Naresh Tech - Kotlin Android Development - Batch 24 Notes

Demo Video Sessions

Day 1	**https://youtu.be/yVznBG1aRqI**
Day 2	**https://youtu.be/sBohN-P8e3A**
Day 3	**https://youtu.be/38m_szSdgTU**
Day 4	**https://youtu.be/ZYWGpSPTypU**
Day 5	**https://youtu.be/xYRzgmsd_ms**

Aug 19, 2024

- Introduction to android
- What will we get out of the course ?
- Duration
- Syllabus

Introduction to Android & History - Kotlin

Aug 20, 2024

Agenda

- Installation of Android studio, Emulator and Connecting the Physical device to android studio.
- How to create your first project in android studio ?

Android studio Installation:

- Download android studio from here
- Follow the installation instructions for installing android studio

FAQ

Do we need to install JDK before installing android studio ?

- No, Android studio comes with open JDK

Install An Emulator

- Emulator is an android virtual device that helps you run the apps and visualize the output.
- Please follow the instructions here in the link

How to Create a Project in Android Studio ?

- Create a project | Android Studio

Connect your Physical Device to Android studio to run Apps on Android studio
Run apps on a hardware device | Android Studio

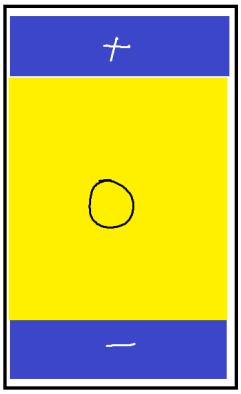
Assignment - 1:

- Install Android Studio
- Install an Emulator (if it is supported on your laptop/desktop)
- Create Hello World Project (Empty Views Activity)
- Connect your android physical device
- Run your app on both Emulator and Physical Devices

Aug 21, 2024

Agenda

- Project Structure in android
- Views, ViewGroups and Layouts in android
- Create an App called Score Tracker and design the following screen



Project Structure in android (link)

What is an Activity in Android ?

An Activity is a screen that a user sees on his mobile device while your app is running

Conventionally, To design an activity, you need two files

XML holds the design (res->layout->xml*)

Kt files handles the logic (kotlin+java->packagename->.kt*)

Views, ViewGroups and Layouts in android

Layouts in views | Views | Android Developers

View: All UI components in android are called as views

Ex: Buttons, EditText, TextViews, RadioButtons, etc.,

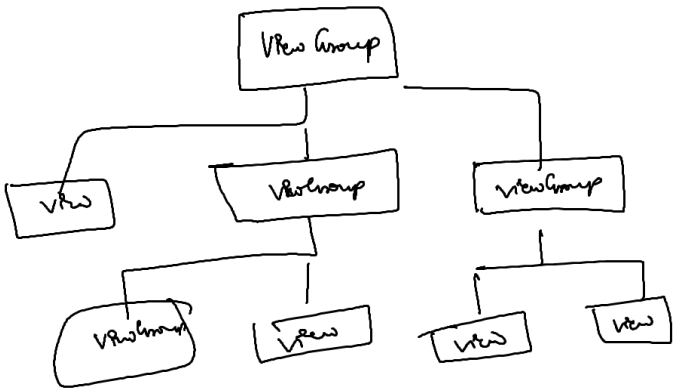
ViewGroups: can hold other views inside of them.
Ex: LinearLayout, ConstraintLayout, ListView, RecyclerView etc.,

Layouts: Layouts offer the basic structure for your activities (Screens). All Layouts are ViewGroups but not vice versa.

XML tags -> These are pre-defined tags that are defined in the android sdk.
Two types of Tags

- Self closing tags (are used for Views)
- Pair of Start tag and end tag (Are used for ViewGroups)

View Hierarchy



activity_main.xml code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="+"
        />
```

```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="0"
        android:textStyle="bold"
        android:textSize="100sp"
        android:textColor="#000"
        android:gravity="center"
        android:background="#FFD400"
        />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="-"
        />
</LinearLayout>
```

Aug 22, 2024

Agenda

- Scale the UI screen to occupy the fullest space of the activity on both portrait and landscape orientation.
- Bring the activity to action.

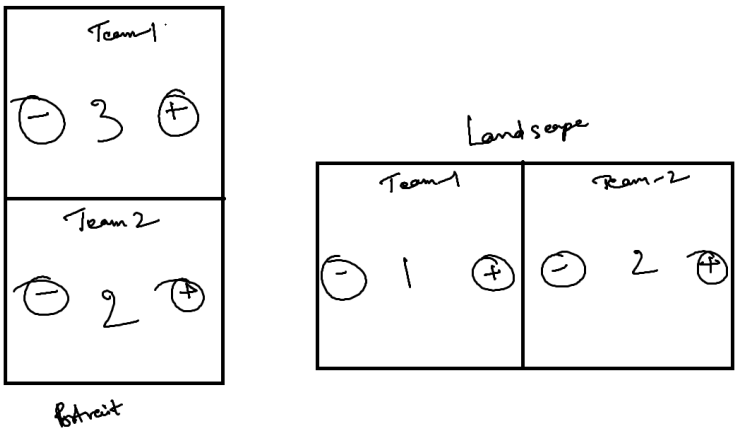
Dp vs SP

- **Display Pixels**
- **Scalable Pixels**
    - This is the unit for sizing your text that gets displayed.

[TextView in android]

Assignment - 2:

Let's assume that you are designing the same score tracker app of a football match where there will be two teams. Come up with a design (see below) and implement the functionality as well.



Share the code via drive to Pavan Kumar Reddy Tadi

Aug 23, 2024

Agenda

- Design for DarkTheme
- Save data while the configuration of the screen changes (Portrait to Landscape and vice versa)

Save UI states | Android Developers

Bundle | Android Developers

Score Tracker - Complete app code

MainActivity.kt

```
package com.nareshit.scoretracker

import android.os.Bundle
import android.view.View
import android.widget.TextView
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class MainActivity : AppCompatActivity() {
    var count:Int = 0
    lateinit var res:TextView
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main))
{ v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom)
            insets
        }

        res = findViewById(R.id.result)
        if(savedInstanceState!=null &&
savedInstanceState.containsKey("COUNT")){
            count = savedInstanceState.getInt("COUNT",0)
            res.text = count.toString()
        }
    }

    fun incrementScore(view: View) {
        // This function gets invoked as the + button is clicked
        count++
        res.text = count.toString()
    }

    fun decrementScore(view: View) {
        // This function gets invoked as the - button is clicked
        count--
        res.text = count.toString()
    }

    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putInt("COUNT",count)

    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_margin="5dp"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:textColor="@color/btn_txt_color"
        android:backgroundTint="@color/btn_background"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="+"
        android:textSize="20sp"
        android:onClick="incrementScore"
        />

    <TextView
        android:id="@+id/result"
        android:layout_weight="9"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="0"
        android:textStyle="bold"
        android:textSize="100sp"
        android:textColor="@color/tv_text_color"
        android:gravity="center"
        android:background="@color/tv_background"
        />

    <Button
        android:textColor="@color/btn_txt_color"
        android:backgroundTint="@color/btn_background"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="-"
```

```
        android:onClick="decrementScore"
        />
</LinearLayout>
```

res-> values -> colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="tv_background">#FFD400</color>
    <color name="tv_text_color">#000000</color>
    <color name="btn_background">#6750A3</color>
    <color name="btn_txt_color">#FFFFFF</color>
</resources>
```

res->values-> colors.xml (night)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="tv_background">#000000</color>
    <color name="tv_text_color">#FFFFFF</color>
    <color name="btn_background">#000000</color>
    <color name="btn_txt_color">#FFFFFF</color>
</resources>
```

Aug 26, 2024

Agenda

- Introduction to Kotlin Programming Language
- Discuss Basics
- Assignments

Introduction to Kotlin Programming Language

Why to learn Kotlin?

1. Google Announced kotlin as the official programming language for android development.

2. Over 50% of Professional Android Developers use kotlin as their primary programming language. Only 30% of the developers still use java. All the kotlin users (developers) said that kotlin makes them more productive.
3. Benefits
    a. Less Code combined with greater readability
    b. Fewer common errors
    c. Kotlin offers a great set of jetpack Libraries - extensive support is also there
    d. Mature language and the environment is also equally matured.
    e. Interoperability with the java
    f. Easy to Learn
    g. Big Community.

How to Run Kotlin Programs ?

1. Android Studio
2. Intellij IDEA
3. You can also integrate kotlin into visual studio code.
4. You can also run kotlin programs online

First Kotlin Program

```
fun main(){
   println("Android")
   print("Tech")
}
```

Output:

Android
Tech

1. All your kotlin files has **.kt** as the extension.
2. Your program execution begins from main()

Working with Variables in Kotlin

- There are two types of variables in kotlin
    - Immutable variables (Cannot be changed)
    - Mutable Variables (Can changed)

Immutable Variables

```
fun main(){
   // use val keyword to create an immutable
variable as follows
```

```
    val x = 10
    // Kotlin uses type inference
    // x=12 (val cannot be re-assigned)
    println(x)
}
```

Mutable Variables

```
fun main(){
    // use var keyword to create an mutable variable
as follows
    var x = 10
    // Kotlin uses type inference
    x=12
    println(x)
}
```

Check the data type of a variable to understand TypeInference Concept

```
package com.nareshit.kotlinfundamentalsforandroid

fun main(){
    // use var keyword to create an mutable variable
as follows
    val x = 10
    val y = 13.5
    val z = false
    // Kotlin uses type inference
    println(x::class.java.simpleName)
    println(y::class.java.simpleName)
    println(z::class.java.simpleName)
}
```

O/P:

int
double
boolean

Process finished with exit code 0

Data Types in Kotlin

Kotlin supports a set of Built in types that represent numbers. For Integer numbers there are four types with different sizes and hence the value ranges.

- Byte
- Short
- Int
- Long

While you initialize a variable with no explicit type specification, the compiler automatically infers the type with the smallest range enough to represent the value starting from Int.

If the value is not exceeding the range of Int,The type will be Int.

To append a Long value, we need to append the suffix L to the value.

```
package com.nareshit.kotlinfundamentalsforandroid

fun main(){
   // use var keyword to create an mutable variable
as follows
   val x = 10L
   val y = 13.5
   val z = false
   // Kotlin uses type inference
   println(x::class.java.simpleName)
   println(y::class.java.simpleName)
   println(z::class.java.simpleName)
}
```

Now x becomes Long because suffix L is added.

In Kotlin, unlike java, all data types are non primitive.

Every Data type will have the first letter capitalized

Floating point data types:

- Float (single Precision)
- Double (Double Precision)

To represent the float value, add F or f as a suffix to the actual value.

Explicit Type Conversion

- toByte()
- toShort()
- toInt()
- toLong()
- toFloat()
- toDouble()

```
fun main(){
    val a:Byte = 1
    val b = a.toInt()
    println(b::class.java.simpleName)
    println(b)
}
```

We can even declare the variables in the global section so that other functions in the program can access your variables

```
var a = 0
fun main()
{
    println(a)
    increment()
    println(a)
}

fun increment(){
    a++
}
```

Functions in Kotlin

- All functions in kotlin must be defined with a keyword called **fun**
- The return type of a function should be mentioned after the function declaration

```
package com.nareshit.kotlinfundamentalsforandroid

// The program execution always starts with main
fun main(){
   println(sum_display(10,20,30))
}

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

// You can write single line functions in kotlin
fun sum(a:Int, b:Int, c:Int):Int = sum(a,b)+c

// I want to get the output as this
// The sum of 10,20 and 30 is 60
fun sum_display(a:Int, b:Int, c:Int){
   println("The sum of $a, $b and $c is
${sum(a,b,c)}")
}
```

Using Existing Java Libraries in Kotlin (Interoperability with Java)

```
import java.util.Scanner

fun main(){
   println("Enter a value")
   val s:Scanner = Scanner(System.`in`)
   val a:Int = s.nextInt()
   println("Enter a value")
   val b:Int = s.nextInt()
```

```
    println("The sum of $a and $b is ${a+b}")
}
```

Output

Enter a value
13
Enter a value
15
The sum of 13 and 15 is 28

Process finished with exit code 0

Condition Control Structure

```
package com.nareshit.kotlinfundamentalsforandroid

import java.util.Scanner

fun main(){
    println("Enter a Value")
    val a = Scanner(System.`in`).nextInt()
    /*if(a%2 == 0){
        println("Even")
    }else{
        println("Not Even")
    }*/

    /*if(a%2==0) println("Even") else println("Not
Even")*/

    /*if statments can also be used as expressions
in kotlin*/
    val result = if(a%2==0) true else false
    println(result)
}
```

```
package com.nareshit.kotlinfundamentalsforandroid

import java.util.Scanner

fun main(){
   println("Enter a Value")
   val a = Scanner(System.`in`).nextInt()
   /*if(a%2 == 0){
       println("Even")
   }else{
       println("Not Even")
   }*/
   if(evenOrNot(a)) println("Even") else
println("Not Even")
}

fun evenOrNot(a:Int) = if(a%2==0) true else false
```

```
fun main() {
   println("Enter a Value")
   val a = Scanner(System.`in`).nextInt()
   /*if(a>50){
      println("The number is greater than 50")
   }else if(a>40){
       println("The number is greater than 40")
   }else{
       println("The number is not greater than 40")
   }*/

   if(a>50) println("Num greater than 50") else
if(a>40) println("Num greater than 40") else
println("Num not greater than 40")
```

```
}
```

Logical Operators in Kotlin

- &&
- ||
- !

Assignment 3:

A certain grade of steel has to be graded when hardness, tensile strength and carbon content are given. These values must meet the following criteria

Condition 1: Hardness must be greater than or equal to 50
Condition 2: Tensile Strength must be less than or equal to 5600
Condition 3: Carbon content must be less than 0.7

The grades must be printed as follows

- If all three conditions are met, the grade is 10
- If only 1 & 2 are met, the grade is 9
- If only 2 & 3 are met, the grade is 8
- If only 1 & 3 are met, the grade is 7
- If only one of the three conditions meet and others are not, the grade is 6
- If all conditions are not met, the grade is 5.

When Expression
when in kotlin is like a **switch** in java & c programming language. **When** is used when you have multiple branches and when the code looks complex with these multiple branches if used with in an if condition.

```
package com.nareshit.kotlinfundamentalsforandroid

import java.util.Scanner

fun main(){
   println("Choose your option: \n1.
Square\n2.Twice\n3.Half")
   val s:Scanner = Scanner(System.`in`)
```

```
    val option = s.nextInt()
    println("Enter a number")
    val a = s.nextInt()
    /*when(option){
        1 -> println(a*a)
        2 -> println(2*a)
        3 -> println(a/2)
        else -> println("Wrong Option")
    }*/

    val result =  when(option){
        1 -> a*a
        2 -> 2*a
        3 -> a/2
        else -> 0
    }

    if(result!=0) println(result) else
println("Wrong Option")
}
```