

## Mile/km converter

From the initial application, we are going to build step by step a simple application that lets you convert miles to kilometres or vice-versa.

You will create the application in two steps:

1. Create the user interface.
2. Program the behaviour.

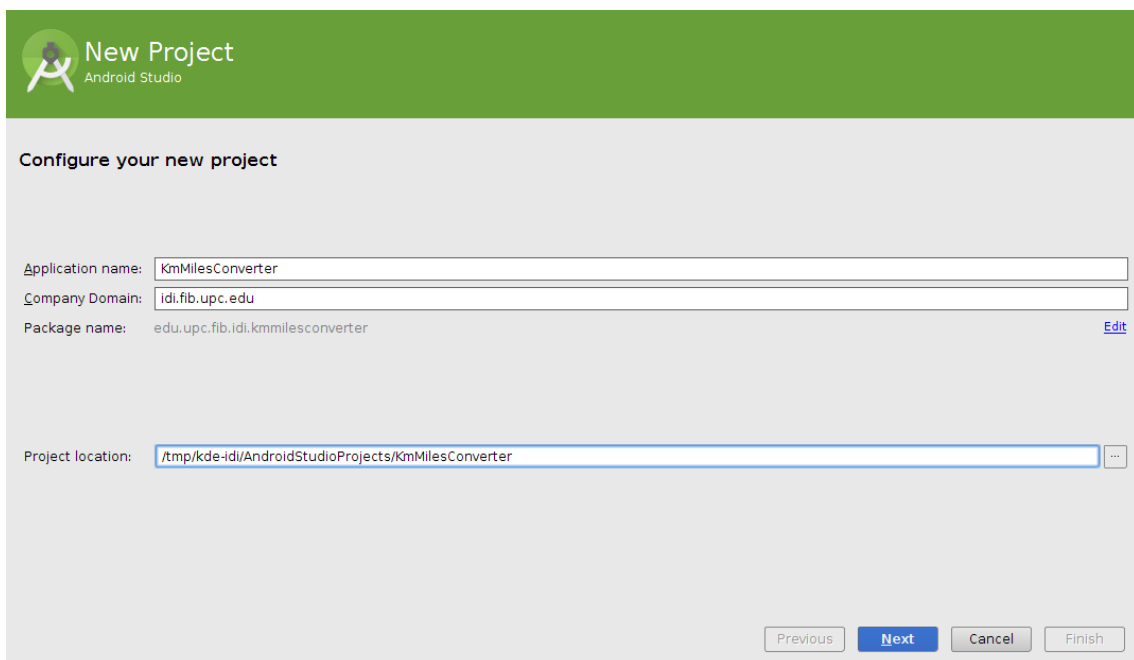
This is the normal building step, but, as you will know, for complex apps, it will require a previous (probably lengthy stage) of design. You cannot design from scratch the UI without the proper analysis and planning.

### 1. Creating the user interface

Application creation can be done following these steps:

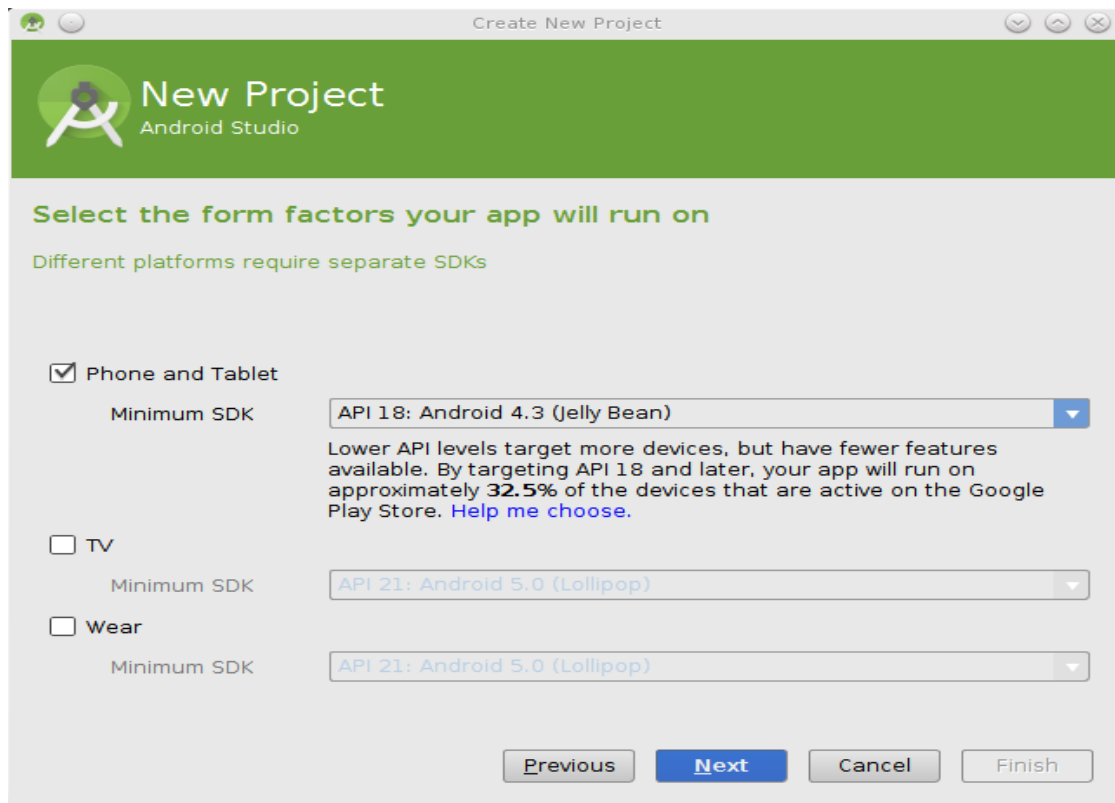
1. Run Android Studio and in the welcome screen, choose the Quick Start option *"Start a new Android Studio project"*.
2. Write the Application name, company domain and project location for your app (see figure 1). Then click **Next** button.

**Attention!!** Choose the correct project location depending on where you are working (university or your own PC).



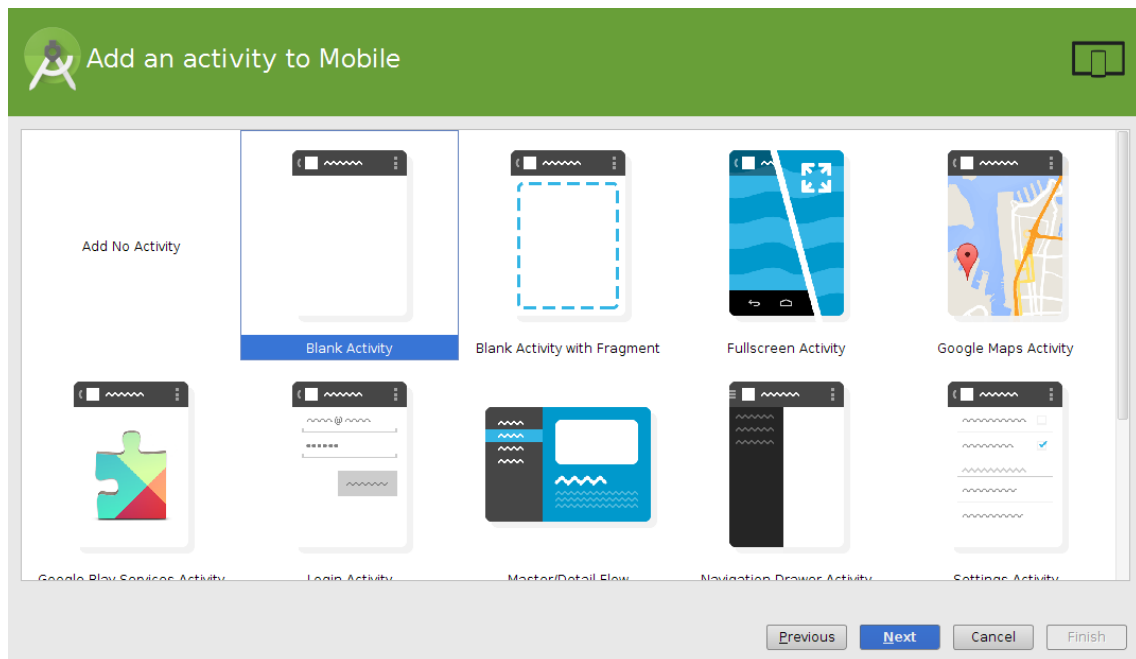
**Figure 1:** Write the name and project location.

3. Choose the Android minimum SDK that your app will run on.  
Select **API 18** (see figure 2). Then click **Next** button.



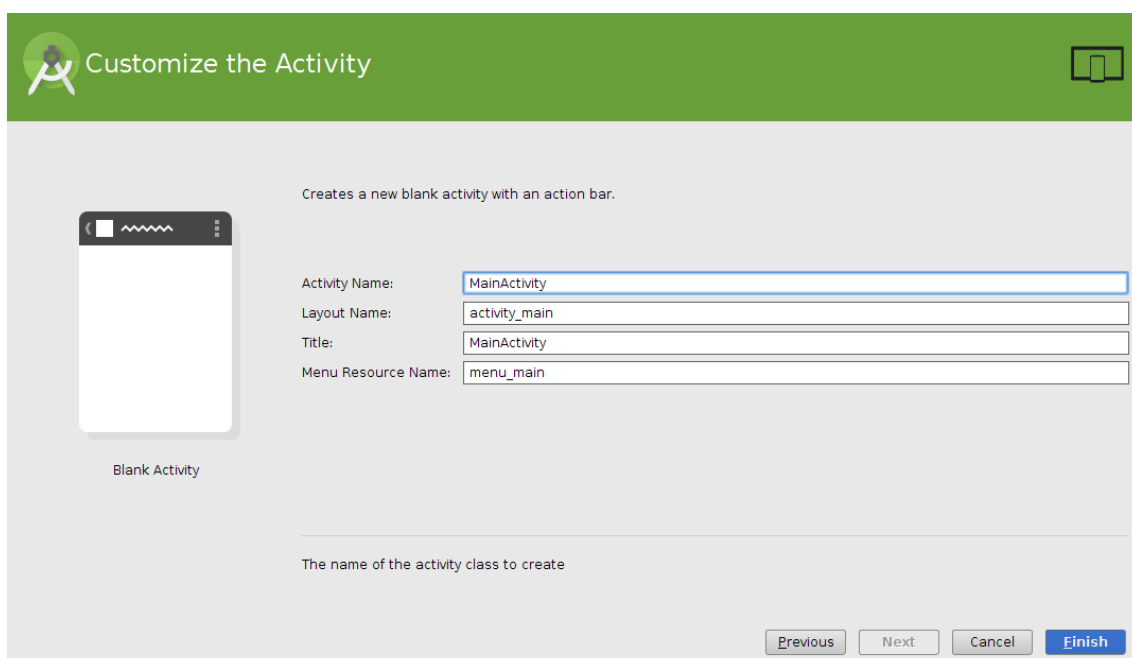
**Figure 2:** Select the minimum android SDK.

4. Choose the main activity of your app. You can choose many options: *No Activity*, *Blank Activity*, *Blank Activity with Fragment*, ... For our first app select **Blank Activity** (see figure 3). Then click **Next** button.



**Figure 3:** Select the main activity.

5. Customize the filenames of the activity. Default names are ok (see figure 4). Then click **Finish** button.

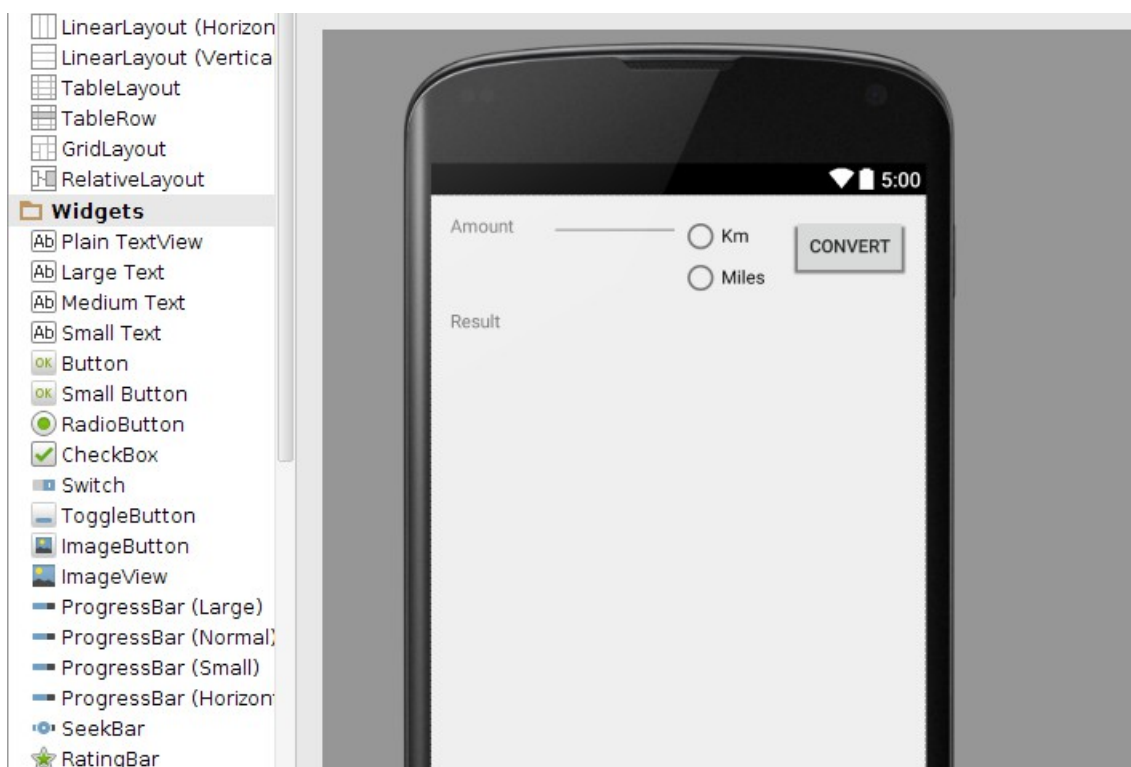


**Figure 4:** Customizing the filenames of the activity.

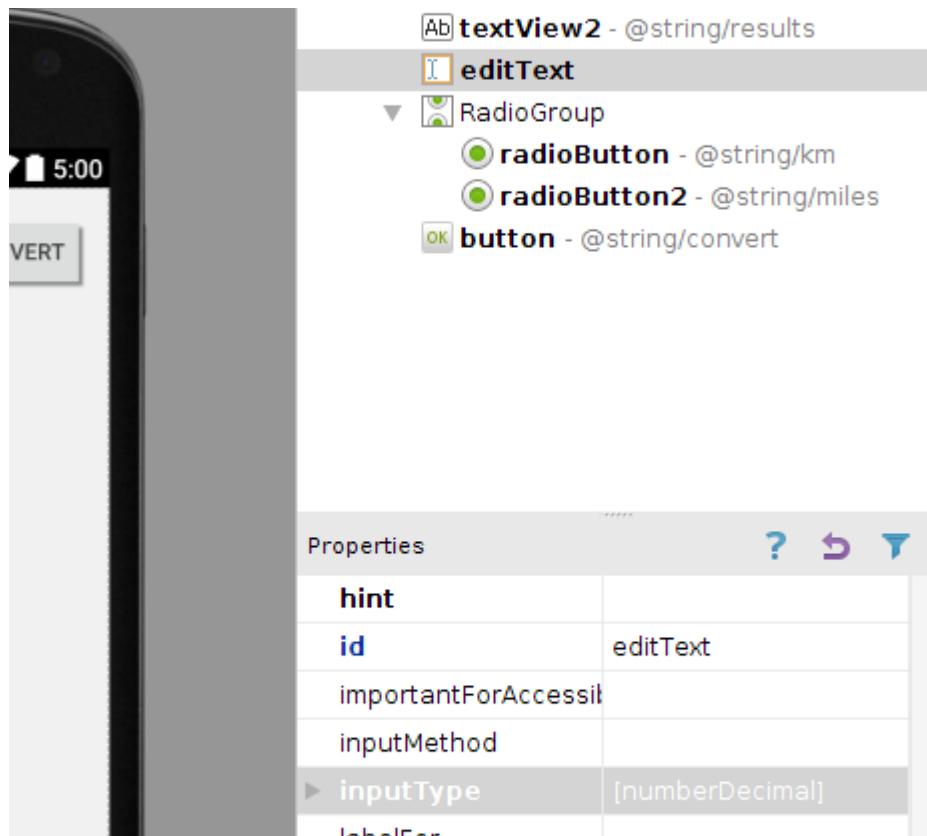
6. Now, we can add the components to the **activity\_main.xml** layout. For this simple case, we will add the following components:

- Textfield: Lets you introduce numbers with decimals and sign. This will hold the input value.
- A radio group that let you decide the direction of the conversion, from miles to kilometres or vice-versa. You must add two radio buttons inside this group.
- The corresponding labels (TextView) for all these elements and to show the results.
- A button to perform the change.

You may add all those components by dragging from the left toolbox. Note that the different tools are grouped by categories. The result should be similar to one shown in figure 5.



**Figure 5:** Initial layout.



**Figure 6:** Editing properties.

7. Once a component has been included in your UI, you can change its properties by clicking the element and visiting the right menu (see figure 6). Make sure the Textfield will accept the type you expect.
8. To add strings, you only have to right-click the element you want to modify (e.g. "amount" TextView) and click to "Edit Text..." and then add a new string to the resources file. You may add several strings in this stage, and then only select the one you need each time.

Now we can define its behaviour.

## 2. Defining the behaviour

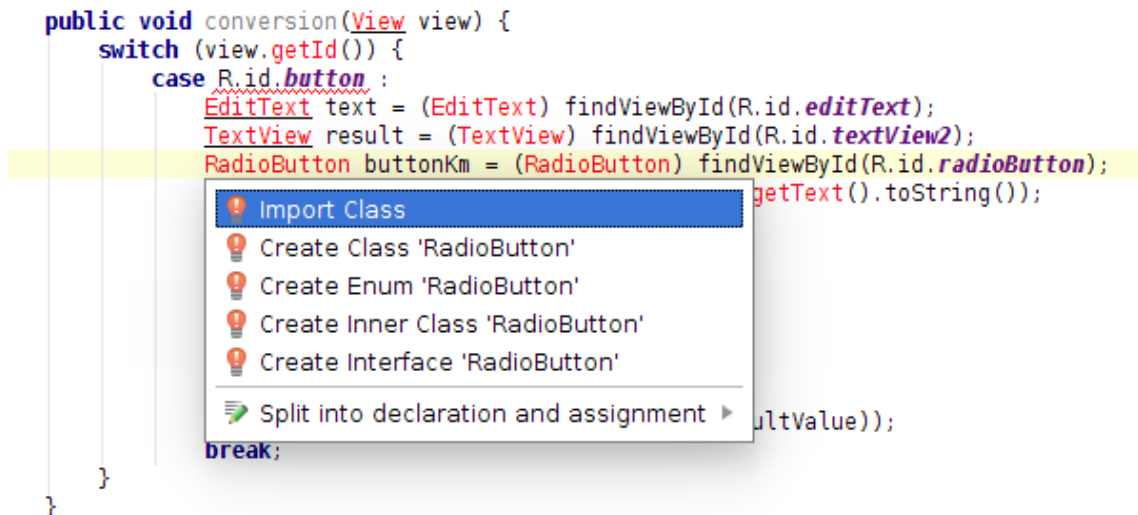
In order to determine how the application behaves, we must add code.

We first create the conversion function, and then, we will associate it with the clicking event of the button.

1. First, we go to the project files, search for the java files, and then double-click your **MainActivity.java** file (it is placed on a "src" folder). The code you may add inside the MainActivity class is that:

```
public void conversion(View view) {
    switch (view.getId()) {
        case R.id.button :
            EditText text = (EditText) findViewById(R.id.editText);
            TextView result = (TextView) findViewById(R.id.textView2);
            RadioButton buttonKm = (RadioButton)
                findViewById(R.id.radioButton);
            float inputValue = Float.parseFloat(text.getText().toString());
            double resultValue;
            if (buttonKm.isChecked()) {
                resultValue = inputValue / 1.609344;
            }
            else {
                resultValue = inputValue * 1.609344;
            }
            result.setText(String.format("%.03f", resultValue));
            break;
    }
}
```

2. If you copy this code, a lot of warnings will appear. These are caused by the fact that we did not include the proper imports to the java file. Some of them can be simply solved by pressing **Alt+Enter** over the line with the warning and choosing the suggestion of **Import class** (see figure 7).



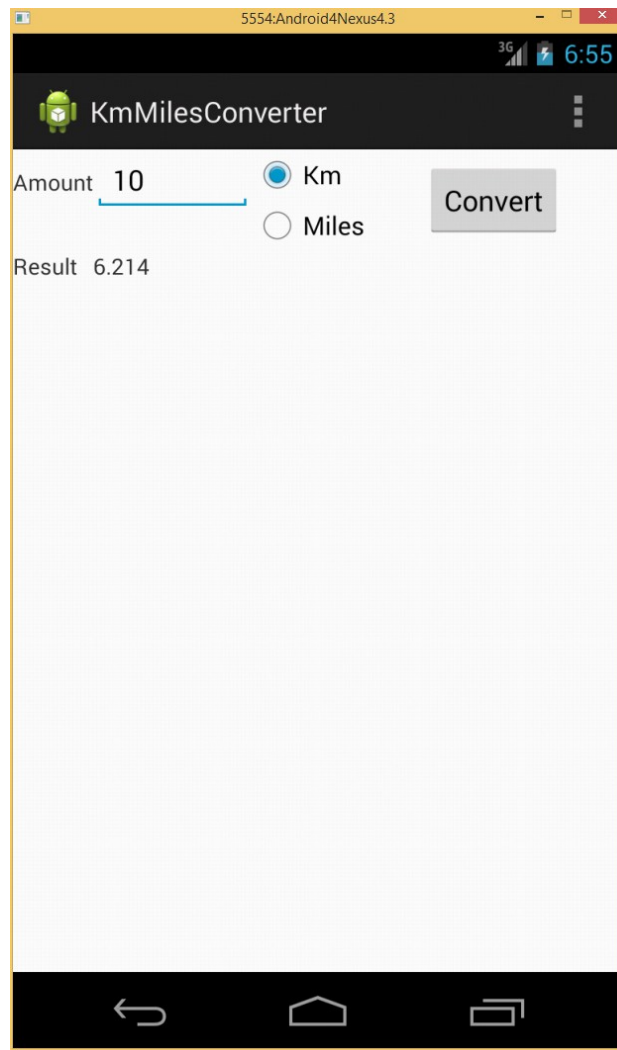
**Figure 7:** Adding imports

3. Note that a good practice is to indent the code and add comments.
4. Now you have to associate the “conversion” function to the “onClick” event of the button. You can do this by assigning the “conversion” value to this property in the property editor of the activity xml editor. The result should execute as something like the figure 8.

### 3. Homework

If you did not finish this example, finish it and modify it to show, for instance, the result in a more verbose way. You could issue a sentence saying something like “X miles are Y kilometres” in the results TextView.

For the next week you should create a small example that lets you input two numbers and may add or subtract them according to a radio button in the UI.



**Figure 8:** Result