Mobile Applications for Business

Master SIA/SDBIS

Octavian Dospinescu

2021

General topics

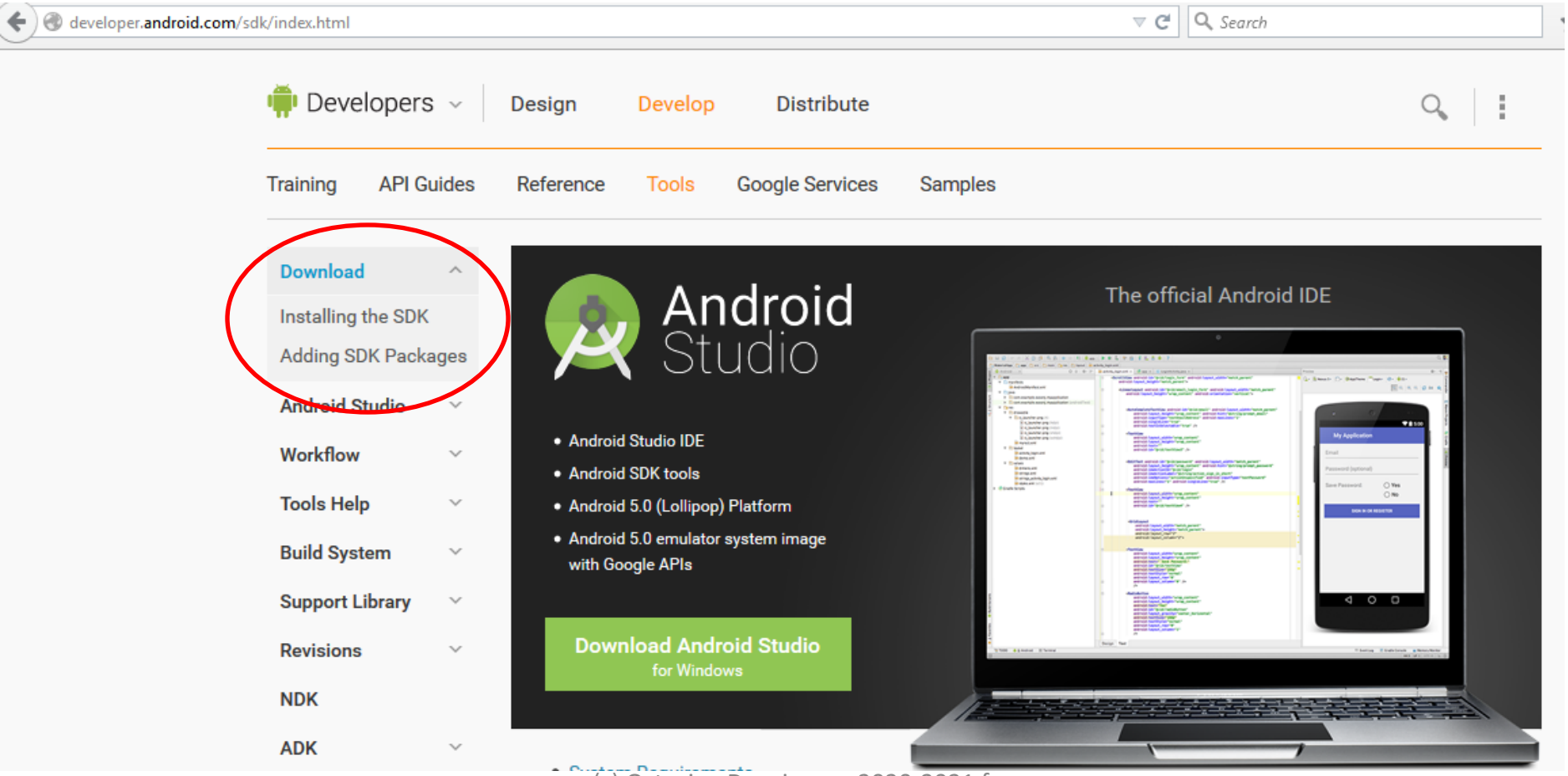
- Mobile devices
- Operating systems for mobile applications
- Mobile platforms
- Development environments for mobile applications

Tools used in the development process

- Operating system (Windows, Linux, Mac OS etc)
- Android Studio (https://developer.android.com/studio/index.html)
- JDK 7 or higher (JDK 7 recommended)
- Android SDK Manager
- AVD Manager (Android Virtual Device Manager) or Genymotion

Tools used in development

Android Studio (https://developer.android.com/studio/index.html)



Tools used in development

JDK (http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)

www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

ORACLE

Sign In/Register Help Country Communities I am a... I want to... Search

Products Solutions Downloads Store Support Training Partners About OTN

Oracle Technology Network > Java > Java SE > Downloads

- Java SE
- Java EE
- Java ME
- Java SE Support
- Java SE Advanced & Suite
- Java Embedded
- Java DB
- Web Tier
- Java Card
- Java TV
- New to Java
- Community
- Java Magazine

Overview | Downloads | Documentation | Community | Technologies | Training

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter (tick the checkbox under Subscription Center > Oracle Technology News)
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK MD5 Checksum

Looking for JDK 8 on ARM?

JDK 8 for ARM downloads have moved to the JDK 8 for ARM download page

Java SDKs and Tools

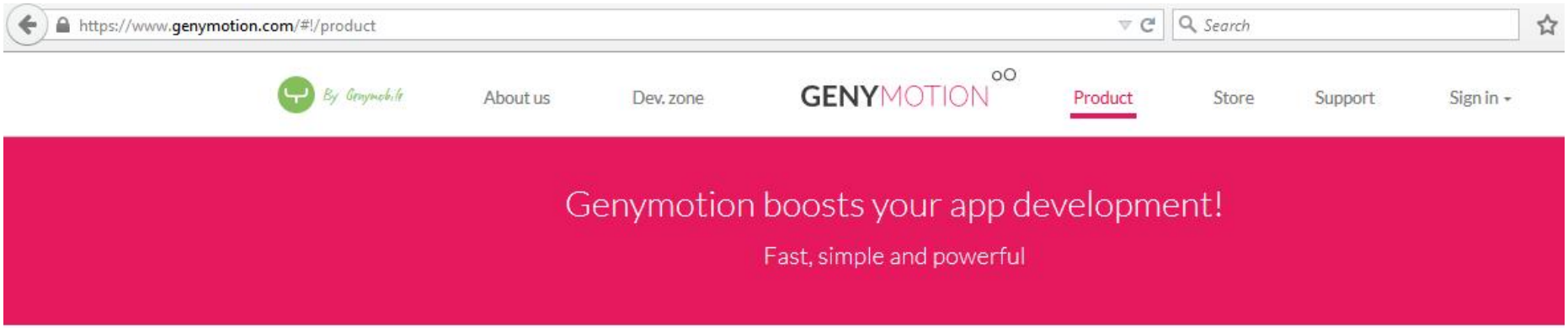
- Java SE
- Java EE and Glassfish
- Java ME
- Java Card
- NetBeans IDE
- Java Mission Control

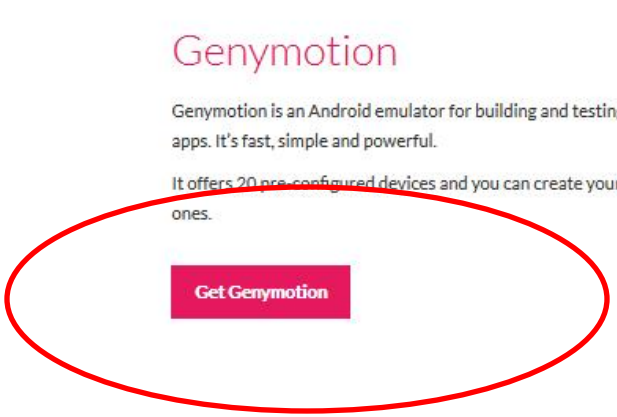
Java Resources

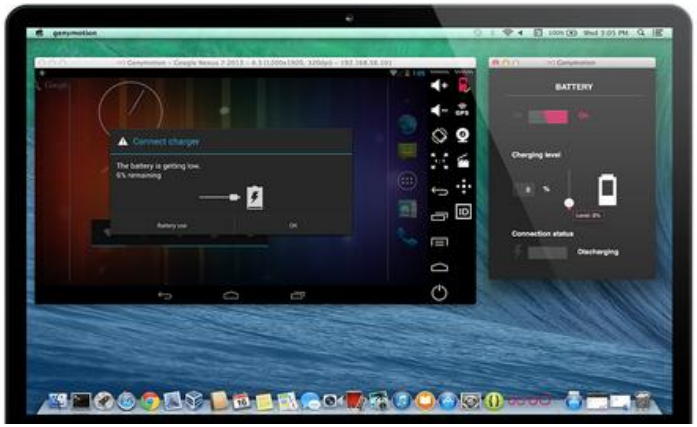
- Java APIs
- Technical Articles
- Demos and Videos
- Forums
- Java Magazine
- Java.net

Tools used in development

Genymotion (https://www.genymotion.com/#!/product)







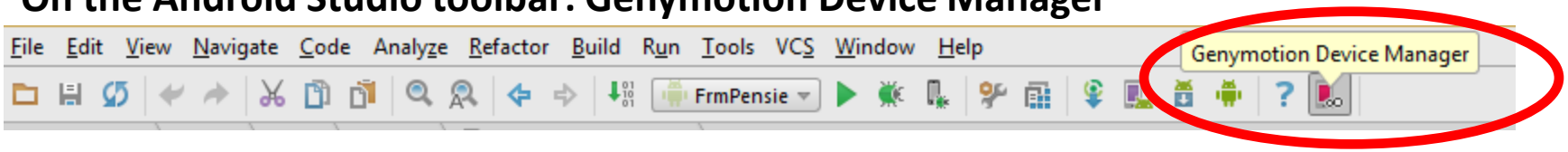
How to install the development platform

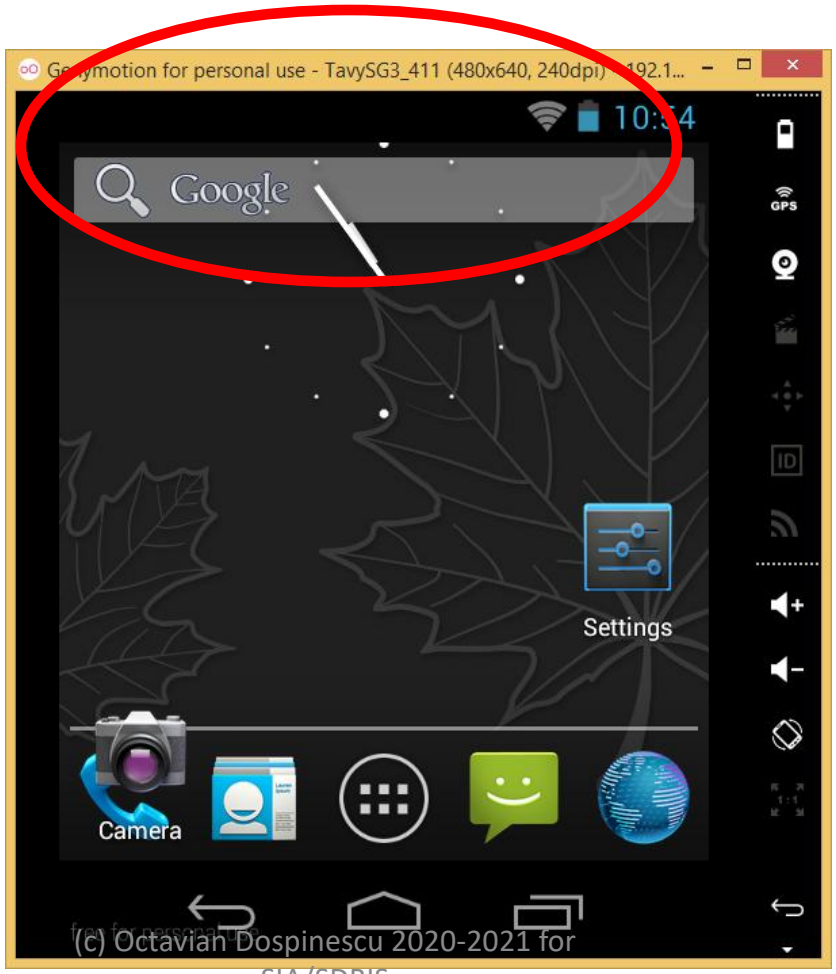
We have the following major steps:

1. Installing JDK
2. Installing Android Studio
3. Installing Android SDK(s)
4. Installing Genymotion (https://www.genymotion.com/#!/developers/user-guide)
5. Configuring Android Studio with Genymotion plugin (https://www.genymotion.com/#!/developers/user-guide#genymotion-plugin-for-android-studio)
6. Configuring a virtual device (in Android Studio or in Genymotion)

Launching the emulator

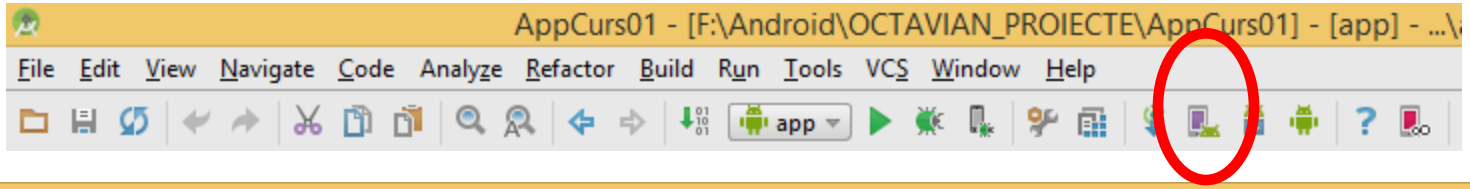
On the Android Studio toolbar: Genymotion Device Manager

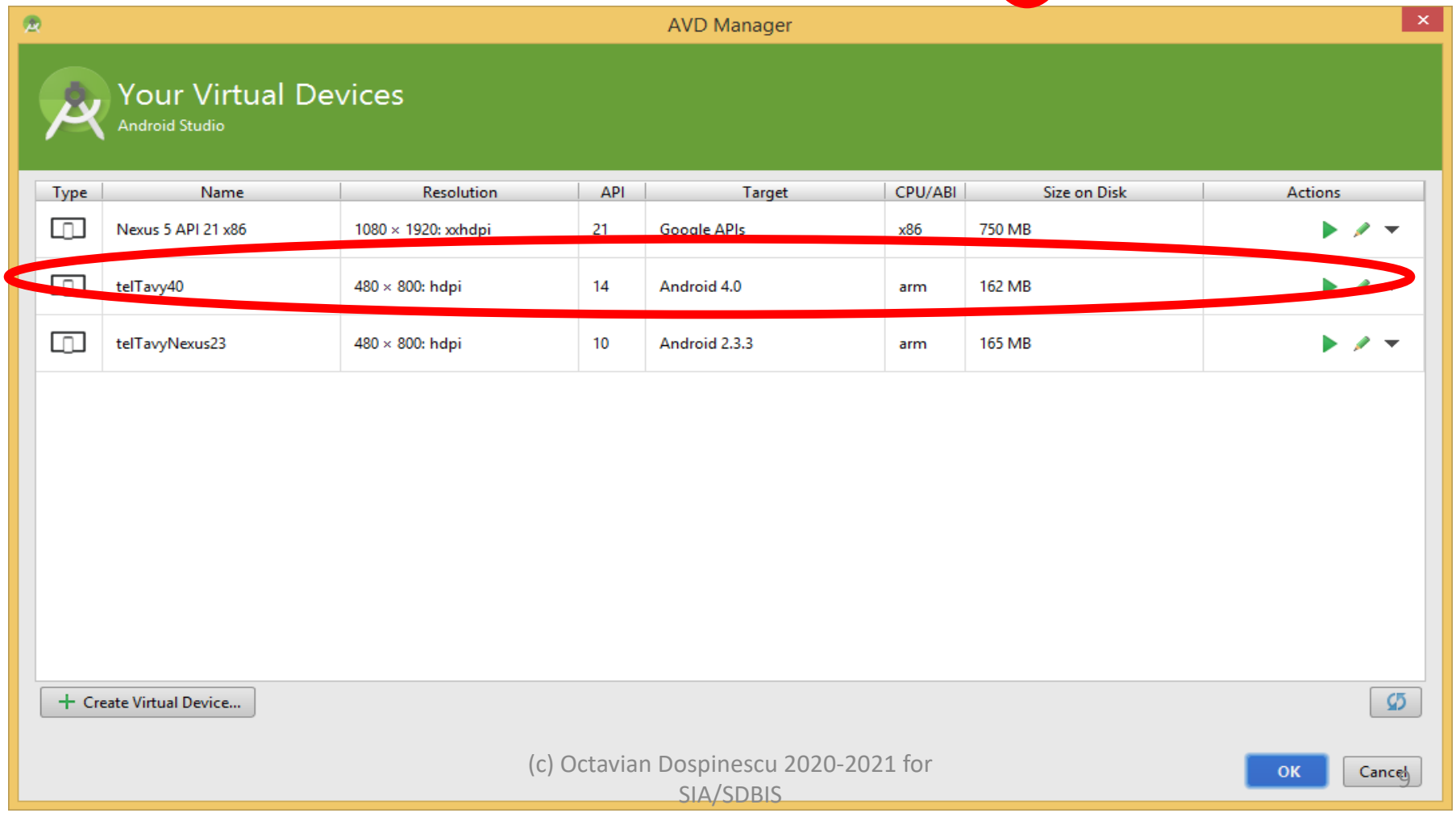




Launching the emulator

On the Android Studio toolbar: AVD Manager

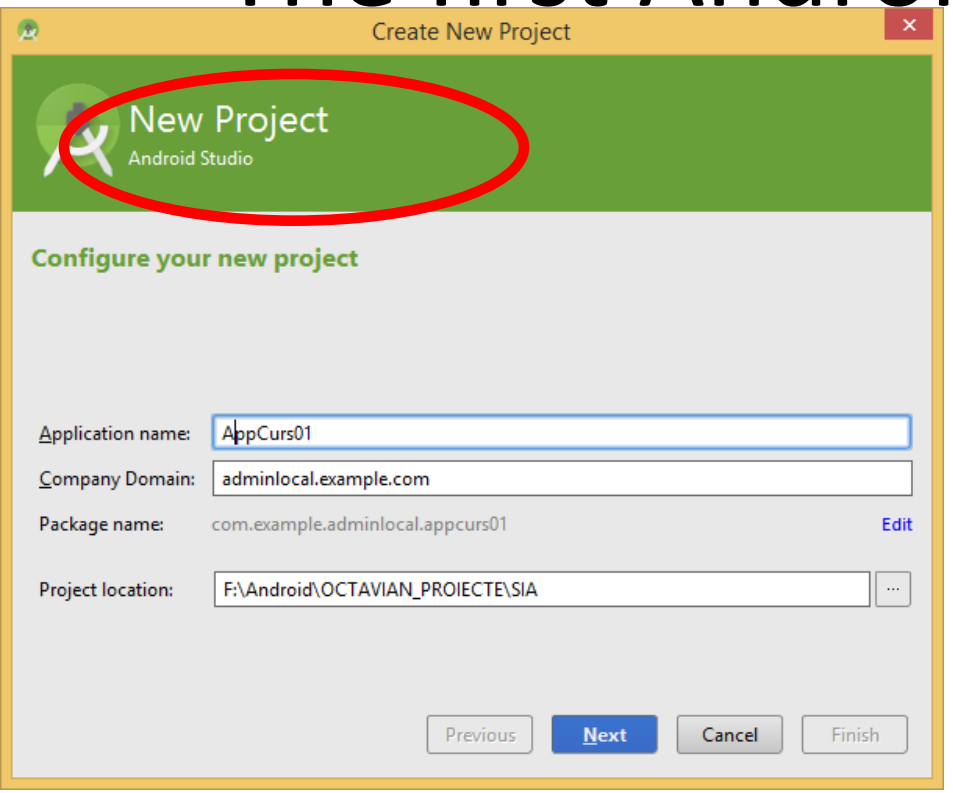


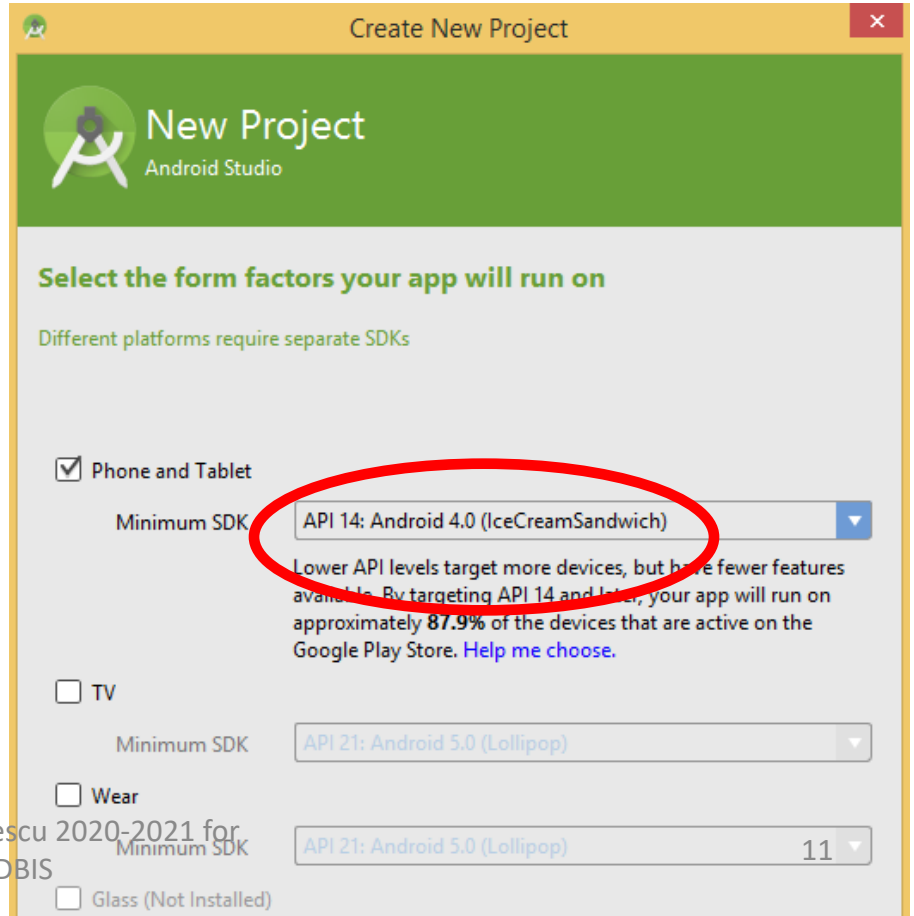


The first Android application ☺

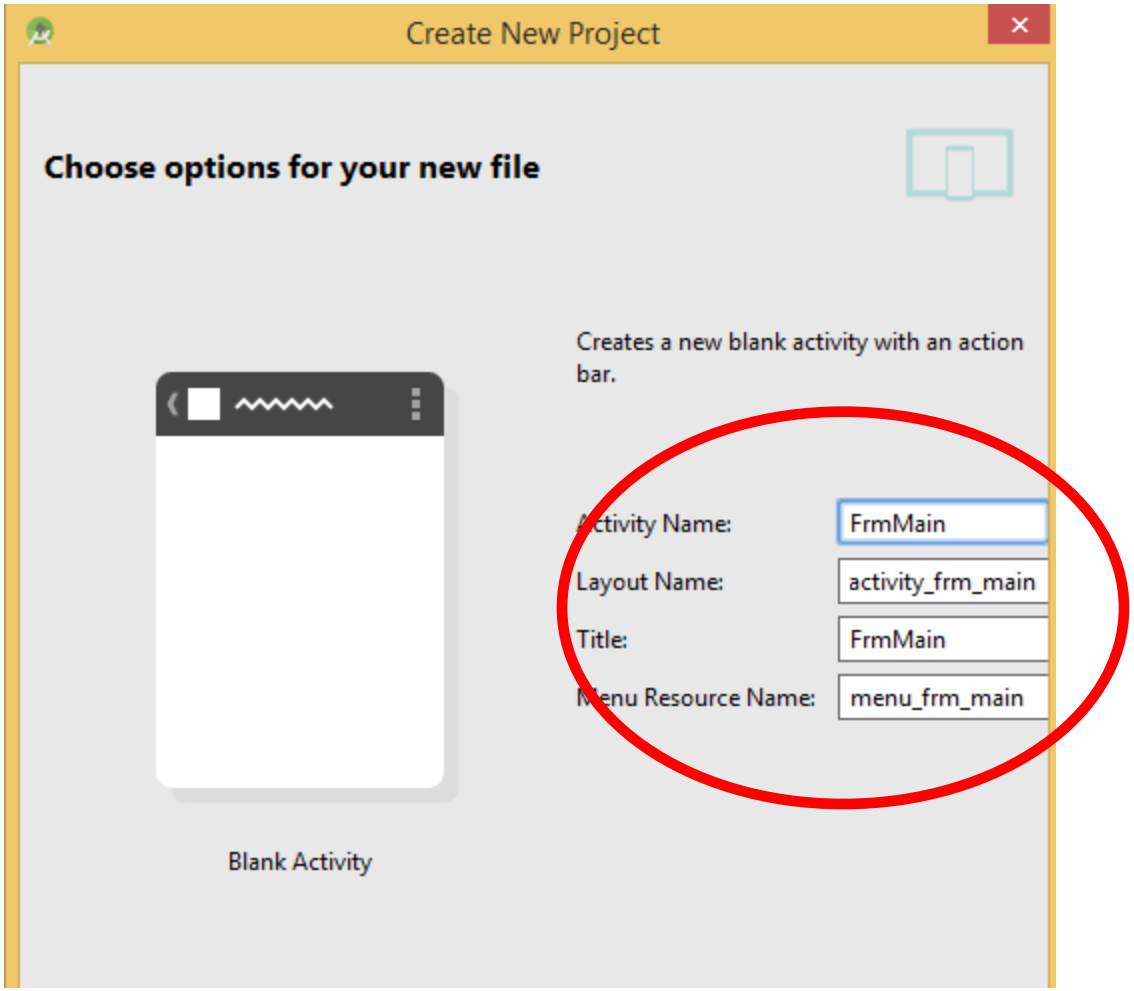
1. **Creating a new Android project**
2. **Defining the layout in .xml file**
3. **Displaying a message entered by the user**

The first Android application ☺





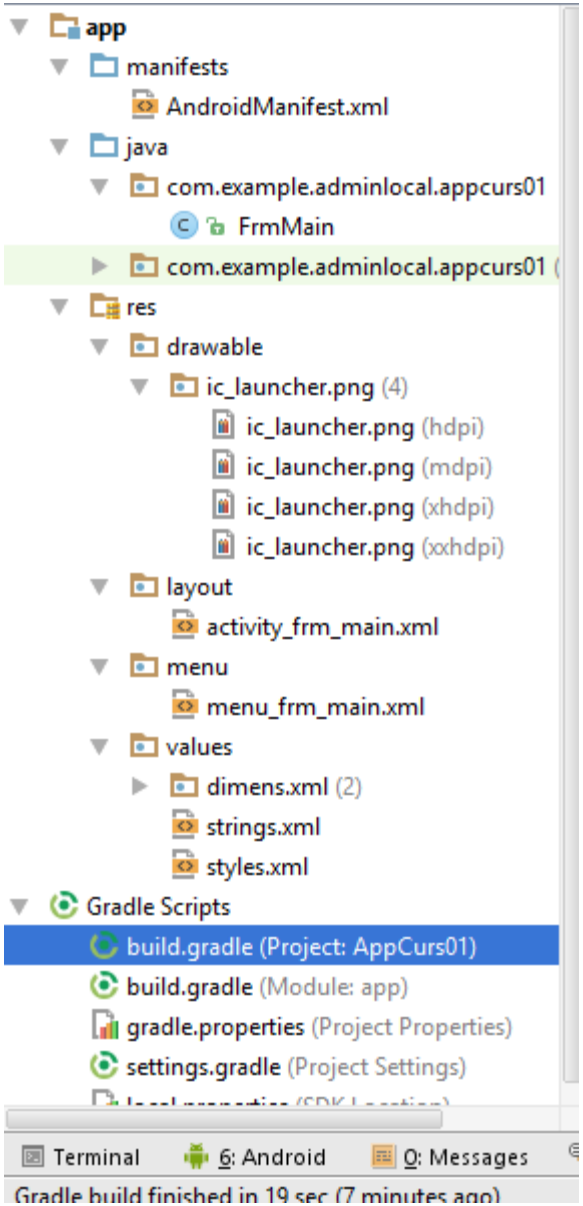
The first Android application ☺



The first Android Application ☺

- **Application Name** = the name of the application (the name which will appear on the mobile device)
- **Package Name** = the name of the package where we will define the classes of the application
- **Activity Name** = the name of the class generated when the project is created (an activity is like a form)
- **Minimum SDK Version** = the lowest API level for which we are making the application

The project's structure



- **java** = the folder where we write the Java code for our project;
- **res** = folder for the application's resources
- **AndroidManifest.xml** = an XML file that contains the description of the application (activities, services, permissions required to run the application)

More details can be read here:

http://javatechig.com/android/android-studio-project-structure

The Activity class

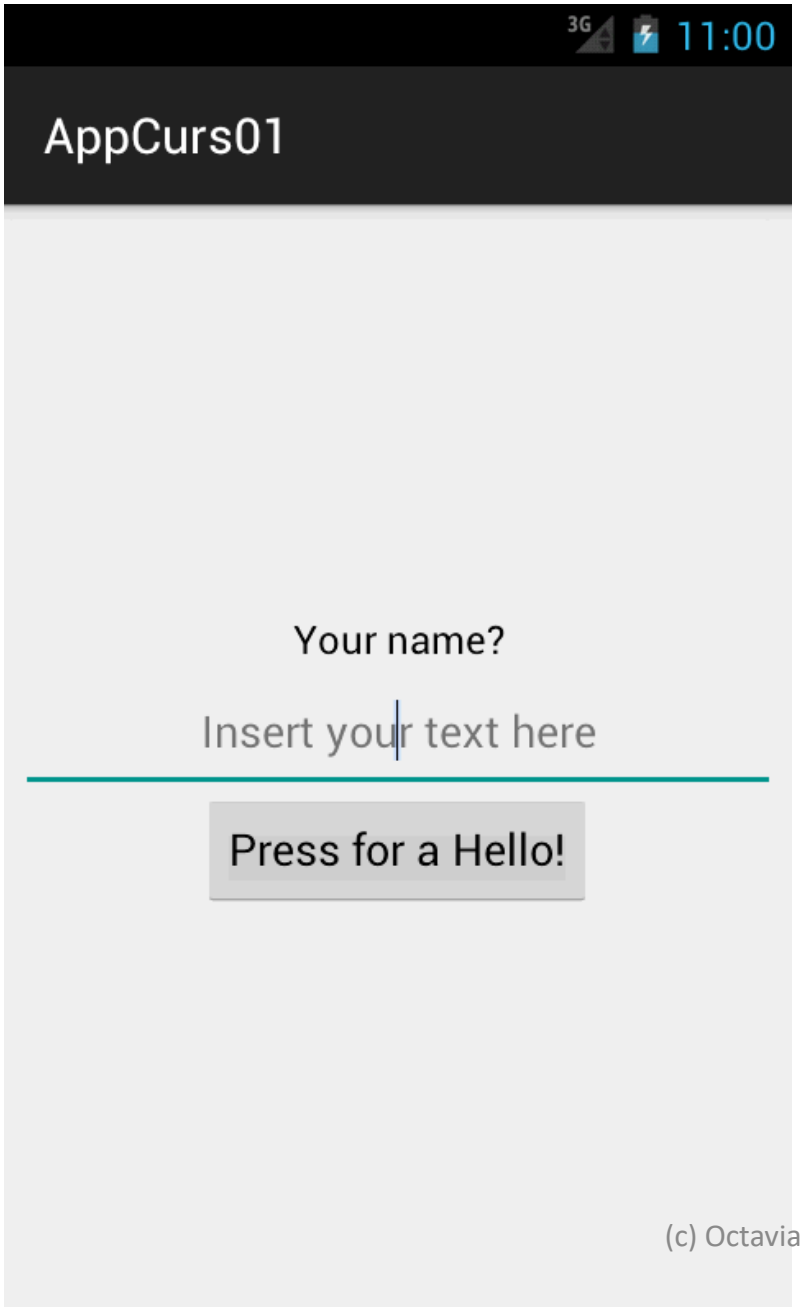
```
public class HelloWorld extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate (Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_frm_main);
   }
}
```

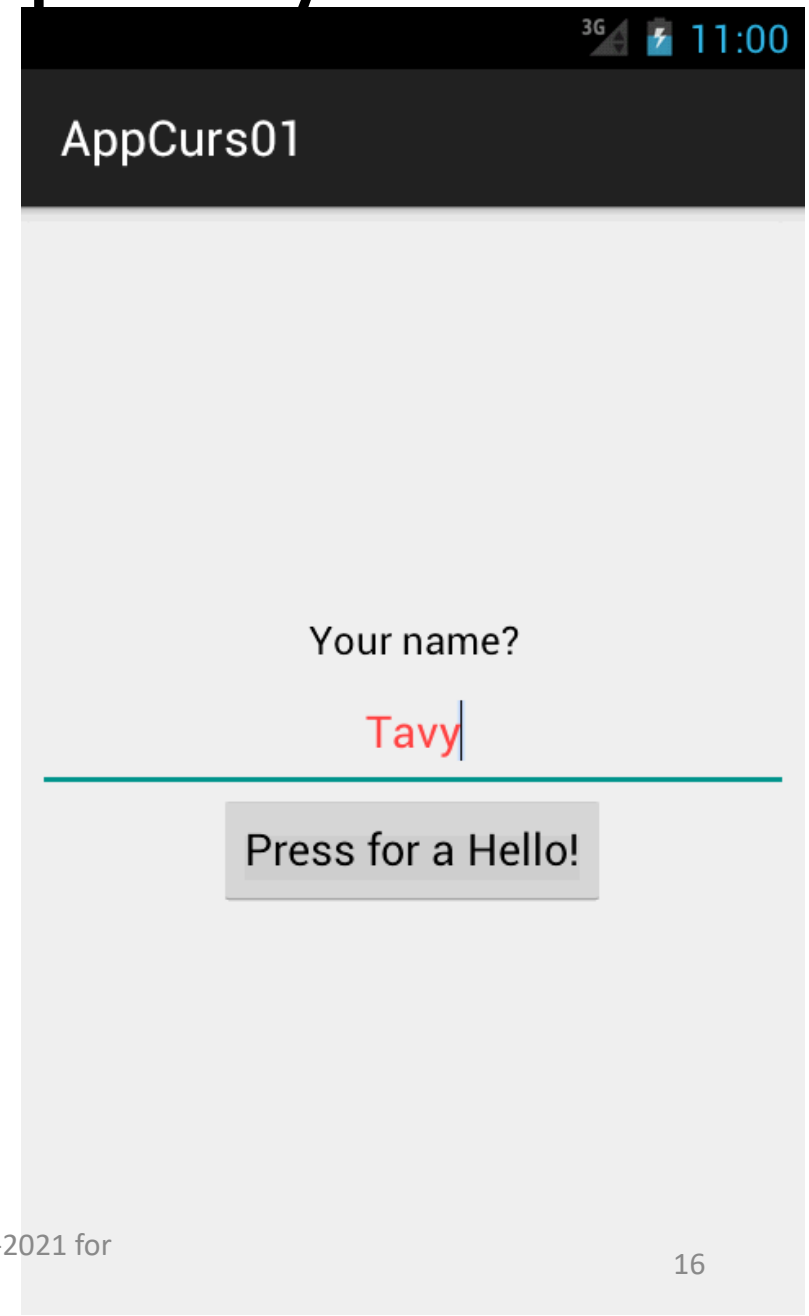
Activity = parent class for the Android "forms"

onCreate= the method called when the activity starts (when the activity is created)

R.layout.activity_frm_main = the xml description of the form (it is used to define the layout)

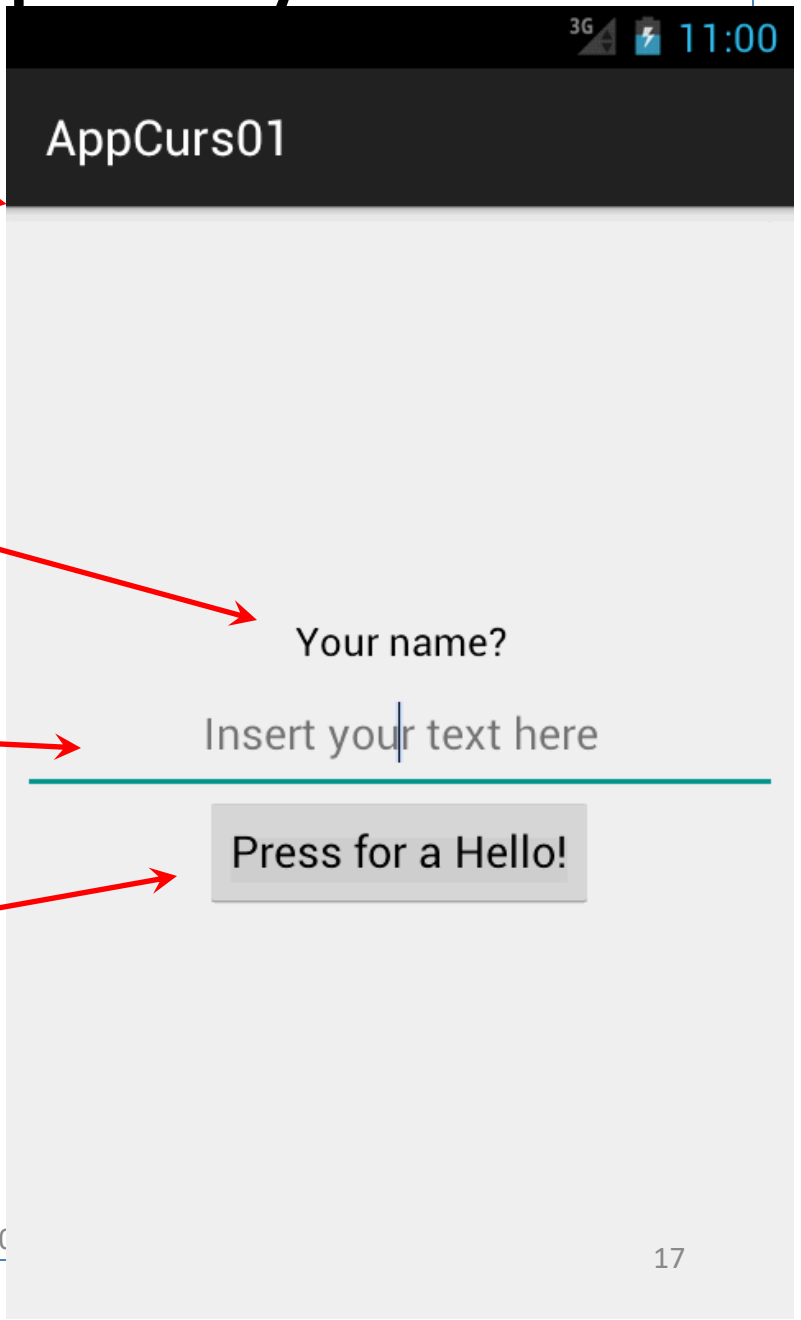
A model of a simple layout





A model of a simple layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:gravity="center"
  android:padding="5dip">
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/titlu_aplicatie"
    android:gravity="center"
    android:textSize="16dip"
    android:paddingBottom="10dip"
    android:textColor="@android:color/black" />

  <EditText
    android:id="@+id/notificare"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:textColor="@android:color/holo_red_light"
    android:hint="@string/hint_notificare" />

  <Button
    android:id="@+id/afiseaza_mesaj"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/text_afiseaza_notificare" />
</LinearLayout>
```



String (res)ources

AppCurs01 - [F:\Android\OCTAVIAN_PROIECTE\AppCurs01] - [app] - ...\app\src\main\res\values\strings.xml - Android Studio 1.0.2

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Update Info
Android Studio is ready to update.

AppCurs01 > app > src > main > res > values > strings.xml

Android
- app
  - manifests
    - AndroidManifest.xml
  - java
    - com.example.adminlocal.appcurs01
      - FrmMain
    - com.example.adminlocal.appcurs01 (an
  - res
    - drawable
    - layout
    - menu
    - values
      - dimens.xml (2)
      - strings.xml
      - styles.xml
- Gradle Scripts

strings.xml | FrmMain.java

Edit translations for all locales in the translations editor. Open editor Hide notification

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Appcurs01</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="titlu_aplicatie">Your name?</string>
    <string name="hint_notificare">Insert your text here</string>
    <string name="text_afiseaza_notificare">Press for a Hello!</string>
</resources>
```

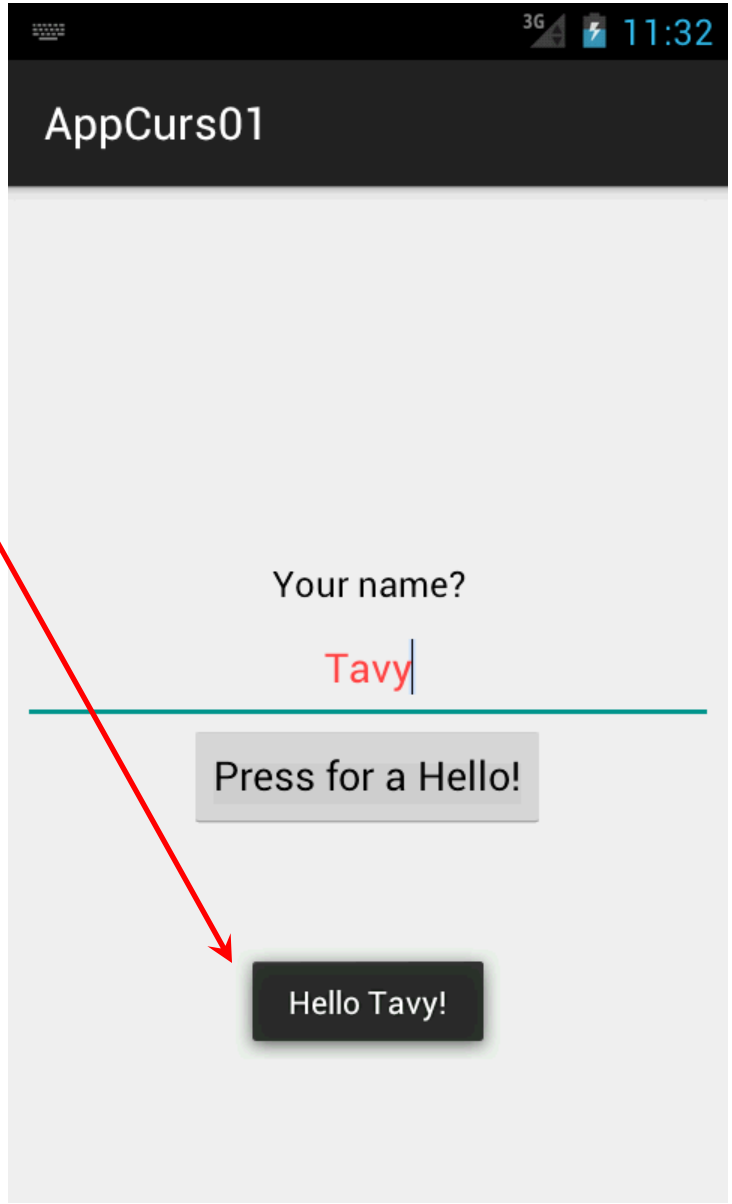
Android DDMS: logcat | ADB logs | Verbose | app: com.example.adminlocal.appcurs01

Event Log: 11:52:49 AM Update Info: Android Studio is ready to update.

The Android/Java code

```
package ro.uaic.feaa.helloworld;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class HelloWorld extends Activity implements OnClickListener{
   /** Called when the activity is first created. */
   @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_frm_main);
                Button btnNotificare = (Button)findViewById(R.id.afiseaza_mesaj);
                btnNotificare.setOnClickListener(this);
}

        @Override
        public void onClick(View v) {
                EditText textNotificare = (EditText)findViewById(R.id.notificare);

                Toast toast = Toast.makeText(getApplicationContext(), “Hello “ + textNotificare.getText() + “!”, Toast.LENGTH_SHORT);
                toast.show();
        }
}
```

The result is fabulous!!! ☺



Shortcuts in Android Studio

Go to class CTRL + N
Go to file CTRL + Shift + N
Navigate open tabs ALT + Left-Arrow; ALT + Right-Arrow
Look up recent files CTRL + E
Go to line CTRL + G
Navigate to last edit location CTRL + SHIFT + BACKSPACE
Go to declaration CTRL + B
Go to implementation CTRL + ALT + B
Go to source F4
Go to super Class CTRL + U
Show Call hierarchy CTRL + ALT + H
Search in path/project CTRL + SHIFT + Fs

- **Programming Shortcuts:**

Reformat code CTRL + ALT + L
Optimize imports CTRL + ALT + O
Code Completion CTRL + SPACE
Issue quick fix ALT + ENTER (useful for auto casting)
Surround code block CTRL + ALT + T
Rename and Refractor Shift + F6
Line Comment or Uncomment CTRL + /
Block Comment or Uncomment CTRL + SHIFT + /
Go to previous/next method ALT + UP/DOWN
Show parameters for method CTRL + P
Quick documentation lookup CTRL + Q
Delete a line CTRL + Y
View declaration in layout CTRL + B

Practical implementation

☺☺☺

Android Layouts – a short introduction

Layout = defines the visual structure of a UI component, like an activity or a widget.

A layout defines what will be drawn on the screen.

Defining layouts

A **layout** can be defined in 2 ways:

- **XML** – using the android specific attributes, we can declare the visual interface by a XML structure

- **Instantiate the elements during runtime** – we can programatically create View objects, so that we can add components/change properties depending on some conditions/requests

Types of layouts

LinearLayout – displays the sub-components in a specific direction: horizontal/vertical

RelativeLayout - displays the sub-components in a relative manner

ListView – displays a list of scrolling elements

GridView – displays elements in a bidimensional grid which cand be scrolled

Important to know! ☺

- Every View and ViewGroup supports a specific set of attributes. Some of them are the same, because every visual component has the class **View** as parent.

- For every view we can set an ID that can be used in code or to position the view in a Relative Layout:

```
android:id="@+id/btn_apasa"
```

Important to know! ☺

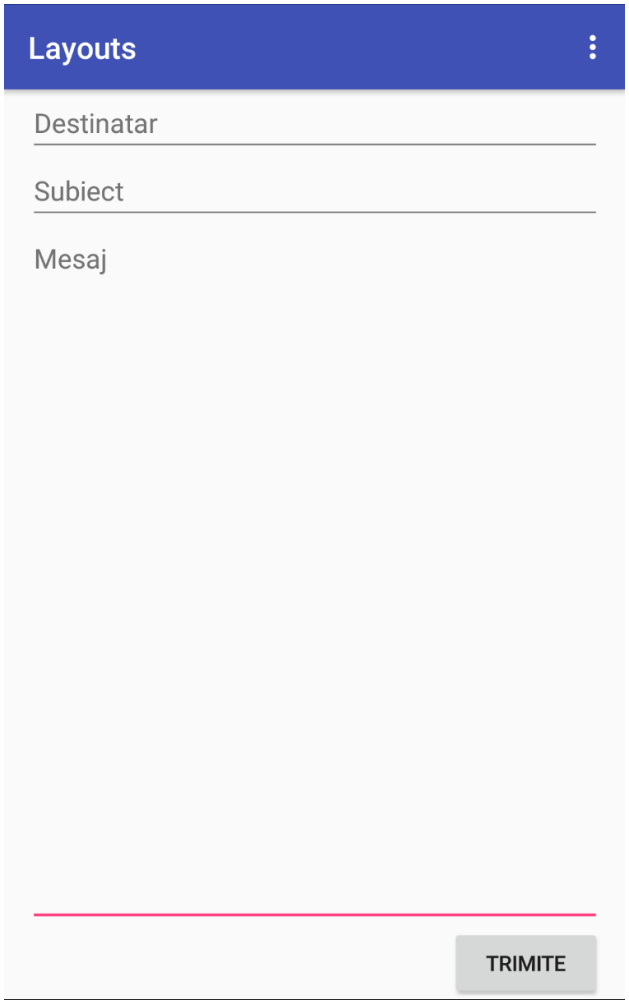
- All the attributes like layout_*something* define the way the View is displayed/positioned and are specific to the parent ViewGroup

- All the components must have **width** and **height** using the attributes:

```
android:layout_width="match_parent|wrap_content|<valoare specifica dp>"
android:layout_height="match_parent|wrap_content|<valoare specifica dp>"
```

Linear Layout

```
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/an
      droid"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



Linear Layout

To control the dimension of the layout/children, we use the attributes android:width and android:height

The possible values for both properties are:

- MATCH_PARENT
- WRAP_CONTENT
- Specific value, ex: 100dp

```
<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/an
     droid"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

Linear Layout

Using this type of layout, we can display View elements horizontal or vertical. To control the displaying direction, we use:

android:orientation

```
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android
      "
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

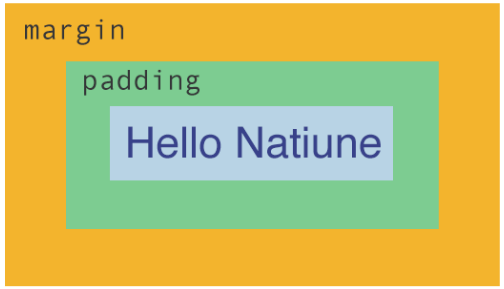
Linear Layout

The spaces are managed by the following attributes:

android:padding

and

android:margin



The **padding** is inside the view and it is included by the background. The **margin** is outside the view.

```xml
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

Linear Layout

When the View is bigger than the containing text, we can align the text inside the view using: android:gravity

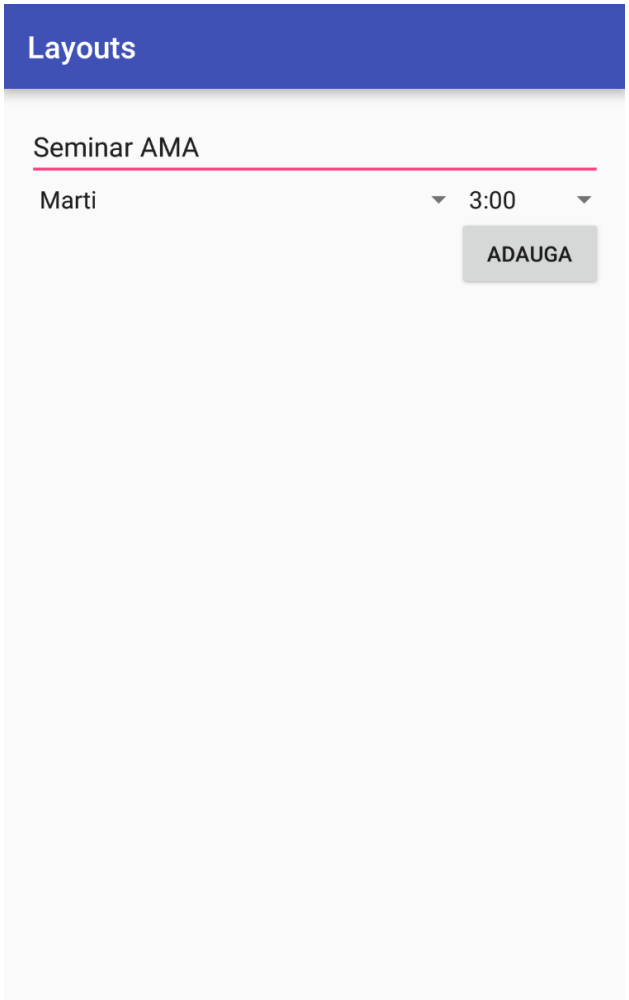
If we want to align the view inside the parent, we use the attribute: android:layout_gravity

* **Homework** ☺: check the available values using the Android Documentation

```
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/an
      droid"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

Relative Layout

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```



Relative Layout

This type of layout is used when we want to place the Views relative to "brothers" or "parent".

All the properties available for relative positioning could be found in the official documentation:

RelativeLayout.LayoutParams

Developers

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```