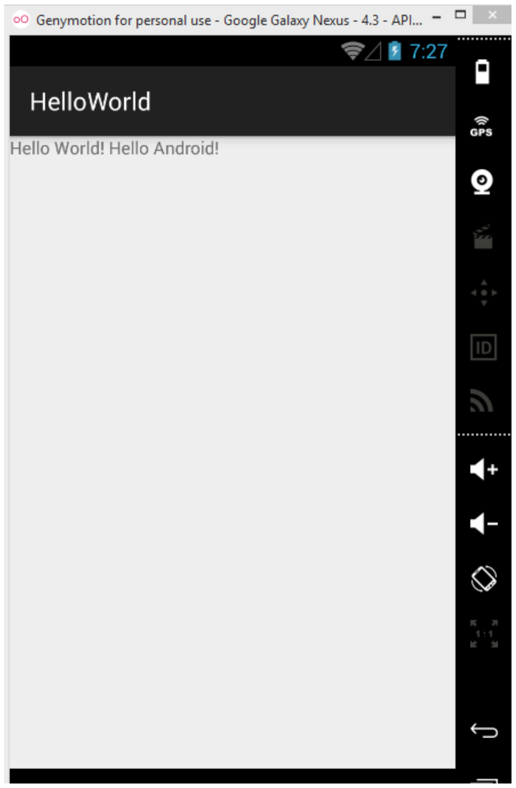
Lab 4 – Basic App Development I

The purpose of this lab is brief introduction to Basic App Development for Android apps. You will create your first "Hello Android" program. You will add additional requirements to your application.

1.1 App Components


Genymotion for personal use - Google Galaxy Nexus - 4.3 - API...
7:27
HelloWorld
Hello World! Hello Android!

App components are essential building blocks of one Android app. Each of the components exists as its own entity and plays a specific role

Activities – a single screen with a user interface. Activities work together to form a cohesive user experience however they are independent of the others.

Services – runs in the background to perform a long – running operations for remote processes. It does not provide a user interface.

Content Providers – manages a shared set of app data**.**

Broadcast receivers – a component that responds to system-wide broadcast announcements

All of these elements are declared in the manifest file in the following way:

<**activity**> elements for activities
<**service**> elements for services
<**receiver**> elements for broadcast receivers
<**provider**> elements for content providers

Read more: http://developer.android.com/guide/components/fundamentals.html#Components

1.2 Basic App

As in every Project that is created for the first time when learning a new programming language is writing a "Hello World" program. You are supposed to do the same what is shown in the figure on the left.

- **Main Activity file**

When you créate a Project in Android Studio (or if you use Eclipse it is the same) there is automatically generated piece of code that helps you to start coding.
Below is the code provided for your first program with some explanation. Of course this is not the only way to do that. We encourage you to try to do it your own way.

****

```
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView textView = new TextView(this);
        textView.setText("Hello World! Hello Android!");
        setContentView(textView);
    }
}
```

****

The bolded lines are additionaly added into the program.

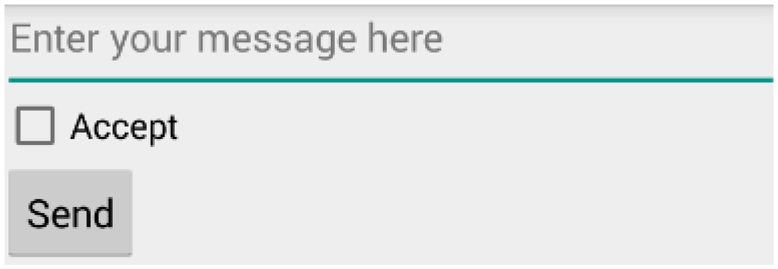
1. import android.widget.TextView; - This line allows you to output text to the screen.
2. **TextView textView = new TextView(this);** - This line is necessary if you are going to put out the first line. In this case **Hello World! Hello Android!"** will be printed.
3. The actual printing of the text is done through this line: **textView.setText("Hello World! Hello Android!");**
4. The **setContentView(textView);** goes back to the second line and actually outputs the text to the screen.

1.3 Extending your first Basic Android App

Implement a layout in XML that contains a text field and a button. The text on the button should be set to **Send**.

If you open **res** -> **layout** -> **your_Xml_file.xml** you will see two tabs **Design** and **Text**. If you click on the **Design** tab you will be able to see a phone with all its components that are currently used in your program. You can drag and drop different components in this view while the code for the elements you drag and drop will be automatically generated and it can be seen under the tab **Text**.
It is a good practice to write the code, instead of dragging elements in the **Design mode**. We will stick to that in the lab.

Below it is shown how it should look like. The main idea is for you to get comfortable with the layouts, widgets etc.



Implementing this will teach you how to:
1. Create a Linera Layout
2. Add a Button and a TextField
3. Add String resources

For now clicking on the Send button won't result in anything. In the next lab when the button is clicked the content of the TextField will be sent to another avtivity.

- ### Linear Layout

1. In Anroid Studio in **res/layout** folder open your_activity.xml.
2. Change the <**RelativeLayout>** to <**LinearLayout>**
3. Add the **andorid:orientation** and set it to **horizontal**
4. Set the width (layout_width) and the height (layout_height) to **match_parent**

LinearLayout is a subclass of **ViewGroup**. It lays out child view in either vertical or horizontal orientation.

- ### String Resources

In **res/values/strings.xml** define the following:

1. Add **edit_message** with a value “Enter your text here”
2. Add **button_send** with the value “Send”

- ## TextField

1. Define **<EditText>** element inside the <**LinearLayout**>
2. Set the id of this element to **@+id/edit_message**
    - The id provides a unique identifier for the view. It can be used in order to reference the object from your app.
    - ‘@’ sign is used when you’re referring to any resource object from XML.
    - ‘+’ sign is required only when you are defining a resource ID for the first time.
3. Set **layout_height** as **wrap_content** and **layout_width** as **match_parent**
    - **wrap_content** specifies that the view should be as big as needed to fit the contents of the view.
4. Set a hint attribute, **android_hint** named **edit_message.**
    - This will be the default text displayed when the text fiels is empty.

- ## CheckBox

1. In **res/layout** define a component <**CheckBox>**
2. Set the text **android:text**="Accept"

- ## Button

1. In **res/layout** define a component **<Button>**
2. Set **layout_height** and **layout_width** as **wrap_content**
3. Set **android:text** to the value defined in String Resources, **button_send**

References:

http://www.tutorialspoint.com/android/android_hello_world_example.htm
http://developer.android.com/training/basics/firstapp/building-ui.html#Strings