**Assignments: Core 1 Spike report**

## **Dilni De Silva 103616345**

**Repository:**

*Repository using classroom link but has pushing issues and running issues:*

<https://github.com/SDMD-2022/core-1-Dilni-DS113>

*Own repository which pushes correctly and has NONE running issues:*

<https://github.com/Dilni-DS113/Dilni-DS113-New-Core1>

# Goal:

Program an android application that increases, decreases and resets a number on a text view using buttons.

# Resources:

Gap 3 → Incrementing and decrementing

<https://www.demo2s.com/kotlin/kotlin-increment-and-decrement-operators.html>

Gap 6 → Localisation

<https://developer.android.com/guide/topics/resources/localization>

<https://lokalise.com/blog/android-app-localization/>

Gap 8→ String colour change

<https://rrtutors.com/site/answer/How-To-Set-TextView-Text-Color-Programmatically>

Gap 8 → Implementing audio to application

<https://developer.android.com/guide/topics/media/mediaplayer>

<https://blog.mindorks.com/using-mediaplayer-to-play-an-audio-file-in-android>

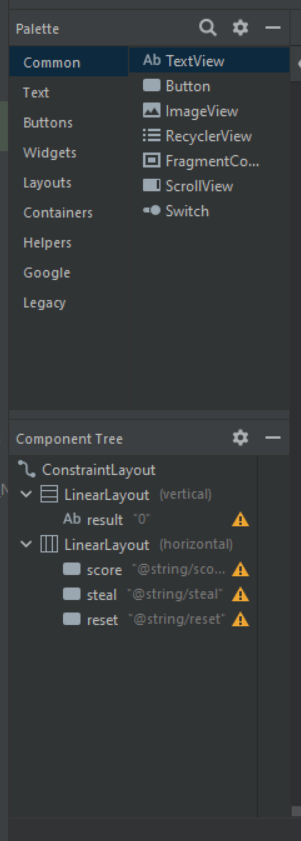
## Skill Gaps and Solutions:

1. **Use linear and constraint layouts:**

Creating at least two layouts linear (horizontal or vertical) and constraint layout that sections off different views.

**Solution:**

* Be able to add a constraint layout from the palette
* Be able to add both linear layouts horizontal and vertical and place them under the constraint layout.
* Be able to add Textview (result), and three buttons (Score, Steal, Reset).

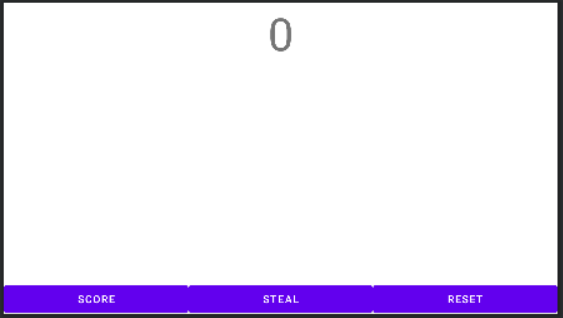
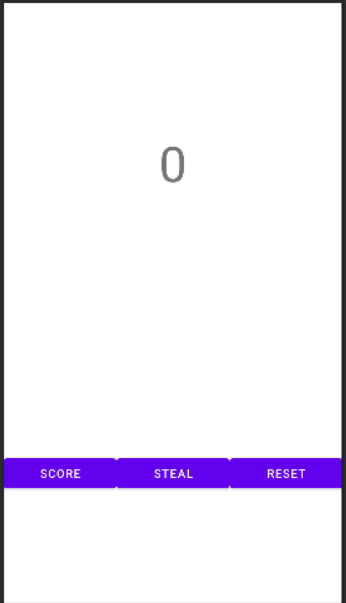


1. **Design layouts for different orientations**

Enable layout of application to switch the orientations of landscape and portrait by creating a new activity main that is landscape.

**Solution:**

* Be able to right-click the activity main.
* Be able to new layout resource file.
* Be able to set the file name.
* Be able to add orientation qualifiers.
* Be able to enable landscape screen orientation.
* Be able to create a file.
* Be able to replicate the same layouts and views in the landscape orientation.



1. **Writing/using appropriate listeners**

Program using Kotlin code the functionality of views using listeners, which will give an output when the user interacts with a view.

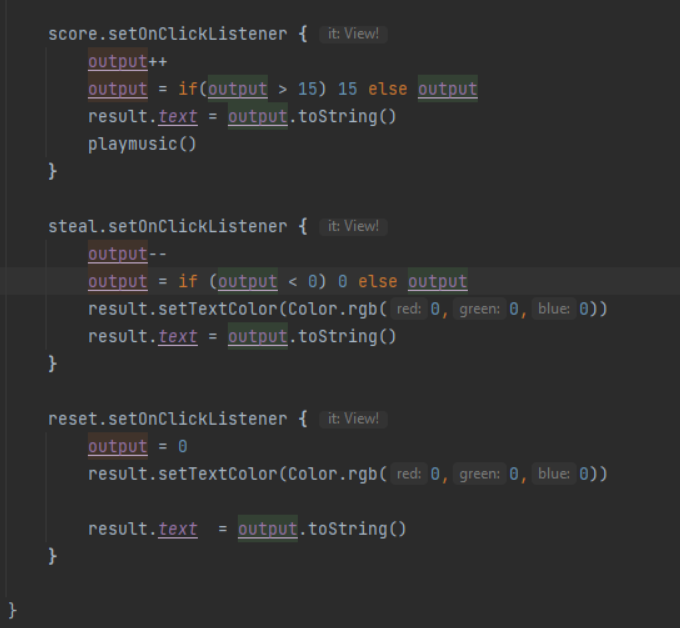
**Solution:**

* Be able to create a score, steal, reset and result in variables and assign them to the same name view initialised in the activity main.
* Be able to create an output variable, type integer and initialise it to 0.
* Be able to create three sets of click listeners that set each button

*Reference: Incrementing and decrementing*

[*https://www.demo2s.com/kotlin/kotlin-increment-and-decrement-operators.html*](https://www.demo2s.com/kotlin/kotlin-increment-and-decrement-operators.html)

* + Be able to create a score button set on click listener increment output variable and print out the variable to result text view and call play music function.
  + Be able to create a steal button set on click listener decrement output variable and print out the variable to result text view and continue to set text colour as black.
  + Be able to create a reset button set on click listener to re-initialised output variable to 0, which will print out the variable to result in text view and continue to set text colour as black.

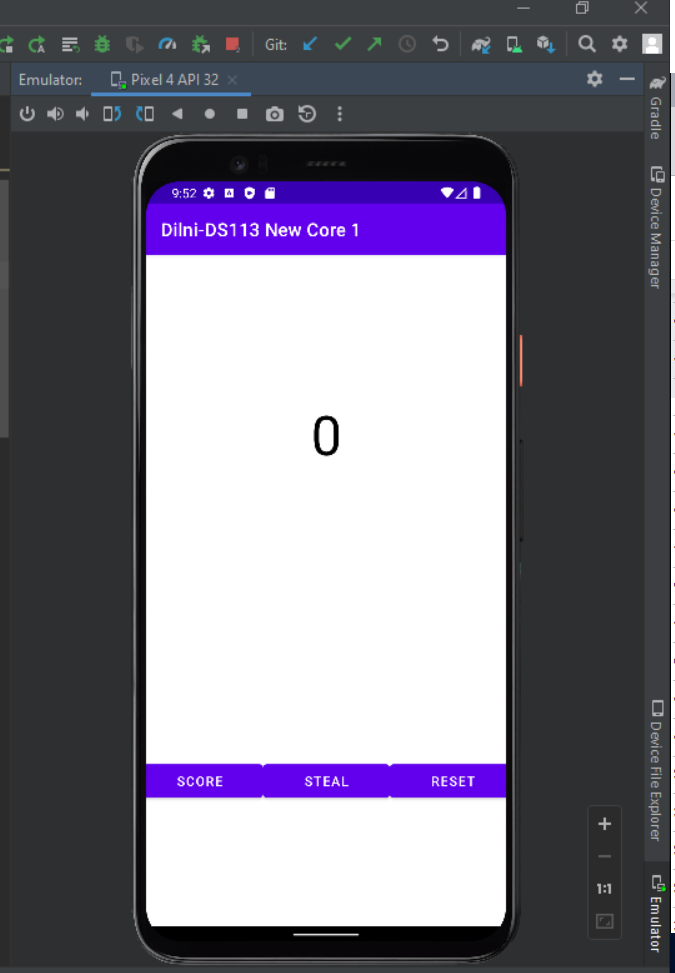


1. **Command of IDE**

Running the emulator that runs a virtual device, which builds the core 1 app increasing, decreasing and displaying the results as an integer displayed on a text view.

**Solution:**

* Be able to select a device from a device manager and start up the device on the emulator.
* Be able to locate the power button on the emulator and turn on the device.
* Be able to run the app by pressing the run button either from the top navigation bar run menu option or pressing the green play button.

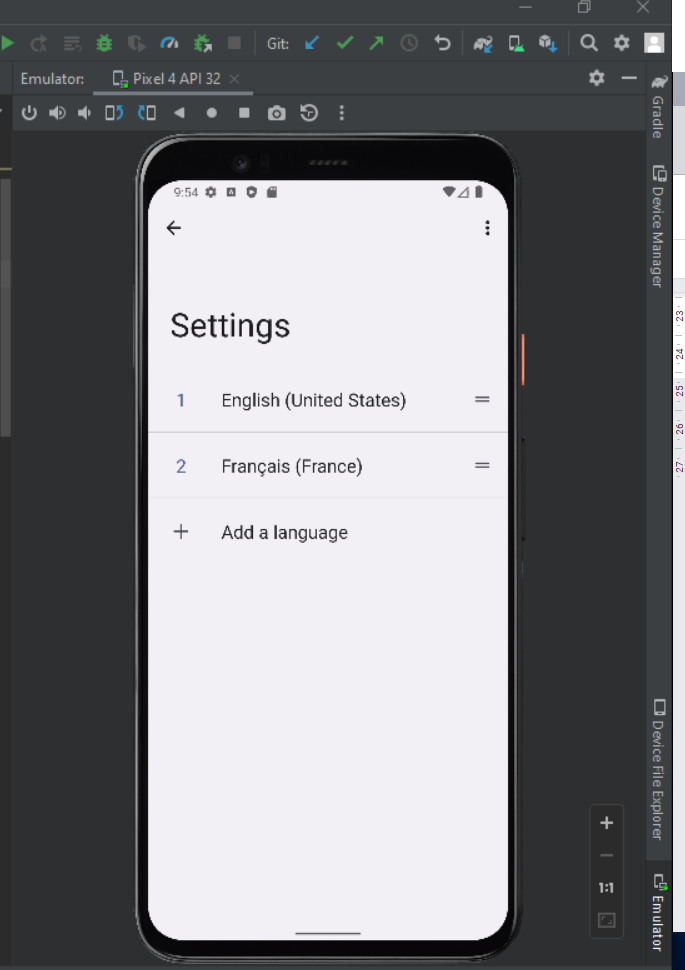
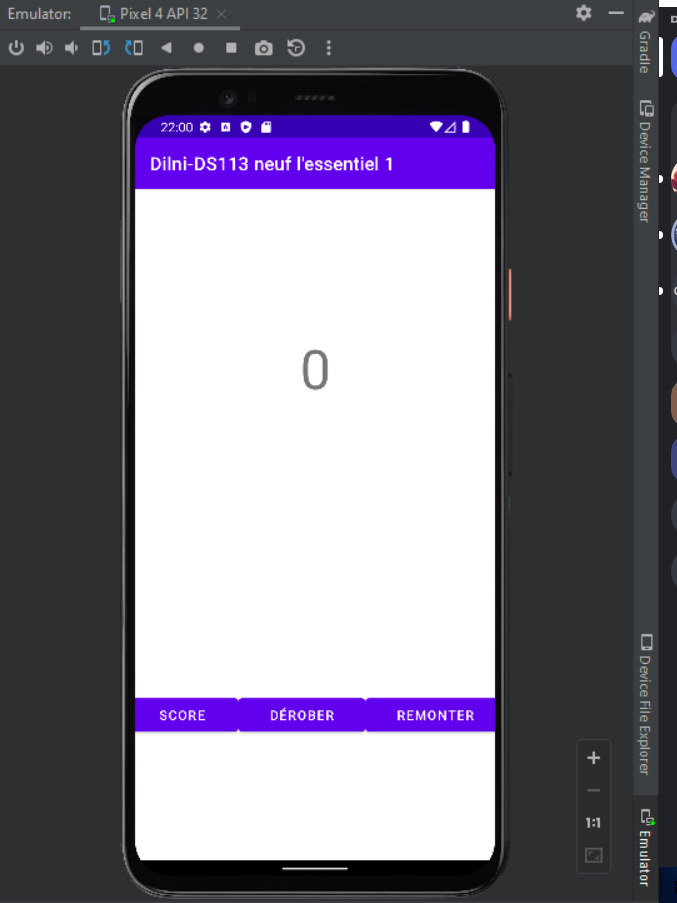


1. **App usable in multiple languages**

By changing the settings in the emulator the app can run in more than 1 language.

**Solution:**

* Be able to start the device on the emulator
* Be able to open settings and navigate to the language section of the settings menu
* Be able to click on the Add language option
* Be able to search and add required languages
* Be able to potion the first preferred language to number one by moving the language block to the top position than the second, third, and fourth.



1. **Incorporate localisation**

To enable an app to be used in multiple languages using string localisation and externalization.

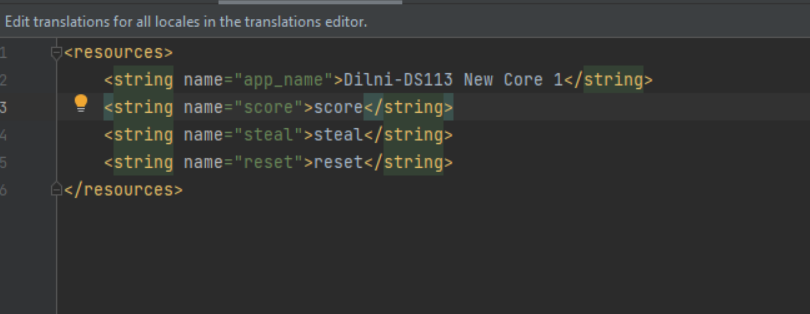
*Reference: Gap 6 → Localisation*

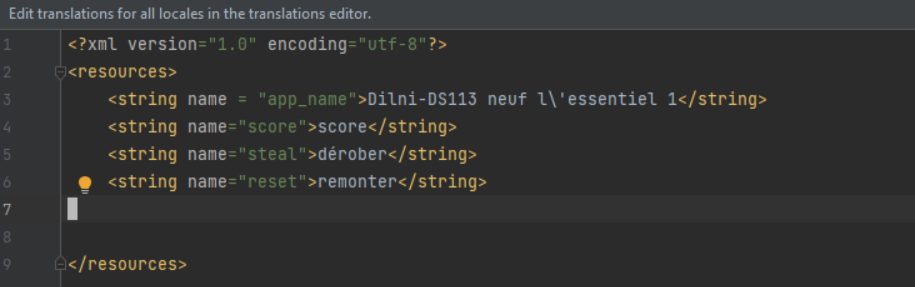
<https://developer.android.com/guide/topics/resources/localization>

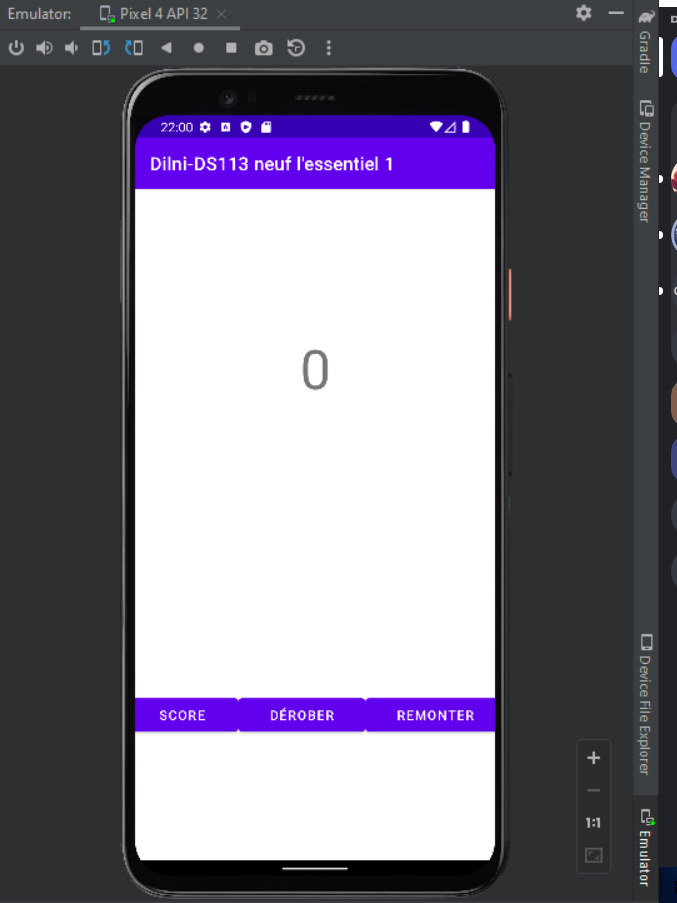
<https://lokalise.com/blog/android-app-localization/>

**Solution:**

* Be able to open the res folder.
* Be able to create a new values folder
* Be able to add the local qualifiers on the values folder
* Be able to select a preferred language, French
* Be able to select a preferred region France
* Be able to create a strings.xml file.
* Be able to make the same string elements with the same English name but the string has to be translated into the French language.

****

****



**String externalization and Localization:**

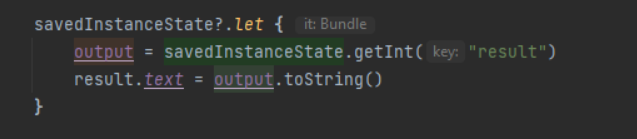
String externalization in programming is defined as keeping another string in another file. Such as the string.xml file in the values folder which holds string labels of views such as the buttons and text view which can be accessed by the main activity file. This helps in localization as it allows different value folders to have an alternate string.xml file under different languages which can be located and used when the language is set to a specified language in the mobile phone.

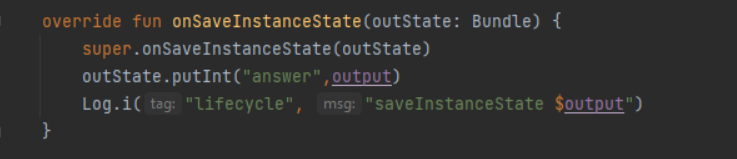
1. **Handle saving of state**

Using a save state instance to save data such as the score stays the number entered in the activity even when the device is roasted from portrait to landscape and landscape to portrait.

**Solution:**

* Be able to code an override on the save instance state function, which saves the output variable.
* Be able to use the let function to initialise the output to the saved instance state.
* Be able to output the output variable to the result text view.





1. **Follow the documentation to implement the adding audio concept and colour change**

Download and add an audio file and play that audio when the score reaches a score of 15 and the score colour changes to green.

**Solution:**

* Be able to first read “How to Set Text Colour programmatically”
* Be able to first read “MediaPlayer overview” and “Using MediaPlayer to play an Audio File in Android” to understand how to implement the code which will use a download audio file and play it when the score reaches 15.

*Reference: Gap 8→ String colour change &* Implementing audio to application

[*https://rrtutors.com/site/answer/How-To-Set-TextView-Text-Color-Programmatically*](https://rrtutors.com/site/answer/How-To-Set-TextView-Text-Color-Programmatically)

[*https://developer.android.com/guide/topics/media/mediaplayer*](https://developer.android.com/guide/topics/media/mediaplayer)

[*https://blog.mindorks.com/using-mediaplayer-to-play-an-audio-file-in-android*](https://blog.mindorks.com/using-mediaplayer-to-play-an-audio-file-in-android)

* Be able to create a function called play\_music, then initialise an if statement with a condition that if the output variable is equal to 15 then using relevant documentation the following is archived:
  + Be able to the text view result sets the text colour to light green using colour.rbg() function.
  + Be able to download an audio file.
  + Be able to create a Directory folder and name the folder raw and add the downloaded audio file to the folder.
  + Be able to create a Mediaplayer variable that would be using the audio file called beep\_short and then start the audio variable using the start function.



1. **Use Logs**

Used to display messages of different stages of the lifecycle of the application build.

**Solution:**

* Be able to create 6 new functions onStart, onResume, onPause, onStop, onDestroy, onRestart.
* Be able to create a log function Log.i() in the life cycle functions each including a message and a tag.
* Be able to create a Log.i() to create function and save statefunction



1. **Testing**

Use provided code with a test case to test for errors in the application. Each test cases test for functionality in certain features in the application.

**Solution:**

* Be able to copy provided code into an android test folder and file.
* Be able to press green arrows next to the main activity in the test case file and run test cases.
* Be able to watch test cases and edit not passing test cases.