Università
della
Svizzera
italiana

**Faculty
of Informatics**

Mobile and Wearable Computing SA
2024-2025

Assignment Report
Assignment 1
Alessio Tommasi

Github Repo:
https://github.com/AlessioTommasi-supsi/USI_MobileWearableComputing/tree/main/Lab12

Component of Android Application:

Activity: represents a single screen with a user interface.
Example done in Lab01:

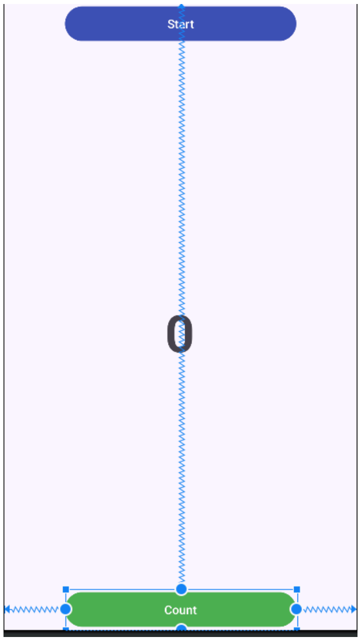
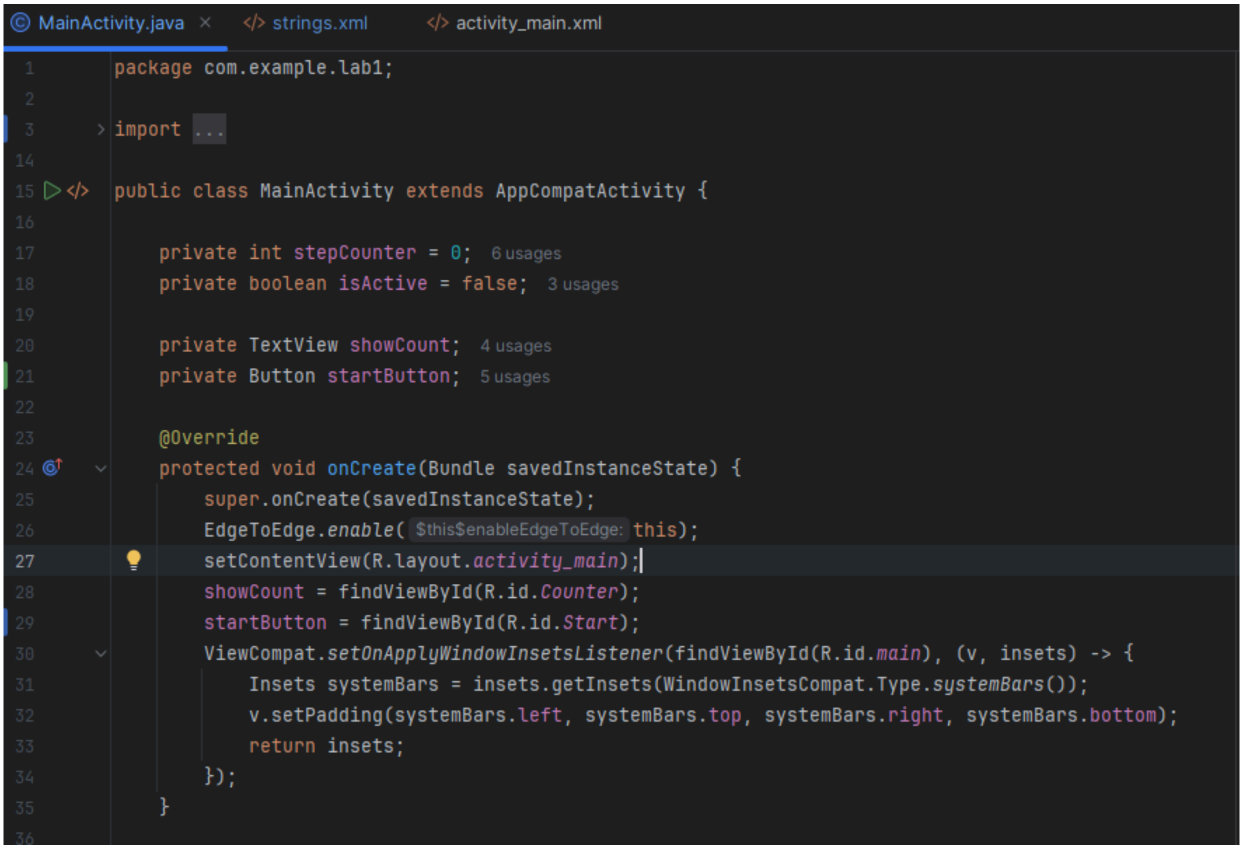


Fig1 Location: /res/layout/activity_main.xml



```
1   package com.example.lab1;
2
3   import ...
14
15  public class MainActivity extends AppCompatActivity {
16
17      private int stepCounter = 0;  6 usages
18      private boolean isActive = false;  3 usages
19
20      private TextView showCount;  4 usages
21      private Button startButton;  5 usages
22
23      @Override
24      protected void onCreate(Bundle savedInstanceState) {
25          super.onCreate(savedInstanceState);
26          EdgeToEdge.enable( $this$enableEdgeToEdge: this);
27          setContentView(R.layout.activity_main);
28          showCount = findViewById(R.id.Counter);
29          startButton = findViewById(R.id.Start);
30          ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
31              Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
32              v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
33              return insets;
34          });
35      }
36
```

Fig2 Location: /java/com/MainActivity.java

The Activity is linked into java on line 27.done only during onCreate to guarantee correct usage.

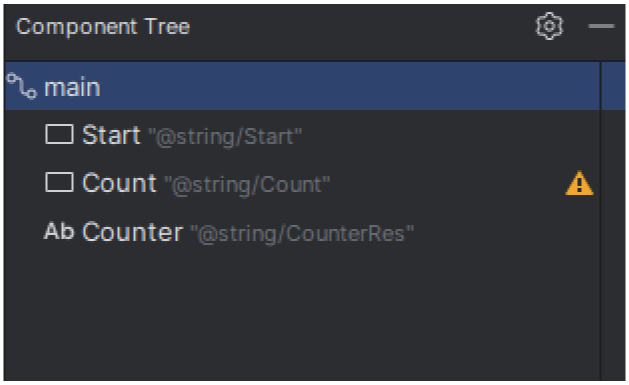
Service: handles background operations without a user interface, we haven t use them Lab01 but in the twitter app they could be synchronizing notifications or downloading new tweets.

Broadcast Receiver: allows the app to respond to system events or events from other apps,
we haven t use them Lab01 but could be the limit of resource usage when battery low or reset counter when i put my app on background.

Content Provider: Allows the sharing of data between different applications. Content providers provide a standard interface for accessing data, such as contacts or media, that can also be used by other apps.

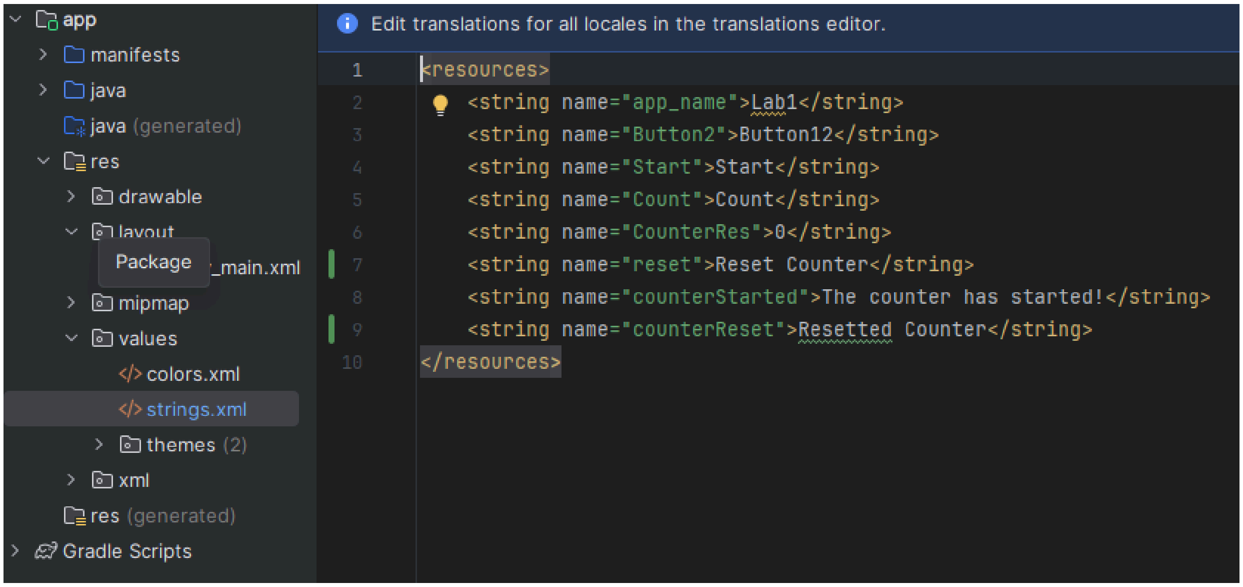
View: A `View` is the base class for all UI elements in Android, representing a single, interactive component. It can be something as simple as a button, a text field, or an image.

- **Example of a View**:
  - A **Button** that the user can click.
  - A **TextView** that displays a string of text.
- an example could be seen in **Fig1** the buttons Start, Count and the TextView Counter



main
- □ Start "@string/Start"
- □ Count "@string/Count" ⚠
- Ab Counter "@string/CounterRes"

Fig3 Location: /res/layout/activity_main.xml
Here i provide the hierarchy structure of the example done in class



```xml
<resources>
    <string name="app_name">Lab1</string>
    <string name="Button2">Button12</string>
    <string name="Start">Start</string>
    <string name="Count">Count</string>
    <string name="CounterRes">0</string>
    <string name="reset">Reset Counter</string>
    <string name="counterStarted">The counter has started!</string>
    <string name="counterReset">Resetted Counter</string>
</resources>
```

Fig4 Location: /res/values/ …
Here we can store all the values that we would like to use in our Application
for example in **Fig3** the TextView with id Counter has his value stored in file *string.xml* with the name CounterRes and has 0 as his value

ViewGroup: A `ViewGroup` is a container that holds multiple `View` objects (or even other `ViewGroup`s), organizing them in a layout. It defines how its child elements are arranged on the screen.

For now we haven't seen this in Lab01