KOTLIN

Research Document

Abstract

This research document informs about the new way of making android apps with Kotlin. This research is a part of the App-controlled Self-balancing Robot.

Eveline Ververgaert

28-08-2017

Version management

Version	Description	Date
0.1	Setup	14-08-2017
0.2	Sub question 1	14-08-2017
0.3	Sub question 2	20-08-2017
0.4	Foreword, sub question 3	21-08-2017
0.5	Conclusion and recommendation, summary	28-08-2017
1.0	Spelling and grammar checks	29-08-2017

Summary

Kotlin is a brand new programming language build by JetBrains for android app development and potentially good enough to replace Java someday. Many developers, whom tried Kotlin so far, have decided to cross over to the Kotlin side. Because of these reasons this article looks into how Kotlin works and if it’s good enough to use for the Self-balancing App-controlled Robot (SAR) project (see also Foreword).

This research consists of 3 parts. First Kotlin is inspected by looking at thoughts of other developers by looking into published pros and cons. This clears up how it could possibly be better (than Java). Next code is inspected specifically for an Android app, because it is a requirement for the SAR project. By building a simple and flawed calculator app some basics will be covered like how to build an app with Android studio and how to make variables, functions and classes in Kotlin. Lastly, it’s important to be able to test your work (also a requirement for the SAR project). Kotlin being a relatively new language, could’ve had the issue of not having a testing framework yet. However due to some pros described in the Pros and cons chapter, there is a way to test Kotlin easily.

The conclusion shows whether Kotlin is indeed the new Java for app development or not. It is expected with all new things, it is possible it still has plenty of bugs and errors which JetBrains will have to fix. However this article, no matter the cons, roots in favor of Kotlin.

Foreword

This document is about research on the new programming language Kotlin. This research is done for the project Self-balancing App-controlled Robot. This project requires an app for Android phone to control the robot. This research document is written for the course Special Topic semester 2 2017 at Otago Polytechnic IT school.

The project team exists out of two members, Kong and Eveline. Two international students from the Netherlands. The Kotlin research has been carried out by Eveline. The complexity of the research was low, because most information was easy to understand. Building and playing with the POC app (see chapter Research) was fun and easier than expected. Easier in this case is positive, because learning a new language can bring heaps of troubles.

Glossary

Word	**Definition**
POC	Proof of Concept
Java, C, C++, C#	Programming languages
Lib	Library
Unit testing	Testing of units of code through a testing framework.
Beta	In Software terms it means a program which is still undergoing testing and has not been released officially yet. However it is available to a particular group of users. This will help test the software on a greater scales as well as different computers.
GUI or UI	(Graphical) User Interface
Visual Studio	Developer software by Microsoft
Testing framework	An environment to run test cases and make them automated, as well as providing a report on failing and successful test cases.

Table of content

Table of figures

Introduction[1]

Kotlin is a programming language based on Java and build by the company JetBrains. Recently Google started promoting Kotlin to be used in Android Studio for app development.

During the project about the app-controlled self-balancing robot Kotlin will be used to expand knowledge with a new language. In order to get started Kotlin shall first be tried in this research. The conclusion will decide if Kotlin is used in the project.

Goal

The goal of this research is to become familiarized with Kotlin in terms of coding and testing. This will help determine if it should be used in the project.

Research questions

Main question

- What can Kotlin mean for Android app development?

Sub questions

- What are the pros and cons of Kotlin?
- How do you code with Kotlin?
- How do you test a Kotlin app?

Research method / set up[2]

This research will involve one POC to understand the building of an application for Android 4.x or higher (work place method). This POC will also be used to understand how to test Kotlin best (lab method). In order to make the first step other people's work are reviewed as well (Field method).Furthermore, to answer the first sub question the internet will be used (Lib method).

During this research…

- …the POC will be very simple with enough elements to try testing. In other words, it will not be made pretty.
- …Android Studio 3.0 or lower with Kotlin plugin.
- …the testing shall not be made to cover all code.
- …the meaning of testing in this context is unit testing.

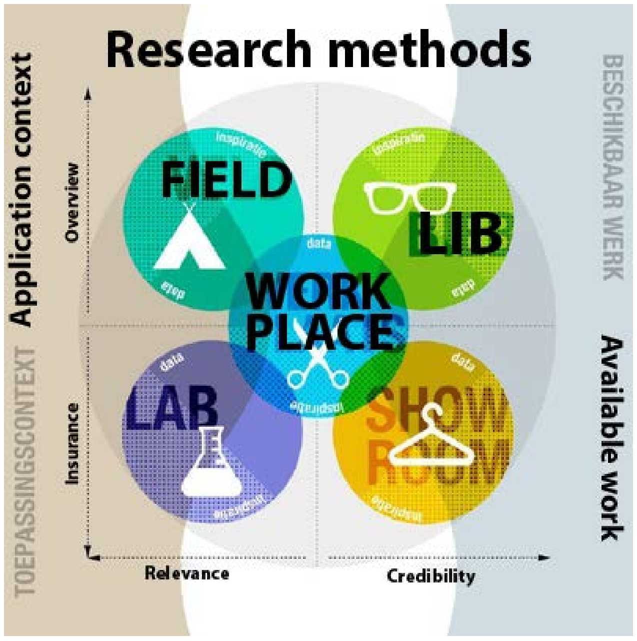


Figure 1 - Research Methods according to Fontys lectures

Research

This chapter provides the research and answer to the sub questions.

Pros and cons[3][4][5][6]

Nowadays many app developers use Java for Android apps. However, Kotlin is reasonably new and most developers, whom used Kotlin so far, seem to be rather happy with the statically-typed programming language. Because Java the official language to build apps with Kotlin will be compared to this language:

Pros

- Kotlin needs less source code and does not lose its efficiency
- Kotlin is easier to read than Java.
- Kotlin can control null exceptions (this makes it safer)
- Kotlin is interoperable with java libraries and java source code
- Kotlin is considered an alternative to Java
- By being more compact, Kotlin source code needs less quality assurance time (testing).

Cons

- Kotlin is new, which means less tutorials, documentation or other forms of help is available
- Kotlin Standard Library and runtime increases the size of your .apk file.
- While Kotlin is easier to read, it is also trying to be smaller, which means some parts of code can become more difficult to decipher because many things are happening in a little part of code.
- Kotlin makes for slower compilation speed.

Conclusion

Most articles praise Kotlin for being in several ways better than Java. Kotlin could be a challenge because it is new and has less help from the internet. Furthermore, during the project of App-controlled Self-balancing Robot, the app will be built as a student project. The project is not meant to be published on the android market, therefor size of the .apk file is of no concern to us. Kotlin being slower in compiling is another con which is of little concern.

Programming in Kotlin[1][7]

Requirements

In order to develop applications with Kotlin, the tool programming tool required which can work with Kotlin. The Kotlinlang website provides 4 options. For this experiment Android Studio has been chosen, because of previous experience and reliability. It's very important that it is Android Studio version 3.0 or higher. All previous versions do not include Kotlin. However, currently the 3.0 version is a Beta version. Which means the stable versions of Android Studio do not include Kotlin. However, they can get a Kotlin plug-in.

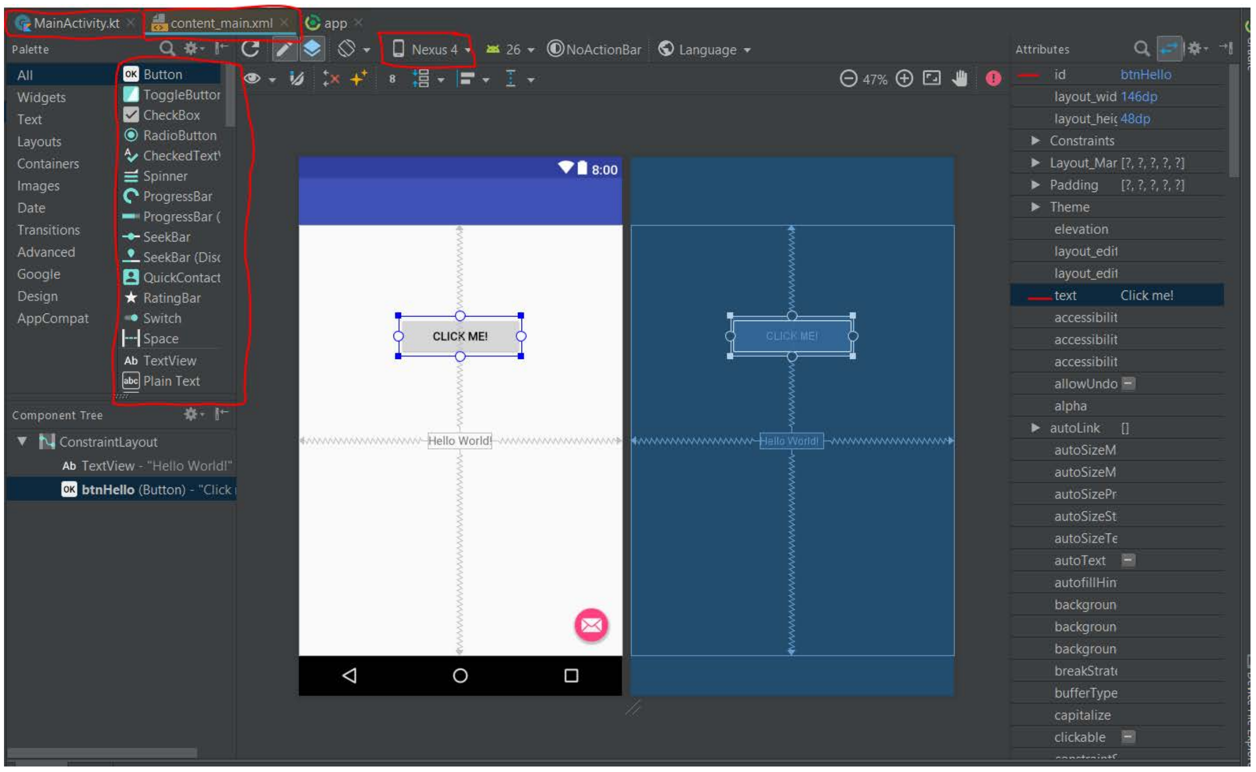
Furthermore, to test a developed application it is handy to have an android phone with the correct OS version. If this is not possible Android Studio provides a way to use a Virtual Device.

Hello world![8]

The following Proof of Concept has been built with the support of YouTube tutorial[9].

After installing the preferred IDE it is time to start a project. For this project I installed Android Studio the latest Beta version in order to get the build-in Kotlin advantages.

Android Studio will help you create a new project. Give it a proper name, make sure you select the right Android OS version and don't add anything you do not need. For this little project names will be less relevant, however one must not forget to give all variables, activities, etc. a proper quickly recognizable name in order to improve code readability. When all of this is done, Android Studio will start preparing the project.



id	btnHello
layout_wid	146dp
layout_heig	48dp
Constraints	
Layout_Mar	[?, ?, ?, ?, ?]
Padding	[?, ?, ?, ?, ?]
Theme	
elevation	
layout_edit	
layout_edit	
text	Click me!
accessibilit	
accessibilit	
accessibilit	
allowUndo	
alpha	
autoLink	[]
autoSizeM	
autoSizeM	
autoSizePr	
autoSizeSt	
autoSizeTe	
autoText	
autofillHin	
backgroun	
backgroun	
backgroun	
breakStrat	
bufferType	
capitalize	
clickable	

Figure 2 - Android Studio

At start up you'll find yourself in a similar environment as shown on the figure above. You can drag and drop items into your design. Make sure to give these items a proper ID and text on the right side of the screen. It's also handy to know you can change the screen size by changing the phone kind (at the top bar, see "Nexus 4"). Your code will go in the MainActivity.kt, but first let's make sure your project properly prepared. Click on the following file on the left side of your screen:

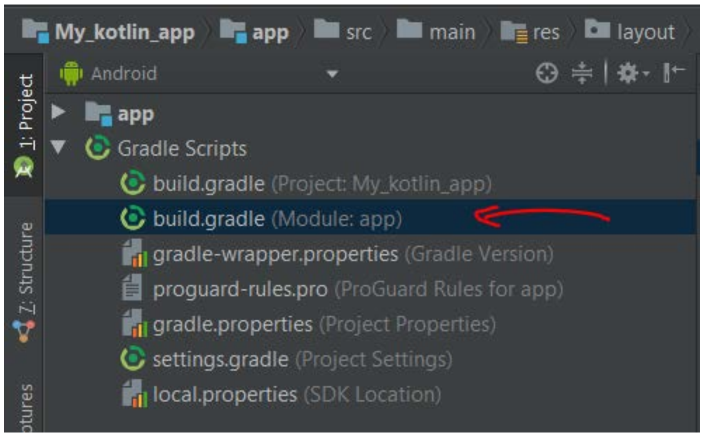
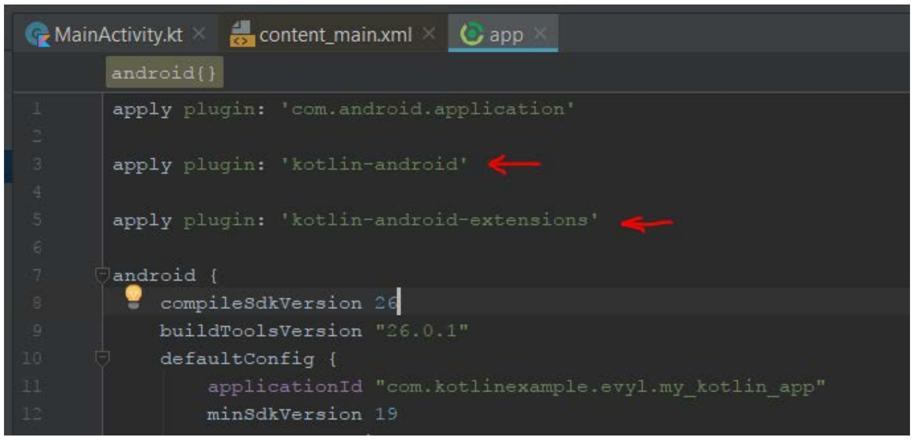

My_kotlin_app
app
src
main
res
layout
Android
1: Project
7: Structure
app
Gradle Scripts
build.gradle (Project: My_kotlin_app)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)

Figure 3 - Android Studio - scripts

Make sure these two lines (see arrows below) are included. This check is only necessary whilst Android Studio 3.0 is Beta version.



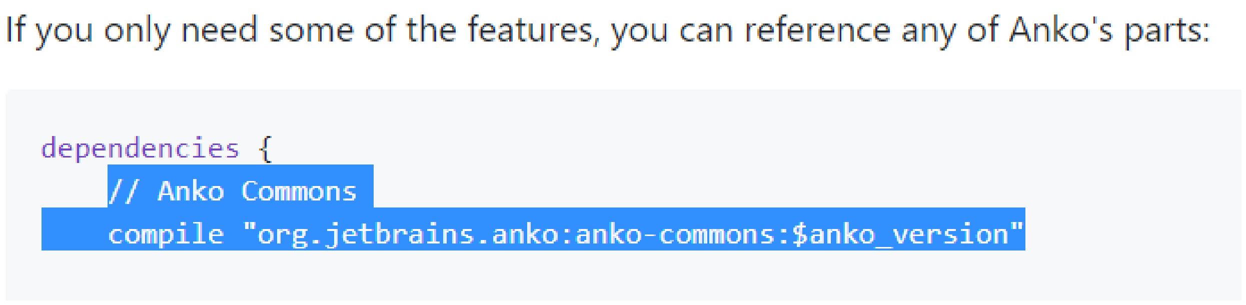
```
apply plugin: 'com.android.application'

apply plugin: 'kotlin-android'

apply plugin: 'kotlin-android-extensions'

android {
    compileSdkVersion 26
    buildToolsVersion "26.0.1"
    defaultConfig {
        applicationId "com.kotlinexample.evyl.my_kotlin_app"
        minSdkVersion 19
```

Figure 4 - Android Studio - extensions

Adding a library

It's easy but also important to know how to add libraries, because they can add functionality to your project. Anko is an example library which is used a lot by fellow Kotlin Developers.

The Anko[10] site provides the line of code you need to add and change the $anko_version to the right version:

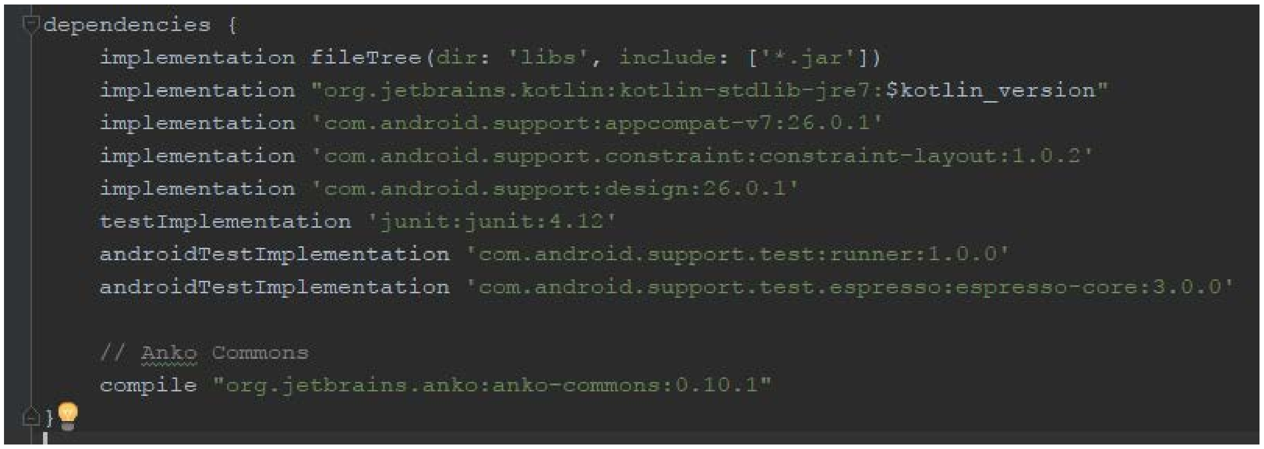


If you only need some of the features, you can reference any of Anko's parts:

```
dependencies {
    // Anko Commons
    compile "org.jetbrains.anko:anko-commons:$anko_version"
```

Figure 5 - Anko dependency

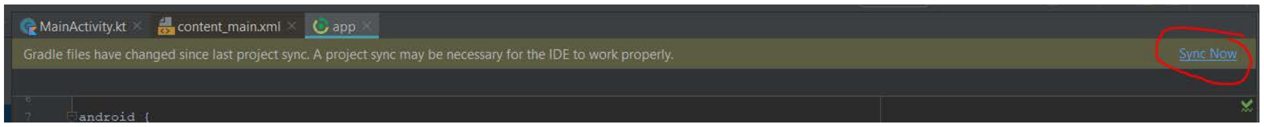
In the build.cradle file you worked in before, add the code like so:



```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jre7:$kotlin_version"
    implementation 'com.android.support:appcompat-v7:26.0.1'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.android.support:design:26.0.1'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.0'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.0'

    // Anko Commons
    compile "org.jetbrains.anko:anko-commons:0.10.1"
}
```

Figure 6 - Android Studio - adding anko library

Click on "sync now" at the top of the screen:



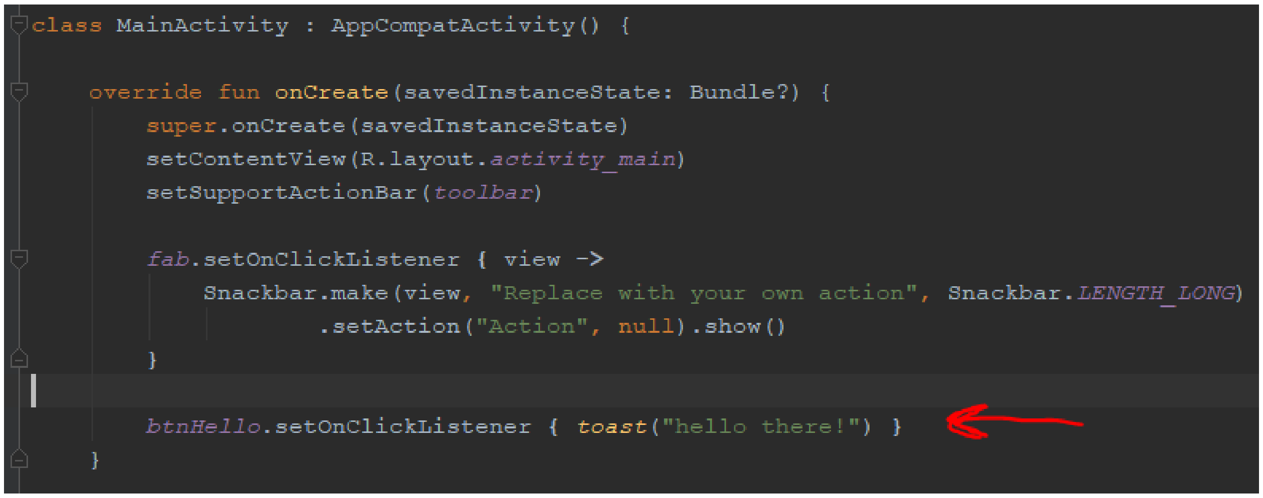
Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
android {
```

Figure 7 - Android Studio - Sync Now

Code

Inside the MainActivity.kt you can put your code. For this simple hello world program all we want is to add a button and make it say something of your liking: <buttonID>.setOnClickListener { //your code }



```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        setSupportActionBar(toolbar)

        fab.setOnClickListener { view ->
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show()
        }

        btnHello.setOnClickListener { toast("hello there!") }
    }
```

Figure 8 - Android Studio - Main Activity code

Result

Figure 9 shows what has been built in a short amount of time. The button is available to click, there is a little bit of text in there “hello world” and once clicked on the button it shows a little toast at the bottom of the screen.

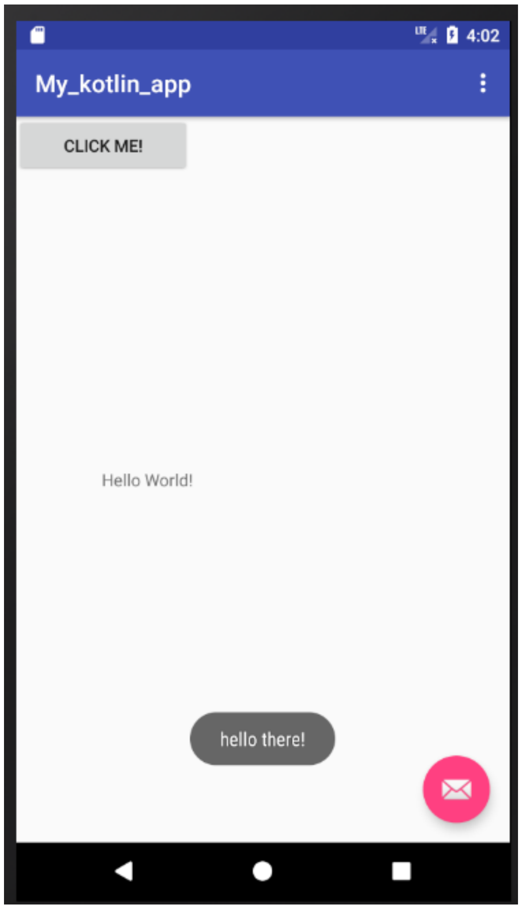


Figure 9 - Android Studio - virtual device displaying Hello World

Just a little more…
Let's enhance the complexity a little bit. This research is about trying out Kotlin, so let's build a simple calculator app.

Steps taken:

1. New project -> named MyCalculator
2. Placed all elements in the GUI (buttons, textViews and Plain text).
3. All these elements need a proper ID and text.
4. Also make sure you constraint them (else they will be pasted to the upper left corner when you run it). For now I will use "Top_toTopOf", "Bottom_toBottomOf", "Right_toRightOf" and "Left_toLeftOf". These are set to "parent" as shown figure 10. Once you've set these to parent, you can drag them to any spot and it will stick where you like.
5. Next is coding. You are going to need a new Kotlin class file, I called Calculator. One way to make a new class file: Help > find action, type "kotlin class", then the first option should be "kotlin file/class".
6. Classes look rather similar to other languages[11]. Personally, as a C and C++ programmer it seems that building a constructor is rather different. For the calculator we need NO constuctor. (see figure 11)
7. However we do need functions, four to be exact: Add, Substract, Divide and Multiply. A function is created like so: Do this for all four functions.
8. Now we go back to the main and make sure we link the buttons to the right function:

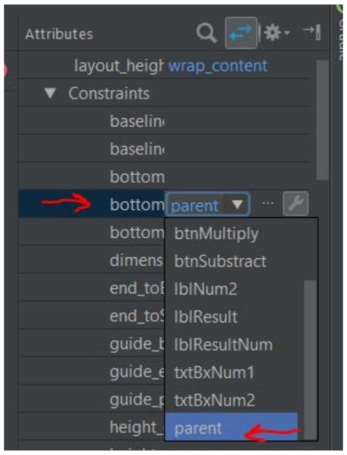
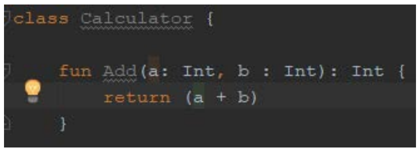
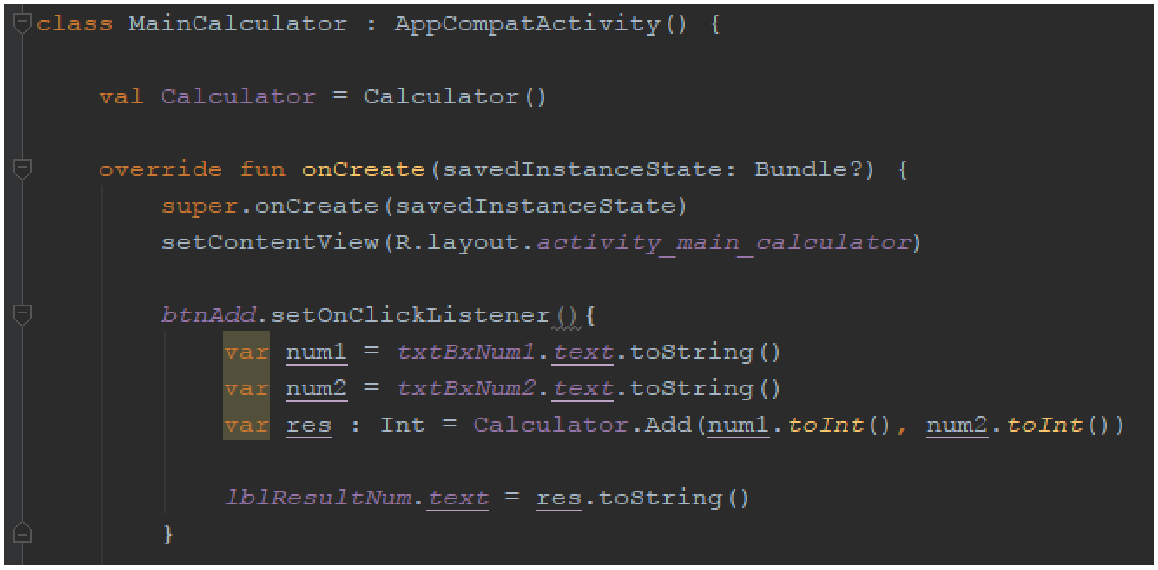


Figure 10 - Android Studio - Constrains



```
class Calculator {

    fun Add(a: Int, b : Int): Int {
        return (a + b)
    }
```

Figure 11 - Android Studio - Class and function



```
class MainCalculator : AppCompatActivity() {

    val Calculator = Calculator()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main_calculator)

        btnAdd.setOnClickListener(){
            var num1 = txtBxNum1.text.toString()
            var num2 = txtBxNum2.text.toString()
            var res : Int = Calculator.Add(num1.toInt(), num2.toInt())

            lblResultNum.text = res.toString()
        }
```

Figure 12 - Android Studio - Main activity code - Calculator

9. And that is it! This is the result:

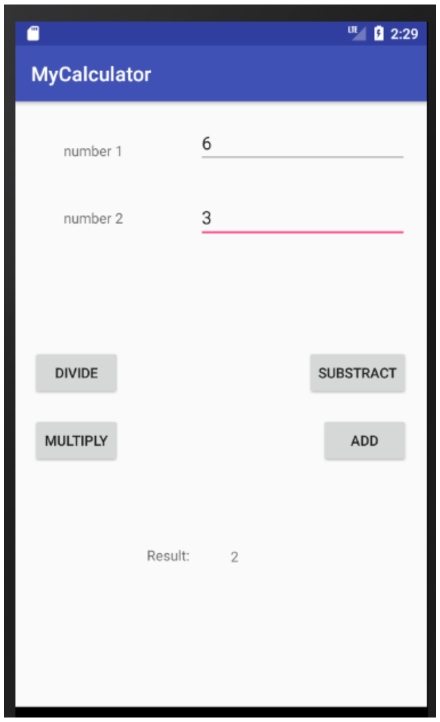


Figure 13 - Android Studio - Virtual device displays Calculator

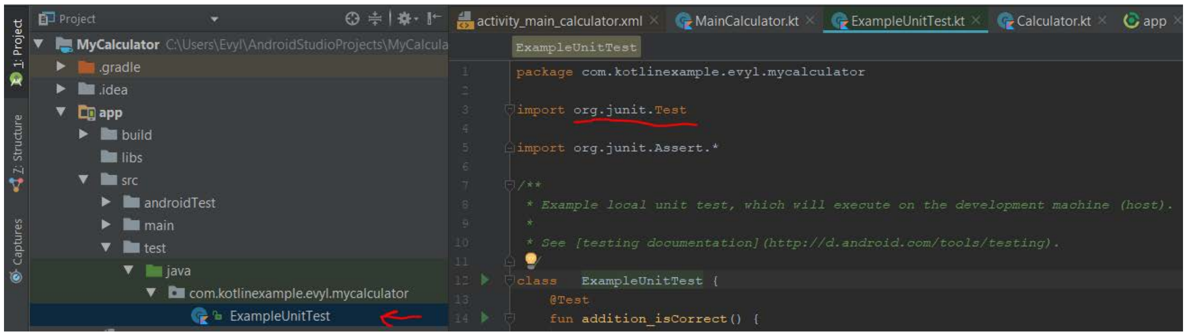
My conclusion from this experience is that Kotlin is in general not a difficult language to learn. As a C++ and C developer I see Kotlin as a version of Java which has been simplified and seems to be a bit more like Visual Studio with C# projects.

Testing a Kotlin Application

Before we can start to test the MyCalculator project from previously, first a testing framework is required. This will help with setting up unit test code, by providing e.g. functions to inspect results like “assert.areEqual(expected, actual)” (Visual Studio C# example).

As previously stated, Kotlin is a language that can work with Java. Which should mean it could possibly work with a testing framework build for Java.

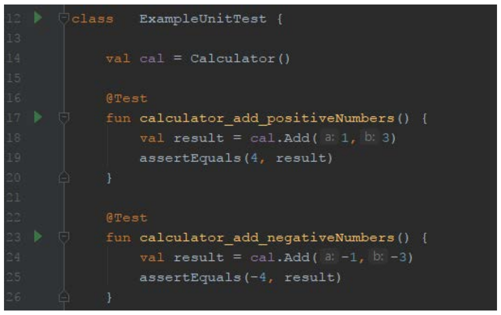
Android Studio has already build-in a testing framework called jUnit.



```
package com.kotlinexample.evyl.mycalculator

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class  ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
```

Figure 14 - Android Studio - Example unit test class

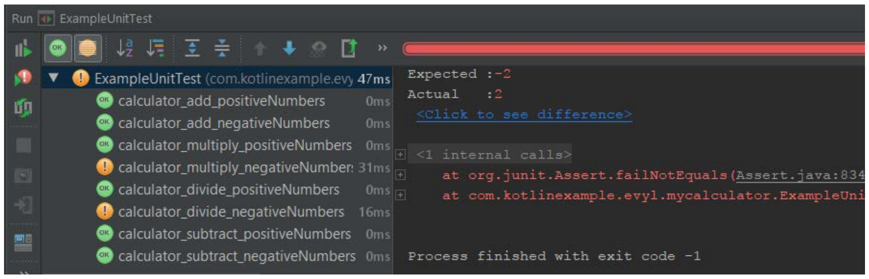
We will use this file to build our test cases in. Here are a few of the test cases build:



```
12 class  ExampleUnitTest {
13
14     val cal = Calculator()
15
16     @Test
17     fun calculator_add_positiveNumbers() {
18         val result = cal.Add( a: 1, b: 3)
19         assertEquals(4, result)
20     }
21
22     @Test
23     fun calculator_add_negativeNumbers() {
24         val result = cal.Add( a: -1, b: -3)
25         assertEquals(-4, result)
26     }
```

Figure 15 - Android Studio - Calculator test cases

Next, simply press the play icon on the side bar. If you want to run all tests click on the one on line 12 next to “class”, else press the icons at a specific test case to run just the one. This will lead to result:



```
ExampleUnitTest (com.kotlinexample.evy 47ms
    calculator_add_positiveNumbers        0ms
    calculator_add_negativeNumbers        0ms
    calculator_multiply_positiveNumbers   0ms
    calculator_multiply_negativeNumber:  31ms
    calculator_divide_positiveNumbers     0ms
    calculator_divide_negativeNumbers    16ms
    calculator_subtract_positiveNumbers   0ms
    calculator_subtract_negativeNumbers   0ms
```

```
Expected :-2
Actual   :2
 <Click to see difference>

<1 internal calls>
    at org.junit.Assert.failNotEquals(Assert.java:834
    at com.kotlinexample.evyl.mycalculator.ExampleUni

Process finished with exit code -1
```

Figure 16 - Android Studio - Unit testing results

It’s easy to analyze in Android Studio what went wrong and what not. So now the developer can fix his or her bugs/errors.

Conclusion and recommendation

The main question for this research was:

- What can Kotlin mean for Android app development?

An overall perspective after following the 3 parts of this research (pros and cons, hello world app and testing) tells me that Kotlin is rather easy to work with even as a C, C++ developer. The pros provided came in handy with the testing framework and coding as well (as it makes it easier and smaller).

Most of the cons are not of importance to the SAR project. It could be an issue there are no heaps of documentation on Kotlin, however there is more on how Java does things. The two language are very similar, which means in some cases you could try to interpret from Java documentation. I have noticed during the build of the calculator that the building time takes a while. This is an issue that the developer cannot help. It’s up to Android Studio and JetBrains developers how to fix that. Furthermore, the SAR project won’t be building a huge app as it is a small project for half a semester. Therefore size of the .apk is of little relevance. In case it is of relevance, I recommend to remove all useless libraries or only add parts of libraries like Anko has divided it’s library into four parts as well.

Once you are coding Kotlin in Android Studio, it is most difficult to find the right code online. If you wish to find just Kotlin code, there is just enough to help you get by. If you wish to find Kotlin code specifically for Android Studio (e.g. get text from a textView) it is more difficult and this is the part where looking at Java code is handy as well as playing with functions given by Android Studio.

Testing Kotlin code is very easy. Android Studio provides you with an example test class, so all you have to do is make proper test cases. Even making a new test class is easy now, because it is mostly a matter of copying all libraries.

References

[1] Kotlin. (n.d.). Retrieved from Kotlinlang: https://kotlinlang.org/

[2] Methoden Toolkit HBO-i. (2016, 02 16). Retrieved from HBO-i: http://onderzoek.hbo-i.nl/index.php/Methoden_Toolkit_HBO-i

[3] Android - Java ns. Kotlin: Why you should use Kotlin? (n.d.). Retrieved from Techiediaries: https://www.techiediaries.com/java-vs-kotlin-why-use-kotlin/

[4] Wodehouse, C. (n.d.). *Java vs. Kotlin: Improving Android Development*. Retrieved from Upwork: https://www.upwork.com/hiring/mobile/java-vs-kotlin-android-programming-languages/

[5] Jonusaite, P. (2017, 04 08). *What are some pros and cons of the Kotlin programming language?* Retrieved from Quora: https://www.quora.com/What-are-some-pros-and-cons-of-the-Kotlin-programming-language

[6] Thornsby, J. (2016, 12 12). *Java vs. Kotlin: Should You Be Using Kotlin for Android Development?* Retrieved from Envatotuts+: https://code.tutsplus.com/articles/java-vs-kotlin-should-you-be-using-kotlin-for-android-development--cms-27846

[7] Getting started with Android and Kotlin. (n.d.). Retrieved from Kotlinlang: https://kotlinlang.org/docs/tutorials/kotlin-android.html

[8] Examples. (n.d.). Retrieved from Kotlinlang: https://try.kotlinlang.org/#/Examples/Hello,%20world!/Simplest%20version/Simplest%20version.kt

[9] Android studio 3 - create hello world app in kotlin. (2017, 05 22). Retrieved from YouTube: https://www.youtube.com/watch?v=-nz-zwfhrLg

[10] Kotlin / anko. (n.d.). Retrieved from GitHub: https://github.com/Kotlin/anko

[11] Classes and Inheritance. (n.d.). Retrieved from Kotlinlang: https://kotlinlang.org/docs/reference/classes.html