Lab 14 (OS): Android

Shokhista Ergasheva, Muwaffaq Imam, Artem Kruglov,
Nikita Lozhnikov, Giancarlo Succi, Xavier Vasquez

Week 14 – Lab

Outline

- Environment Setup
- Creating an android project
- Exercises

Environment Setup

- Download and install your preferable IDE for android applications.
- As a suggestion, you can choose one of the following:
    - Android Studio (link)
    - IntelliJ IDEA (link)

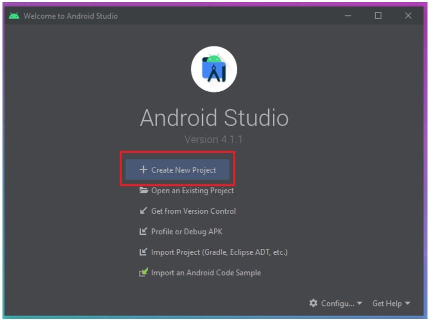
Testing

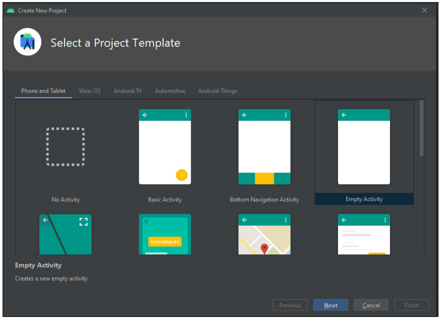
- Good to have Android device during the development
- Otherwise, use the default emulator / Genymotion

Let's start an App

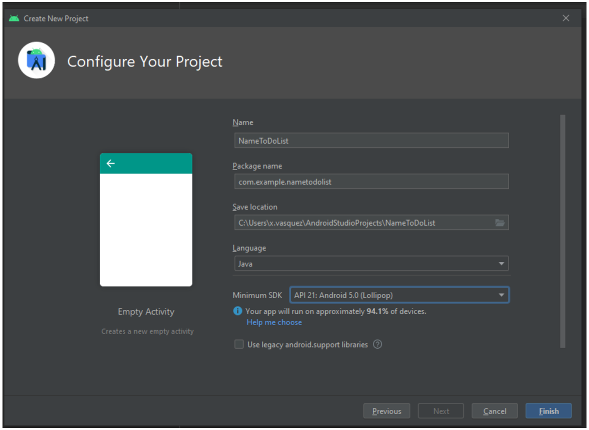
- Open the Android Studio
  - Create a new Android Studio Project
  - Select an Empty Activity - keep the name as MainActivity
  - Select Java Language
  - Select Minimum API level Android 4.0.3 or above - to run the app in 100% of devices
  - Name the application "Name_ToDoList"
- Finish Configuring your project

Let's start an App





Let's start an App


Create New Project

Configure Your Project

Name: NameToDoList

Package name: com.example.nametodolist

Save location: C:\Users\x.vasquez\AndroidStudioProjects\NameToDoList

Language: Java

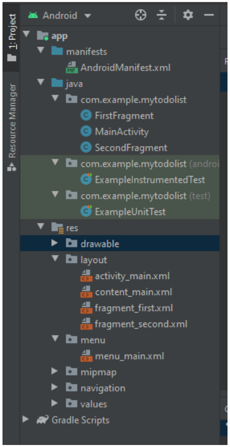
Minimum SDK: API 21: Android 5.0 (Lollipop)

Your app will run on approximately **94.1%** of devices.

Help me choose

☐ Use legacy android.support libraries

IDE introduction

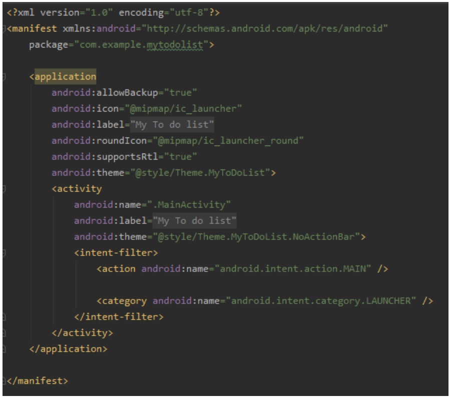


By default, Android Studio displays your project files in the Android project view. This view is organized by modules to provide quick access to your project's key source files

Each app module contains the following folders:

- manifests: Contains the AndroidManifest.xml file.
- java: Contains the Java source code files.
- res: Contains all non-code resources.

IDE introduction



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mytodolist">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My To do list"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyToDoList">
        <activity
            android:name=".MainActivity"
            android:label="My To do list"
            android:theme="@style/Theme.MyToDoList.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Every app project must have an *AndroidManifest.xml* file, which describes essential information about your app to the Android build tools, the Android operating system, and Google Play. The manifest file is required to declare the following:

- The app's package name.
- The components of the app.
- The permissions.
- The hardware and software features the app requires.

Defaults

In the *MainActivity.java* you have default code like:

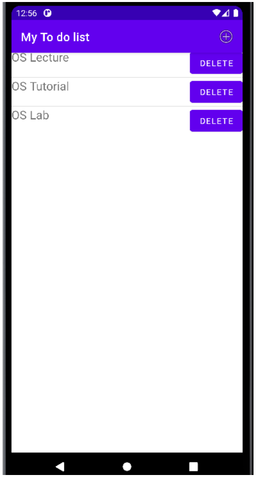
```
package com.shokhista.example.android.todolist;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Defaults

The view of this activity is set to *R.layout.activity_main*, pointing to a file *activity_main.xml* in the /res/layout directory.
And looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="
    http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello␣World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Exercise 1



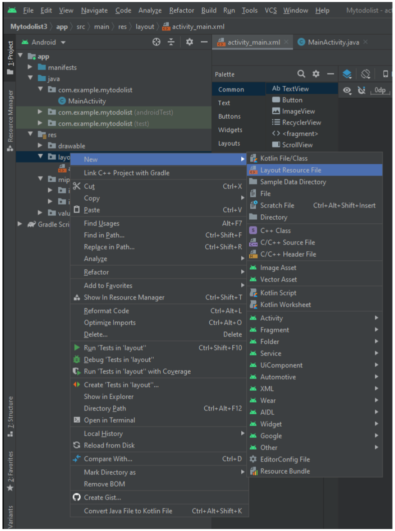
Create an application which allow the user to create a "To Do list".

It will require to have the following functionalities:

- Add item.
- Delete item.

The following slides will give a guideline to build your app.

Designing an item layout



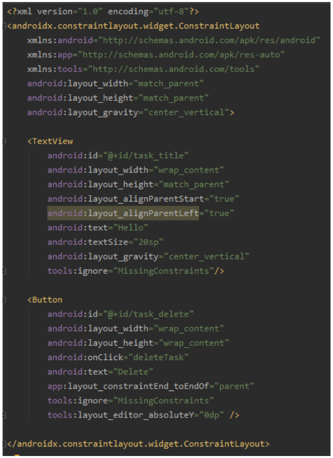
Under the */res/layout* folder lets create a new *Layout Resource file* and name it as *todo_item.xml*

In this file, we will need to add two elements:

- *TextView*, which will show our task description
- *Button*, to delete the item.

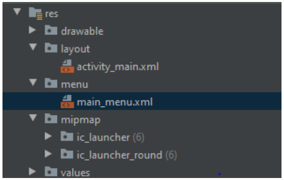
Designing an item layout

And looks like this:



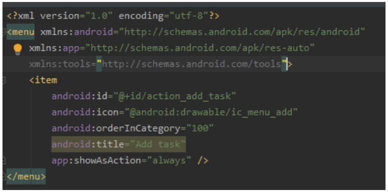
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_vertical">

    <TextView
        android:id="@+id/task_title"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:text="Hello"
        android:textSize="20sp"
        android:layout_gravity="center_vertical"
        tools:ignore="MissingConstraints"/>

    <Button
        android:id="@+id/task_delete"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="deleteTask"
        android:text="Delete"
        app:layout_constraintEnd_toEndOf="parent"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteY="0dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Creating menu



Under the /res folder, create a new *Android Resource Directory* .

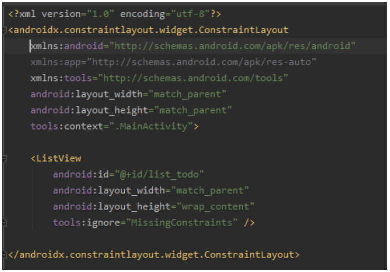
- Directory name: menu.
- Resource type: menu.



```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">
    <item
        android:id="@+id/action_add_task"
        android:icon="@android:drawable/ic_menu_add"
        android:orderInCategory="100"
        android:title="Add task"
        app:showAsAction="always" />
</menu>
```

This menu needs to contain an item, which will allow the user to add new items.

Designing main layout



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/list_todo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:ignore="MissingConstraints" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

In the *activity_main.xml*, we will need to create a *ListView* control, it will contain each item which represent a task.

Application behaviour

In the *MainActivity.java* file, we will add the code to define the behaviour of our application.

Where we need to define:

- *ListView* to represent our ListView Control.
- *ArrayList* as the structure to store our tasks.
- *ArrayAdapter* to bind our task list to application control

Application behaviour

On the onCreate() method we need to initialize our application binding the controls to the data source.

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

    mTaskListView = (ListView) findViewById(R.id.list_todo);

    mAdapter = new ArrayAdapter<String>(this,
            R.layout.todo_item,
            R.id.task_title,
            taskList);
    mTaskListView.setAdapter(mAdapter);
}
```

Application behaviour

Now, we need to define how our application will react to the user interaction. So we will override methods such as:

- *onCreateOptionsMenu()*, to allow us to render the menu in the main activity.
- *onOptionsItemSelected()*, to react to the user interactions with the menu items.

Application behaviour

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_menu, menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_add_task:
                Log.d("MainClass", "Add␣a␣new␣task");
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
```

Application behaviour

Let's create dialog to interact with user and ask a task description.

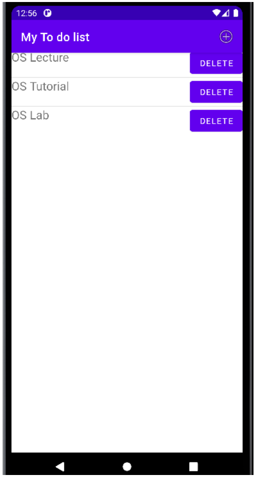
```
final EditText taskEditText = new EditText(this);
AlertDialog dialog = new AlertDialog.Builder(this)
        .setTitle("Add␣a␣new␣task")
        .setMessage("What␣do␣you␣want␣to␣do␣next?")
        .setView(taskEditText)
        .setPositiveButton("Add", new DialogInterface.
            OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which)
                {
                String task = String.valueOf(taskEditText.getText()
                    );
                Log.d(TAG, "Task␣to␣add:␣" + task);
            }
        })
        .setNegativeButton("Cancel", null)
        .create();
dialog.show();
```

Application behaviour

And two additional functions to add and remove items from list.

```java
private void addItem(String itemText){
    taskList.add(itemText);
    mAdapter.notifyDataSetChanged();
}

private void removeItem(String itemText){
    taskList.remove(itemText);
    mAdapter.notifyDataSetChanged();
}
```

Additional task



Improve your existing application, providing these new features:

- Add a tick button.
- Edit text item.
- Save the list items and their states in a local database.

The End.

Good luck in the exam!