

Caleb Millard

CMPT 360 Spring 2023

Assignment 5

Make a Game in a new language from the internet language

Assignment	Due Date	group(s)	Language	Language	Platform
1	Monday, Jan 23	1 & 2	Java	Delphi	Windows
2	Monday, Feb 6	1 & 2	C#	Visual basic	Windows
3	Monday, Feb 27th	3	Javascript		Windows
4	Monday, March, 13	1 & 4	Java	F#	Linux
5	Monday, March 20	4(i think)	Kotlin		Android
6					
7					

This assignment fulfills the following goals:

Group 4(i think): Kotlin

Title:

Sudoku using Kotlin

Problem:

Create a game that is more complicated than tik tac toe in a language that you downloaded from the internet

Documentation:

Run the app and play sudoku click on a container or cell and input a new number after all the cells are filled the app will evaluate and tell you if you are correct

Pseudo Code:

Create containers in a 9 x 9 grid and then fill them with an input of strings, then allow the user to change and add numbers to empty spaces, once the spaces are all filled in check to see if the puzzle is solved by row and then by column then by the box. After this then activate a text that will say if the puzzle is solved or still faulty. If faulty, allow them to change numbers

Imports Java:

```
import android.content.Context
import android.os.Bundle
import android.view.View
import android.widget.*
import androidx.appcompat.app.AppCompatActivity
```

Variables used in the program: Kotlin

Integers: i,j(iterators) i2,j2(iterators for the checkers)

Boolean: permanent(used to define the original numbers on the grid)

Button: Button(to click on each container for a number)

Array: seen(the numbers in the rows or columns that have already been seen)

String: input(string of input integers)

Layout: layout(how the containers are laid out) Llayout(layout for how the checker goes through)

Textview: Text(a text field that the user can see for correct or incorrect notification)

Kotlin Program Start:

```

package com.example.myapplication

//Caleb Millard
//613362
//Title: Sudoku Game in Kotlin
//lab 5 CMPT 360
//Dr.Rick Sutcliffe
import android.content.Context
import android.os.Bundle
import android.view.View
import android.widget.*
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    public inner class Container(var value: Int, THIS: Context?) {
        var permanent = false
        var button: Button

        // initialize the class Container which creates the array of enterable
        fields and then checks them
        init {
            permanent = if (value != 0) true else false
            button = Button(THIS)
            if (permanent) button.text = value.toString()
            button.setOnClickListener(View.OnClickListener {
                if (permanent) return@OnClickListener
                value++
                if (value > 9) value = 1
                button.text = value.toString()
                if (checker()) {
                    Text!!.text = "Correct"
                } else {
                    Text!!.text = "There is a repeated Digit"
                }
            })
        }
    }

    //this function will iterate through each of the fields and make sure each of
    them follows the rules of a correct sudoku
    fun correctchecker(i1: Int, j1: Int, i2: Int, j2: Int): Boolean {
        val seen = BooleanArray(10)
        for (i in 0..8) seen[i] = false
    }
}

```

```

        for (i in i1 until i2) {
            for (j in j1 until j2) {
                val value = table[i][j]!!.value
                if (value != 0) {
                    if (seen[value]) return false
                    seen[value] = true
                }
            }
        }
        return true
    }

//this will call the other checker and it will call each of the rows and areas
to check
    fun checker(): Boolean {
        //checks horizontal lines
        for (i in 0..8) if (!correctchecker(i, 0, i + 1, 9)) return false
        //checks vertical lines
        for (j in 0..8) if (!correctchecker(0, j, 9, j + 1)) return false
        //checks boxes
        for (i in 0..2) for (j in 0..2) if (!correctchecker(
            3 * i,
            3 * j,
            3 * i + 3,
            3 * j + 3
        )) return false
        //only returns true if they are all correct
        return true
    }

//if all the areas are correct will return true
    fun finished(): Boolean {
        for (i in 0..8) for (j in 0..8) if (table[i][j]!!.value == 0) return
false
        return true
    }

//initializing varaibles outside the start function that will be used in the
class
    lateinit var table: Array<Array<Container?>>
    //all variables that are instantiated must have some value or be stated as
null when declaring them.
    var input: String? = null
    var layout: TableLayout? = null

```

```

var Llayout: LinearLayout? = null
var Text: TextView? = null
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    //could be just pasted in one string but for my visual put it in like
this
    input = "3 - 6 8 9 1 5 7 2 " +
            "2 - 1 7 3 5 6 8 4 " +
            "5 7 8 2 6 4 3 9 1 " +
            "8 5 9 4 - 3 1 2 6 " +
            "4 6 3 9 1 2 8 5 7 " +
            "7 1 2 6 5 8 4 3 9 " +
            "1 3 7 5 4 9 2 6 8 " +
            "9 2 4 3 8 6 7 1 5 " +
            "6 8 5 1 2 7 9 4 3 "

    //will be split based on the delimiter " " and will create this an array
from the string input
    val split = input!!.split(" ".toRegex()).dropLastWhile { it.isEmpty() }
        .toArray()
    //this is the array that is being filled
    table = Array(9) {
        arrayOfNulls(
            9
        )
    }
    //forces the array to be displayed and shown as a 9 x 9 grid
    layout = TableLayout(this)
    for (i in 0..8) {
        val row = TableRow(this)
        var j = 0
        while (i < 9) {
            val s = split[i * 9 + j]
            val c = s[0]
            table[i][j] = Container(s[if (0 == '-'.code) 0 else c.code -
'0'.code].code, this)
            row.addView(table[i][j]!!.button)
            j++
        }
        layout!!.addView(row)
    }

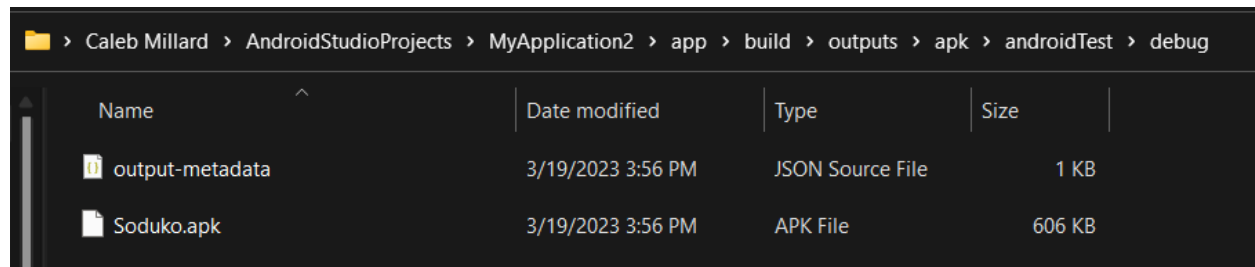
    //this was the most difficult part of this lab since it is dealing with
the layout on screen.

```

```
// this ended up being the most effective way of doing all of this
without major issues.
layout!!.isShrinkAllColumns = true
Text = TextView(this)
Llayout = LinearLayout(this)
Llayout!!.addView(Text)
Llayout!!.addView(layout)
Llayout!!.orientation = LinearLayout.VERTICAL
setContentView(layout)
}
}
```

End of required code

Screenshots:



Caleb Millard > AndroidStudioProjects > MyApplication2 > app > build > outputs > apk > androidTest > debug				
Name	Date modified	Type	Size	
output-metadata	3/19/2023 3:56 PM	JSON Source File	1 KB	
Sudoku.apk	3/19/2023 3:56 PM	APK File	606 KB	

I did not have an android suitable to run the most recent SDK which the build used, but it ran in the editor

Conclusions:

This was a challenging lab because I created it in android studio which is very difficult to use. This app was difficult to create since it was a lot of checking to see if there was correct input; a sudoku is essentially just a giant error checker. This also gave me experience with learning layouts and views in android studio. Kotlin is an interesting language that I can't say I enjoyed too much, but it is similar to Java in terms of syntax. While researching, I found that java and kotlin use the same byte compiler. Sudoku is a fun game to make overall.