 10 seconds) you can assume that that location may not be correct and require a new location again.

Sound

- Android system supports different sound types (mp3, mp4, ogg , etc)
- There are two ways an application can use the sound manager of Android:
  - Use the MediaPlayer class (usually for larger sound)
  - Use SoundPool class (for small sound ~ a couple of seconds)

SoundPool

- Constructor:
  public SoundPool (int maxStreams, int streamType, int srcQuality)
- Load a stream from resources:
  public int load (Context context, int resId, int priority)
- Play a stream
  public final int play (int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)
- Sound resources can be put in the "res/raw" folder.

Sound - example

```
SoundPool sp = new SoundPool(1, AudioManager.STREAM_MUSIC, 0);
if (sp!=null)
{
        int streamID = sp.load((Context)this,R.raw.my_sound,1);
        float volume = 0.5f;
        sp.play(streamID,volume,volume,0,0,1);
        …
        sp.stop(streamID);
        sp.unload(streamID);
        sp.release();
        sp = null;
}
```

```
MediaPlayer mp = MediaPlayer.create((Context)this,R.raw.my_sound);
if (mp!=null)
{
        mp.prepare();
        mp.start();
        …
        mp.stop(streamID);
        mp.release();
        mp = null;
}
```

Android Programing

Gavrilut Dragos

Camera

- Hardware component that allows one to take picture / record videos / audio stream / preview and so on.
- It can be access indirectly via intent or directly through te Camera object. If access directly the following permissions must be added:
  - android.permission.CAMERA
  - android.hardware.camera
  - android.permission.RECORD_AUDIO (optional if you want to perform audio recording).

Check and Access Camera Object

```
@SuppressLint("NewApi")
public int GetCamerasCount()
{
    if (this.getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA))
    {
        if (android.os.Build.VERSION.SDK_INT >= 9)
            return Camera.getNumberOfCameras();
        else
            return 1;
    } else {
        return 0;
    }
}
public Camera GetCamera(int cameraID)
{
    int cameraCount = GetCamerasCount();
    if (cameraCount <= 0)
        return null;
    try {
        if (cameraCount == 1)
            return Camera.open();
        else
            return Camera.open(cameraID);
    }
    catch (Exception e) {
        return null;
    }
}
```

Camera usage flow

- Obtain an instance of Camera
- Preview
- Take a picture / video
- Be carefull with system events (Pause / Terminate). Camera object should be released (with Camera.release() API)

Camera Preview

```
public class CameraPreviewSurface extends SurfaceView implements SurfaceHolder.Callback {
    private SurfaceHolder surfaceHolder;
    private Camera cameraObject;
    public CameraPreviewSurface(Context context, Camera camera) {
        super(context);
        cameraObject = camera;
        surfaceHolder = getHolder();
        surfaceHolder.addCallback(this);
        surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }
    public void surfaceCreated(SurfaceHolder holder) {
        try {
            cameraObject.setPreviewDisplay(holder);
            cameraObject.startPreview();
        } catch (Exception e) { … }
    }
    public void surfaceDestroyed(SurfaceHolder holder) {
        try {
            cameraObject.stopPreview();
        } catch (Exception e){ … }
    }
    public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
        if (surfaceHolder.getSurface() == null){
          return;
        }
        try {
            cameraObject.stopPreview();
        } catch (Exception e) { … }
        try {
            cameraObject.setPreviewDisplay(surfaceHolder);
            cameraObject.startPreview();
        } catch (Exception e) { … }
    }
}
```

Camera TakePicture

```
public class CameraPreviewSurface extends SurfaceView implements SurfaceHolder.Callback {
    private SurfaceHolder surfaceHolder;
    private Camera cameraObject;

    private String pictureFileName;
    private PictureCallback pictureCallbackObject = new PictureCallback() {
        @Override
        public void onPictureTaken(byte[] data, Camera camera) {

            File pictureFile = new File(pictureFileName);
            if (pictureFile == null)
                return;

            try {
                FileOutputStream fos = new FileOutputStream(pictureFile);
                fos.write(data);
                fos.close();
            } catch (Exception e) { }
        }
    };


    public void TakePicture(String fileName)
    {
        if (cameraObject!=null)
        {
            pictureFileName = fileName;
            cameraObject.takePicture(null, null, pictureCallbackObject);
        }
    }
}
```

Camera Features

- Can be set/read using Camera.Parameters object
- A normal flow for setting the Camera parameters can be done in the following way:
  - Obtain a Camera.Parameters from the current Camera object
  - Set the parameters using the newly obtain object
  - Apply that object to the camera

```
public void SetCameraParameters(Camera cameraObject)
{
    Camera.Parameters params = cameraObject.getParameters();
    // ...
    List<Size> previewSizes = params.getSupportedPreviewSizes();
    // ....
    params.setPreviewSize(previewSizes.get(0).width, previewSizes.get(0).width);
    // ...
    cameraObject.setParameters(params);
}
```

Camera Features

Feature	API	Description
Face Detection	14	Identify human faces within a picture and use them for focus, metering and white balance
Metering Areas	14	Specify one or more areas within an image for calculating white balance
Focus Areas	14	Set one or more areas within an image to use for focus
White Balance Lock	14	Stop or start automatic white balance adjustments
Exposure Lock	14	Stop or start automatic exposure adjustments
Video Snapshot	14	Take a picture while shooting video (frame grab)
Time Lapse Video	11	Record frames with set delays to record a time lapse video
Multiple Cameras	9	More than one camera on a device, including front-facing and back-facing cameras
Focus Distance	9	Reports distances between the camera and objects that appear to be in focus
Zoom	8	Set image magnification
Exposure Compensation	8	Increase or decrease the light exposure level
GPS Data	5	Include or omit geographic location data with the image
White Balance	5	Set the white balance mode, which affects color values in the captured image
Focus Mode	5	Set how the camera focuses on a subject such as automatic, fixed, macro or infinity
Scene Mode	5	Types of photography situations such as night, beach, snow or candlelight scenes
JPEG Quality	5	Compression level for a JPEG image/
Flash Mode	5	Turn flash on, off, or use automatic setting
Color Effects	5	Apply a color effect to the captured image such as black and white, sepia tone or negative.
Anti-Banding	5	Reduces the effect of banding in color gradients due to JPEG compression
Picture Format	1	Specify the file format for the picture
Picture Size	1	Specify the pixel dimensions of the saved picture

Camera – Record Video

- It is done using the MediaRecorder Object
- The following steps must be performed to start recording:
  - Unlock the Camera
  - Prepare a MediaRecorder object
  - Start Recording
- The following steps must be performed to stop the record:
  - Stop the Media Recorder Object
  - Release the Media Recorder Object
  - Lock the Camera

Camera – Record Video

- It is done using the MediaRecorder Object
- The following steps must be performed to start recording:
  - Unlock the Camera
  - Prepare a MediaRecorder object
  - Start Recording
- The following steps must be performed to stop the record:
  - Stop the Media Recorder Object
  - Release the Media Recorder Object
  - Lock the Camera

Camera – Record Video

```
public boolean StartRecord(Camera cameraObject, MediaRecorder mr)
{
    cameraObject.unlock();
    mr.setCamera(cameraObject);
    mr.setAudioSource(MediaRecorder.AudioSource.CAMCORDER);
    mr.setVideoSource(MediaRecorder.VideoSource.CAMERA);
    mr.setProfile(CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH));
    mr.setOutputFile("...");
    mr.setPreviewDisplay(surfaceHolder.getSurface());
    try {
        mr.prepare();
        mr.start();
    } catch (Exception e) {
        mr.reset();
        mr.release();
        cameraObject.lock();
        return false;
    }
    return true;
}
public void StopRecord(Camera cameraObject, MediaRecorder mr)
{
    mr.stop();
    mr.reset();
    mr.release();
    cameraObject.lock();
}
```

Camera – Record Audio

```
public boolean StartAudioRecording(MediaRecorder mr, String fileName)
{
    mr.setAudioSource(MediaRecorder.AudioSource.MIC);
    mr.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
    mr.setOutputFile(fileName);
    mr.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);

    try {
        mr.prepare();
        mr.start();
        return true;
    }
    catch (Exception e) {
        return false;
    }
}

public void StopAudioRecording(MediaRecorder mr)
{
    mr.stop();
    mr.release();
}
```

Android Programing

Gavrilut Dragos

SMSManager

- An object that can be used to send SMS messages programatically
- Any message send using this object will not appear in the outbox
- The following permission needs to be added to the manifest:

<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.WRITE_SMS“/>

```
public void SendSMS(String number,String text)
{
    SmsManager smsManager = SmsManager.getDefault();
    smsManager.sendTextMessage(number, null, text, null, null);
}
```

Receiving a SMS message

- The following permission needs to be added to the manifest:
  <uses-permission android:name="android.permission.RECEIVE_SMS" />
- The following code should be added to the manifest to register a class that will be listening to the messages that are send:

```
<receiver android:name="MySMSReceiver" android:enabled="true">
   <intent-filter>
       action android:name="android.provider.Telephony.SMS_RECEIVED" />
   </intent-filter>
</receiver>
```

- Create the class MySMSReceiver to intercept SMS messages

Receiving a SMS message

```
public class SMSReceiver extends BroadcastReceiver
{
    public void onReceive(Context context, Intent intent) {
        if ( intent.getExtras() != null )
        {
            Object[] smsextras = (Object[]) intent.getExtras().get( "pdus" );
            for (int tr = 0; tr < smsextras.length; tr++ )
            {
                SmsMessage smsMessage = SmsMessage.createFromPdu((byte[])smsextras[tr]);

                String text = smsMessage.getMessageBody().toString();
                String number = smsMessage.getOriginatingAddress();

                 // record text and number
             }
         }
    }
}
```

- Use **abortBroadcast()** this message from beeing send to other receivers.

Listing all SMS messages

```
public List<MySMSMessage> getAllSms() {
        List<MySMSMessage> listSms = new ArrayList<MySMSMessage>();
        MySMSMessage obj = new MySMSMessage();
        Uri message = Uri.parse("content://sms/");
        ContentResolver cr = this.getContentResolver();

        Cursor c = cr.query(message, null, null, null, null);
        this.startManagingCursor(c);
        int totalSMSMessages = c.getCount();

        if (c.moveToFirst()) {
            for (int i = 0; i < totalSMSMessages; i++) {

                obj = new MySMSMessage();
                obj.Id = c.getString(c.getColumnIndexOrThrow("_id"));
                obj.Address = c.getString(c.getColumnIndexOrThrow("address"));
                obj.Body = c.getString(c.getColumnIndexOrThrow("body"));
                obj.ReadState = c.getString(c.getColumnIndex("read")).equals("1");
                obj.Time = c.getString(c.getColumnIndexOrThrow("date"));
                obj.Inbox = c.getString(c.getColumnIndexOrThrow("type")).contains("1");
                listSms.add(obj);
                c.moveToNext();
            }
        }
        c.close();

        return listSms;
    }
```

```
public class MySMSMessage
{
            public String Id;
            public String Address;
            public String Body;
            public boolean ReadState;
            public String Time;
            public boolean Inbox;
}
```

Delete a SMS message

- Two methods:
  - Directly on your receiver class by using abordBroadcast() function
  - Programatically – but you need to know the SMS message ID:

```
public boolean DeleteSMS(String smsId) {
    try {
        this.getContentResolver().delete(Uri.parse("content://sms/" + smsId), null, null);
        return true;
    }
    catch (Exception ex) {
        return false;
    }
}
```

- The following permission needs to be added to the manifest:
  <uses-permission android:name="android.permission.WRITE_SMS“/>

Get all contacts list

- The following permission needs to be added to the manifest:
  <uses-permission android:name="android.permission.READ_CONTACTS“/>

```
private void ReadAllContacts()
{
    ContentResolver cRes = getContentResolver();
        Cursor c = cRes.query(ContactsContract.Contacts.CONTENT_URI,null, null, null, null);
        if (c.getCount() > 0) {
            while (c.moveToNext()) {
                String id = c.getString(c.getColumnIndex(ContactsContract.Contacts._ID));
                String name = c.getString(c.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
                String tmp = c.getString(c.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER));
                int numberID = Integer.parseInt(tmp);
                if (numberID > 0)
                {
                    Cursor result = cRes.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
                                    ContactsContract.CommonDataKinds.Phone.CONTACT_ID +" = ?",
                                    new String[]{id}, null);
                    while (result.moveToNext())
                    {
                        String phoneNumber = result.getString(result.getColumnIndex(
                                             ContactsContract.CommonDataKinds.Phone.NUMBER));
                        // use the phoneNumber
                    }
                    result.close();
                }
            }
        }
    }
```

Create a contact

- The following permission needs to be added to the manifest:
  <uses-permission android:name="android.permission.WRITE_CONTACTS“/>

```
private void CreateContact(String name, String phoneNumber) {
        ArrayList<ContentProviderOperation> ops = new ArrayList<ContentProviderOperation>();
        Builder b;

        b = ContentProviderOperation.newInsert(ContactsContract.RawContacts.CONTENT_URI);
        b = b.withValue(ContactsContract.RawContacts.ACCOUNT_TYPE, "abc@gmail.com");
        b = b.withValue(ContactsContract.RawContacts.ACCOUNT_NAME, "...");
        ops.add(b.build());

        b = ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI);
        b = b.withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0);
        b = b.withValue(ContactsContract.Data.MIMETYPE,
                        ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE);
        b = b.withValue(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME,name);
        ops.add(b.build());

        b = ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI);
        b = b.withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0);
        b = b.withValue(ContactsContract.Data.MIMETYPE,
                        ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE);
        b = b.withValue(ContactsContract.CommonDataKinds.Phone.NUMBER, phoneNumber);
        b = b.withValue(ContactsContract.CommonDataKinds.Phone.TYPE,
                        ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE);
        ops.add(b.build());

        try {
            getContentResolver().applyBatch(ContactsContract.AUTHORITY, ops);
        } catch (Exception e) { … }
}
```

Delete a contact

- The following permission needs to be added to the manifest:
  <uses-permission android:name="android.permission.WRITE_CONTACTS“/>

```
private void DeleteContact(String name)
{
        ArrayList<ContentProviderOperation> ops = new ArrayList<ContentProviderOperation>();
        Builder b = ContentProviderOperation.newDelete(ContactsContract.RawContacts.CONTENT_URI);
        b = b.withSelection(ContactsContract.Data.DISPLAY_NAME + " = ? ", new String[] {name});
        ops.add(b.build());

        try
        {
            getContentResolver().applyBatch(ContactsContract.AUTHORITY, ops);
        }
        catch (Exception e)
        {
            // ...
        }
}
```

Blocking a phone-call

- The following permission needs to be added to the manifest:
  <uses-permission android:name="android.permission.PHONE_STATE" />
  <uses-permission android:name="android.permission.NEW_OUTGOING_CALL" />
- The following code should be added to the manifest to register a class that will be listening to the calls that are made:

```
<receiver android:name="MyPhoneReceiver">
    <intent-filter android:priority="100">
        <action android:name="android.intent.action.PHONE_STATE"/>
        <action android:name="android.intent.action.NEW_OUTGOING_CALL"/>
    </intent-filter>
</receiver>
```

- Create the class MyPhoneReceiver to intercept call messages

Receiving a phone call

```
public class ProcessCall extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        TelephonyManager telephony;
        telephony = (TelephonyManager)context.getSystemService(Context.TELEPHONY_SERVICE);
        MyPhoneStateListener listener = new MyPhoneStateListener (context);
        telephony.listen(listener, PhoneStateListener.LISTEN_CALL_STATE);
    }
}

public class MyPhoneStateListener extends PhoneStateListener {
    Context context;
    public MyPhoneStateListener(Context context) {
        super();
        this.context = context;
    }

    @Override
    public void onCallStateChanged(int state, String callingNumber)
    {
        super.onCallStateChanged(state, callingNumber);
        if ((state == TelephonyManager.CALL_STATE_OFFHOOK) ||
            (state == TelephonyManager.CALL_STATE_RINGING))
            {
                BlockTheCall();
            }
   }
}
```

Blocking a call - undocumented

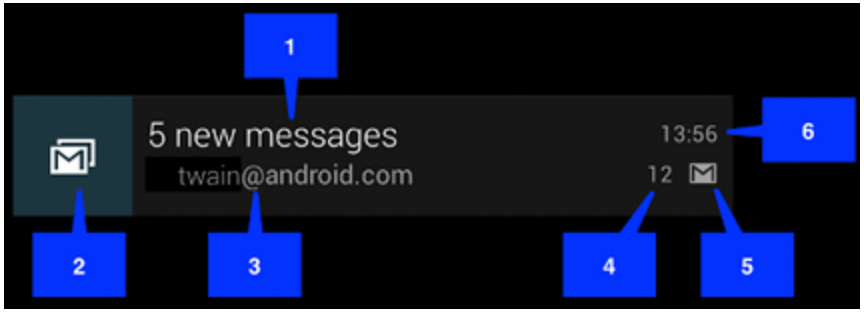
```
private void BlockCall(String callingNumber)
{
    try
    {
        TelephonyManager tm;
        tm = (TelephonyManager) context.getSystemService(Context.TELEPHONY_SERVICE);
        Class c = Class.forName(tm.getClass().getName());
        Method m = c.getDeclaredMethod("getITelephony");
        m.setAccessible(true);
        com.android.internal.telephony.ITelephony telephonyService = (ITelephony) m.invoke(tm);
        telephonyService = (ITelephony) m.invoke(tm);
        telephonyService.silenceRinger();
        telephonyService.endCall();
    }
    catch (Exception e) {

    }
 }
```

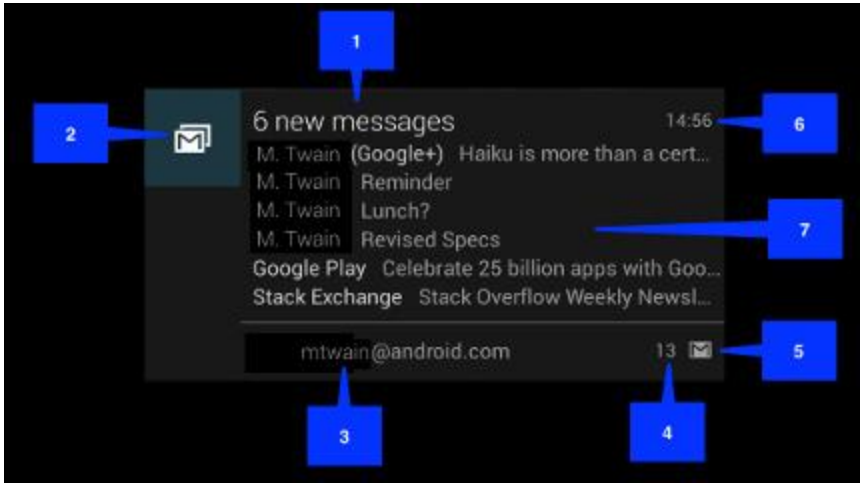
E nevoie de link static la clasa com.android.internal sau apel prin reflexie

Android Programing

Gavrilut Dragos

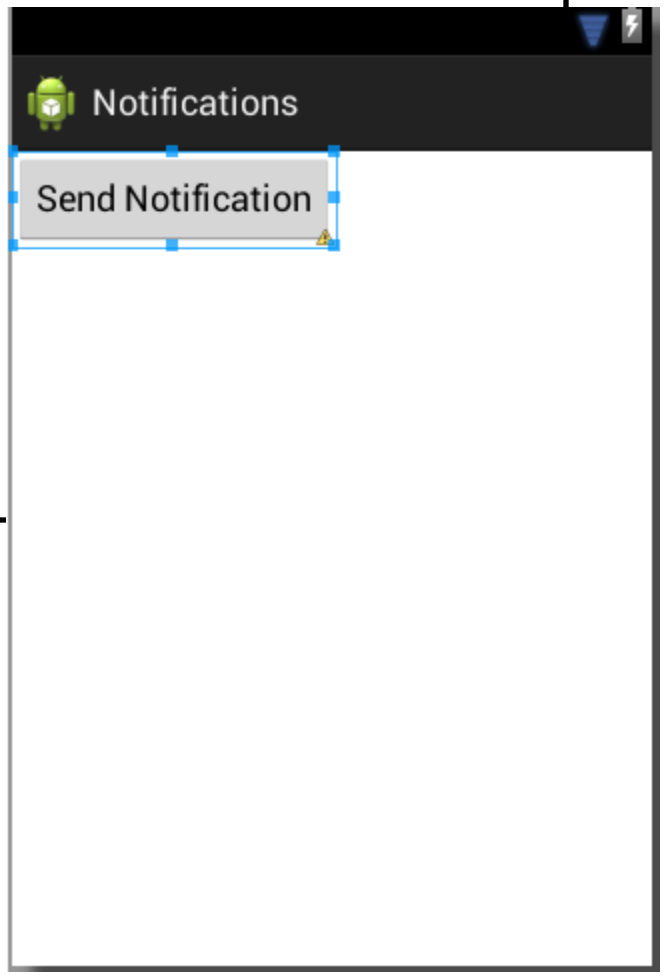
Notifications





Notifications

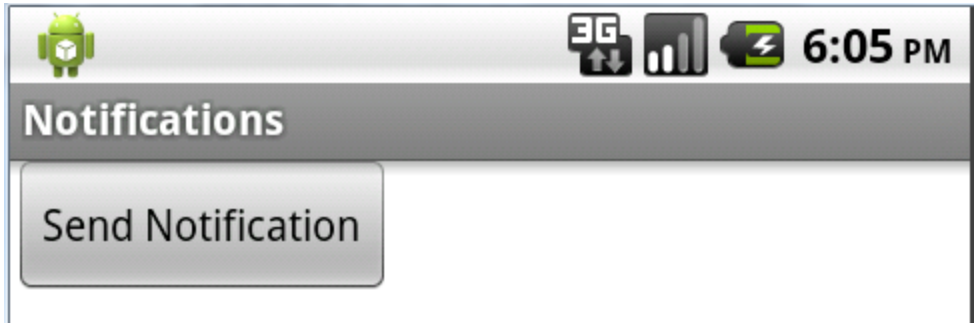
```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send Notification"
        android:onClick="sendNotification" />

</RelativeLayout>
```

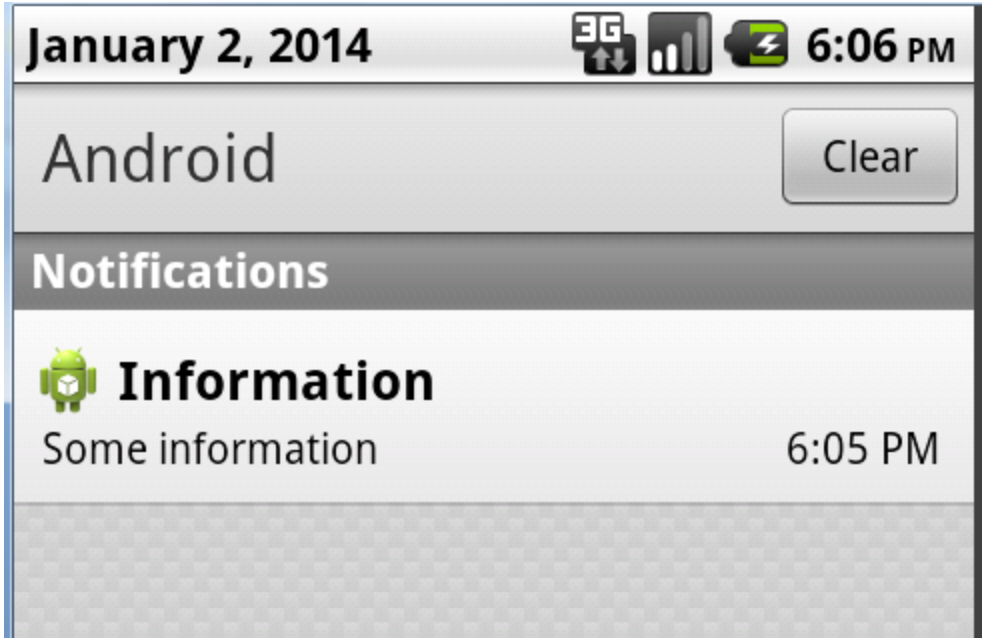


Notifications

```
public class MainActivity extends Activity {

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

    public void sendNotification(View view) {
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.infoiasi.ro"));
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);

        NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
        builder.setSmallIcon(R.drawable.ic_launcher);
        builder.setContentIntent(pendingIntent);

        builder.setAutoCancel(true);

        builder.setLargeIcon(BitmapFactory.decodeResource(getResources(), R.drawable.ic_launcher));
        builder.setContentTitle("Information");
        builder.setContentText("Some information");
        builder.setSubText("Touch to see infoiasi.ro site");

        NotificationManager notificationManager;
        notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        notificationManager.notify(1234, builder.build());

    }
}
```

Notifications





Notifications

```
public class MainActivity extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {
        // …..
}

    public void sendNotification(View view) {
        NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
        builder.setContentTitle("Percentage");
        builder.setContentTitle("Percentage for download");
        builder.setSmallIcon(R.drawable.ic_launcher);
        builder.setProgress(100, 10, false);
        // set the intent and other options

        NotificationManager notificationManager;
        notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        notificationManager.notify(1234, builder.build());

    }


}
```

JNI (C/C++ in Android System)

- Works as a library (for linux system *.so) that is loaded and called directly from dalvik virtual machine
- To compile the library the appropiate NDK package from google must be downloaded and installed. The package contains all the tools necessary to create a .so library that can be used by the Android system
  http://developer.android.com/tools/sdk/ndk/index.html

JNI (C/C++ in Android System)

- To use such a library the following steps must be performed:
  a) Create a wrapper class in Java (a wrapper class is a class that provides the Java prototype functions for the functions that are exported from the C/C++ library)
  b) Create the C/C++ code. Make sure that the function that should be exported are renamed so that they match the funtion rom the wrapper class
  c) Compile the library. The newly created library will be added in the **\libs** folder (under the appropriate architecture type). Currently – android system supports tha following architecture: ARM, x86, MIPS
  d) Compile the whole project

JNI (C/C++ in Android System)

- Create a wrapper class in Java :

```
public class MyWrapper {

     static {
         System.loadLibrary("my_library");
     }

     public static native void function_1(int param1, int param2);
     public static native void function_2();
     public static native char function_3(int param1);
}
```

JNI (C/C++ in Android System)

- Create a folder "jni" in your Android project
- Create "Android.mk" file
- Create several C/C++ files (*.h, *.cpp, *.c) that will be compiled in a single library. For this example we create "test.c" file

```
LOCAL_PATH:= $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE    := my_library
LOCAL_CFLAGS    := -Werror
LOCAL_SRC_FILES := test_c
LOCAL_LDLIBS    := -llog

include $(BUILD_SHARED_LIBRARY)
```

JNI (C/C++ in Android System)

- Create “test.c” file

```
#include <jni.h>
#include <android/log.h>
#include <stdio.h>
#include <stdlib.h>
#define  LOG(...)  __android_log_print(ANDROID_LOG_INFO, “TAG",__VA_ARGS__)

extern "C" {
    JNIEXPORT void JNICALL Java_com_teststudenti_MyWrapper_function_1(JNIEnv * env, jobject obj,
                                                                      Jint param1, jint param2);
    JNIEXPORT void JNICALL Java_com_teststudenti_MyWrapper_function_2(JNIEnv * env, jobject obj);
    JNIEXPORT void JNICALL Java_com_teststudenti_MyWrapper_function_3(JNIEnv * env, jobject obj,
                                                                      jint param1);
};

JNIEXPORT void JNICALL Java_com_teststudenti_MyWrapper_function_1(JNIEnv * env, jobject obj,  jint
param1, jint param2)
{
    LOG(“Apel functie 1 (Param1=%d,Param2=%d)“,param1,param2);
}
JNIEXPORT void JNICALL Java_com_teststudenti_MyWrapper_function_2(JNIEnv * env, jobject obj)
{
    LOG(“Apel functie 2");
}
JNIEXPORT void JNICALL Java_com_teststudenti_MyWrapper_function_3(JNIEnv * env, jobject obj, jint
param1)
{
    LOG(“Apel functie 3");
}
```

JNI (C/C++ in Android System)

- Using Java functions from JNI

```
#include <jni.h>
#include "ArrayHandler.h"

JNIEXPORT jobjectArray JNICALL <path+name> (JNIEnv *env, jobject jobj)
{
    jobjectArray ret;
    char *message[3]= {"text-1", "text-2","text-3"};
    ret= (jobjectArray)env->NewObjectArray(3,
                                    env->FindClass("java/lang/String"),
                                    env->NewStringUTF(""));

    for(int tr=0;tr<3;tr++)
        env->SetObjectArrayElement(ret,i,env->NewStringUTF(message[i]));

    return(ret);
}
```

JNI (C/C++ in Android System)

- Using Java functions from JNI (2)

```
#include <jni.h>
#include <android/bitmap.h>

JNIEXPORT void JNICALL <package_fnc>(JNIEnv * env,
                                     jobject  obj,
                                     jobject bitmap)
{
    AndroidBitmapInfo  imageInfo;
    void*              pixelsMatrix;


    if (AndroidBitmap_getInfo(env, bitmap, &imageInfo) < 0)
          return;
    if (AndroidBitmap_lockPixels(env, bitmap, &pixelsMatrix) < 0)
          return;

    // do some work with the image
    AndroidBitmap_unlockPixels(env, bitmap);
}
```

JNI (C/C++ in Android System)

- Using Java functions from JNI(3)

```
#include <jni.h>

void Java_the_package_MainActivity_getJniString( JNIEnv* env, jobject obj){

    jstring txt = (*env)->NewStringUTF(env, “<some text>”)
    jclass classObject = (*env)->FindClass(env,
                                            “<package>\MainActivity");
    jmethodID classMethod = (*env)->GetMethodID(env,
                    classObject,
                    “<function name>",
                    "(Ljava/lang/String;)Ljava/lang/String;");
    jobject result = (*env)->CallObjectMethod(env, obj, classMethod, txt);
}
```

JNI (C/C++ in Android System)

- Naming conventions / Signatures
  - B = byte
  - S = short
  - I = int
  - J = long
  - F = float
  - D = double
  - C = char
  - Z = boolean
  - V = void
  - L = prefix for a specific type
  - [→ prefix for an array
- Example
  - [B → byte[ ]
  - [[D → double[ ][ ]
  - [Ljava/lang/String; → String [ ]

Open GL in Android

- Android supports OpenGL ES (embedded system) specifications for versions 1.0, 1.1, 2.0 and 3.0
- OpenGL 1.x = fix pipeline
- OpenGL 2.x+ = programable pipeline

OpenGL ES Version	Percentage
1.1	0.1%
2.0	96.3%
3.0	3.6%

- Devices that support a version of OpenGL also suports the previous verions. That means tha 99.9% of the currently available Android devices suports OpenGL 2.0

OpenGL in Android

- There are two methods that can be used in Android to implement OpenGL code
  - Natively through JNI code (complicated but more compatible with other OS that supports OpenGL ES)
  - Through GLSurfaceView class that exista in Android. While the code is similar to the one from JNI, there are still some differences that makes porting a GLSurfaceView class inneficient.

OpenGL in Android

- GLSurfaceView uses a renderer (a class that will be called to create a list of commands for the OpenGL pipe line)
- In case of Android devices OpenGL ES can produce up to 60 FPS
- While the GLSurfaceView is slower than a code written in JNI it has the advantage of beeing able to work directly with resources (and especially with Bitmap object or PNG files). These can also be achieve using JNI with:
  - Loading the Bitmap class from android system and using it internally
  - Including a PNG C/C++ library directly in your JNI code

OpenGL in Android

- Activity Class

```
public class OpenGLActivity extends Activity {
    OpenGLSurfaceView glView;

    @Override
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        glView = new OpenGLSurfaceView(getApplication());
        setContentView(glView);
    }

    @Override
    protected void onPause() {
        super.onPause();
        glView.onPause();
    }

    @Override
    protected void onResume() {
        super.onResume();
        glView.onResume();
    }
}
```

OpenGL in Android

- Surface View Class

```
class OpenGLSurfaceView extends GLSurfaceView {

    public OpenGLSurfaceView(Context context) {
        super(context);
        setEGLContextClientVersion(2);
        setRenderer(new MyRenderer());
    }
}
```

```
class MyRenderer implements GLSurfaceView.Renderer {
    public void onDrawFrame(GL10 gl) {
        …
    }

    public void onSurfaceChanged(GL10 gl, int width, int height) {
        …
    }

    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        …
    }
}
```