Naresh Technologies - Android App Development in Kotlin

Batch - 25

Demo Sessions recordings (Demo 1)

Day 1 : https://youtu.be/6Cy5g1llTkU
Day 2: https://youtu.be/JTE0ksI0A6k
Day-3 https://youtu.be/ujtEmXkxTOo
Day-4 https://youtu.be/qMjgo5jJ_tA
Day 5: https://youtu.be/dbceFGRUIQ0
Day 6: https://youtu.be/_CA5CAPxVs0
Day-7 https://youtu.be/WJRebCA19wI
Day-8 https://youtu.be/H2C8dDPyfEc

Demo Sessions Recordings (Demo 2)

Day-1 https://youtu.be/gEcICZ8LpKc
Day-2 https://youtu.be/A2OS9y5B3so
Day-3 https://youtu.be/XGlHiHYsr68
Day-4 https://youtu.be/i75Twar9FJc

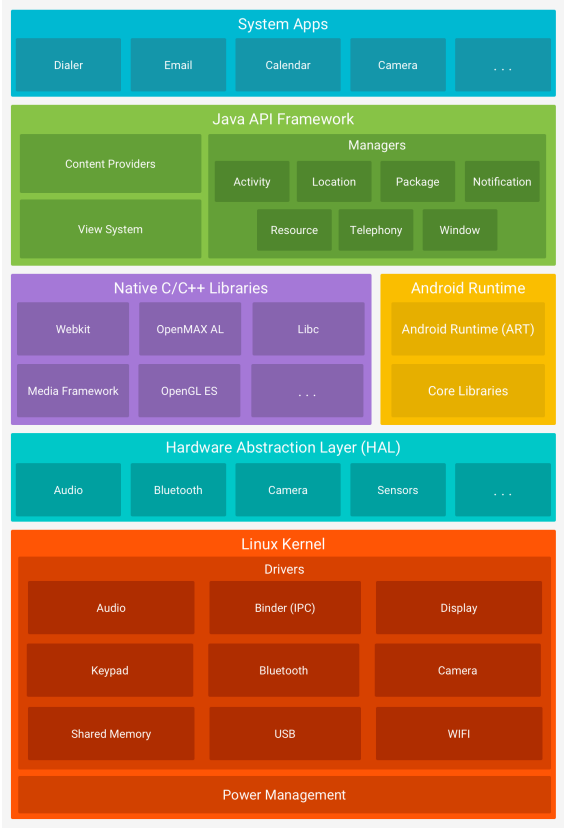
Syllabus: Click here

Nov 18, 2024

- Introduction to Android App Development, Course and the Plan Slides
- Installation of Android Studio (Download Android Studio, Install Android Studio)

Nov 19, 2024

- Please click here for more detailed documentation



Android's Architecture is designed to provide a robust framework for app development, ensuring efficiency, scalability, and ease of use. It is structured into layers, each serving a specific purpose.

1. **Linux Kernel (foundation Layer)**
    - **Purpose:** Acts as the core of Android. It handles hardware abstraction and provides fundamental services like:
        - Memory Management
        - Process Management
        - Networking
        - Security Features
        - Drivers (Display, camera, bluetooth, wifi, etc.)
    - **Why it Matters:** Ensures Efficient interaction between hardware and software.
2. **Hardware Abstraction Layer (HAL)**
    - **Purpose:** Standardizes the interaction between Android's system services and the hardware.
    - Developers don't need to directly handle hardware complexities; they use APIs that the HAL Exposes.
3. **Native Libraries & Android Runtime**
    - **Purpose:** Provides key functionalities through optimized c/c++ libraries.
    - **Examples**
        - SQLite: Database support

        - OpenGL ES: Graphics rendering
        - WebKit: Web browser engine
        - Libc: Standard c library
    - **Android Runtime (ART**)
        - Executes the apps and optimizes their performance through JIT (Just In Time compiler) and AOT(Ahead of Time) Compilation
4. **Java API Framework**
    - Contains higher-level apis for app development, written in kotlin & Java.
    - Example
        - ActivityManager: Manages the app lifecycle
        - NotificationManager: Handles push notifications
        - ContentProviders: Shares data between apps.
        - Resource Manager: Access non code resources like strings, colors, themes, etc,.
5. **Application Layer (System apps & User apps)**

- How to create an application on Android Studio ?
    - Click here

Nov 20, 2024

Android Project Structure Notes

1. **Manifest Folder**
    - Contains the `AndroidManifest.xml`.
    - Declares app components (activities, services, receivers) and permissions.
2. **java Folder**
    - Houses all source code.
    - Includes:
        - **Main Package**: Core app logic.
        - **Test Package**: Unit and instrumented tests.
3. **res (Resources) Folder**
    - Stores non-code resources like XML layouts, images, and strings.
        - **drawable/**: Images and graphics.
        - **layout/**: UI layouts in XML.
        - **mipmap/**: App launcher icons.
        - **values/**: Strings, colors, and dimensions.
4. **Gradle Scripts**
    - **build.gradle (Module)**: Configures dependencies, SDK versions, and plugins.
    - **build.gradle (Project)**: Global build settings and repositories.
    - **settings.gradle**: Lists included modules.
5. **.idea Folder**

    - Contains IDE-specific metadata.
6. **build Folder**
    - Generated files (e.g., compiled classes, APK).

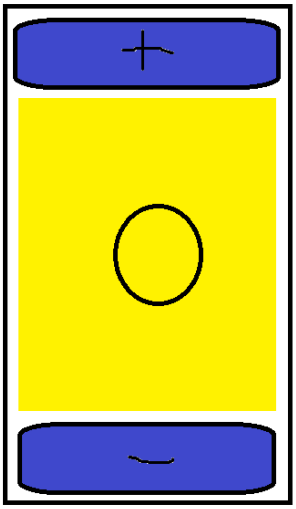
Keep code modular and organized for scalability. Use **Jetpack Compose** for modern UI development.

How to connect your physical device to android studio to run and test apps directly ?

Follow this link to connect your device

Assignment - 1:
Figure out how to use wifi to connect the devices (caution: your android version on the device must be 11 or 11+)

Score Tracker (First App)
Agenda:



Views, viewGroups and Layouts
PPT Link (Views, ViewGroup, Layouts)

Nov 21, 2024

https://drive.google.com/file/d/1t9DasyaionI-a3BxvYfs4R8lv4D43rkV/view?usp=sharing

Nov 22, 2024

Access the updated code (score tracker) here
https://drive.google.com/file/d/1t9DasyaionI-a3BxvYfs4R8Iv4D43rkV/view?usp=sharing

Nov 25, 2024
When the activity is rotated from portrait to landscape or vice versa, the activity restarts. That is the reason the count value is also reset to zero.

To address this challenge and persist the data across configuration changes of an activity, we have savedInstanceState.

Bundle is a data structure, usually helpful in transferring the data from one instance state to another. However, remember that we are supposed to save values in key & Value pairs.

Kotlin Programming Language

Why 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 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. Matured language and the environment is also matured.
    e. Interoperability with java.
    f. Easy to pick up
    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(){
    print("Hello World!")
```

```
}
```

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

print vs println

```
fun main(){
    println("Hello World!")
    print("Android!")
}
```

Working with variables in kotlin

1. There are two types of variables in kotlin
    a. Immutable variables (Cannot be changed)

```
fun main(){
   val x = 10 // we are creating an immutable variable and
assigned 10 to it.
   // x = 12 -> we get an error here saying that the val
cannot be reassigned
   println(x)
}
```

    b. Mutable variables (Can be changed)

```
fun main(){
   var x = 10 // we are creatiung an mutable variable and
assigned 10 to it.
   x = 12 //-> The value is reassigned.
   println(x)
}
```

Check the data type of the variables

```
fun main(){
   val x = 10
```

```
    val y = 13.5
    val z = false
    val a = 10L
    val b = 15.36f

    println(x::class.java.simpleName)
    println(y::class.java.simpleName)
    println(z::class.java.simpleName)
    println(a::class.java.simpleName)
    println(b::class.java.simpleName)
}
```

Output:

int
double
boolean
long
float

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 exceeds the range of Int, the type will be Long.

To append a long value we need to suffix the value with `L`.

Floating Point Data Types

- Float (Single Precision)
- Double (Double precision)

To represent the float value add `F` or a `f` as prefix.

Explicit Type Conversion

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

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

We can even declare the variables in the global section so that other functions can make use of the variables.,

```
var a = 0
fun main(){
   println("Before increment $a")
   increment()
   println("After increment $a")
}

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

Functions in Kotlin

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

```
package com.nareshtech.koltinprograms
```

```
fun main(){
   println(sum(10,20.5))
}

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

// you can write a single line function
fun sum(a:Int, b:Int, c:Int):Int = sum(a,b)+c

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

Kotlin's Interoperability with Java

```
package com.nareshtech.koltinprograms

// inorder to use the class, import it.

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 = ${a+b}")
}
```

Condition Control Structure

```
package com.nareshtech.koltinprograms

// inorder to use the class, import it.

import java.util.Scanner

fun main() {
   println("Enter a value")
   val s:Scanner = Scanner(System.in)
   val a:Int = s.nextInt()
   // We will have to check if the entered value is a
even number or not.
   /*if(a%2 == 0){
       println("Even Number")
   }else{
       println("Not an Even Number")
   }*/

   val result = if(a%2 == 0) true else false

   if(result) print("Even") else print("Not Even")
}
```

```
fun main() {
   println("Enter a value")
   val s:Scanner = Scanner(System.in)
   val a:Int = s.nextInt()
   // We will have to check if the entered value is a
even number or n
   if(evenOrNot(a)) println("Even") else println("not
Even")
}

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

Logical Operators in Kotlin

&& -AND

|| - OR
! - NOT

Assignment -2

Always assignments that I give you, make sure that you are completing the assignments and submitting them through an email. once done with assignment make sure you are dropping the email to pavankreddy.t@gmail.com

You are given a certain grade of steel and you are also given three of its properties values such as hardness, tensile strength and carbon content. You are supposed to grade the steel based on the conditions below

1. Hardness must be greater than 50
2. Tensile strength must be less than 5600
3. Carbon content must be greater or equal to 0.7

Take the values from the user and grade it based on the guidelines below

a. If all the condition are met, grade is 10
b. If only 1 & 2 are true, grade is 9
c. If only 2 & 3 are true, grade is 8
d. If only 1 & 3 are true, grade is 7
e. If only one condition is true, grade is 6
f. If no condition meets, the grade is 5

When Expressions

When 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 a if condition.

```
package com.nareshtech.koltinprograms

// inorder to use the class, import it.

import java.util.Scanner
import kotlin.math.*

fun main() {
   println("Enter your choice\n1.Double the
value\n2.Square\n3.Square Root")
   val s:Scanner = Scanner(System.in)
```

```
    val option = s.nextInt()
    println("Enter the number")
    val a = s.nextInt()

    when(option){
        1 -> {
            println(doubleValue(a))
        }
        2 -> println(powerOfTwo(a))
        3 -> println(squareRoot(a))
        else -> {
            println("Wrong Option! try rerunning the
program again")
        }
    }

}

fun doubleValue(a:Int) = 2*a

fun powerOfTwo(a:Int) = a*a

fun squareRoot(a:Int) = sqrt(a.toDouble())
```

Loops in Kotlin

When you want to repeat a set of statements for some number of times or till the condition fails, we can employ loops.

`in` is a keyword in kotlin that works with the range of values or a collection.

The `for` Loop in Kotlin

`..` -> This defines the range
`1..10` -> It means 1 to 10

```
fun main(){
    for(i in 1..10){
        print("$i ")
    }
}
```

```
for(i in 1..10 step 2){
    print("$i ")
}
```

```
for(i in 10 downTo 1){
    print("$i ")
}
```

```
for(i in 10 downTo 1 step 3){
    print("$i ")
}
```

```
val strings = listOf("Pavan", "Kumar","Reddy", "Tadi")
for(i in strings){
    print("$i ")
}
```

While & do-while loops

Works similar to the ones we have in c, java & Python

```
package com.nareshtech.myfirstapp

fun main(){
    var i = 0
    while(i<=10){
        print("$i ")
        i++
    }

    println()
```

```
    do{
        print("$i ")
        i--
    }while(i>=1)
}
```

Assignment-2:

- Write a program to find if a given number is prime or not.
- Extend the first program to identify all the palindromic prime numbers till 10,000.
- Write a program to find if a given number is present in the fibonacci series.
    - 0,1,1,2,3,5,8,13,21,34,55,89…
- Write a program to find the factorial of any given number.
- Write a program to identify the average of all the values given in an array.

```
package com.nareshtech.myfirstapp

fun main(){
    for(i in 11..10000){
        if(isPrimeOrNot(i)){
            // You will identify if the number is a
palindrome of not
            if(palindromeOrNot(i)){
                print("$i ")
            }
        }
    }
}

fun palindromeOrNot(n:Int):Boolean{
    /*// Reverse the number
    var num = n
    var reverse = 0
    while(num!=0){
        val lastDigit = num%10
        num = num/10
        reverse = reverse*10 + lastDigit
    }
```

```
    return reverse == n*/

    val os = n.toString()
    val rs = os.reversed()
    return os==rs
}

fun isPrimeOrNot(n:Int):Boolean{
    for(i in 2..n-1){
        if(n%i == 0){
            return false
        }
    }

    /*var i = 2
    while (i<n){
        if(n%i==0){
            return false
        }
        i++
    }*/
    return true
}
```

`repeat` in built function

```
repeat(10){
    print("NIIT ")
}
```

Arrays in Kotlin

Array is one of the fundamental data structures in practically all programming languages. The idea behind an array is to store multiple pieces of the same data type, such as integers or strings under a single name.

Arrays are used to organize data in programming so that a related set of values can be easily stored and searched for.

Important Points to make a note of

1. They are stored in contiguous memory locations
2. They can be accessed programmatically through their indices.
3. They are mutable.
4. Their size is fixed
5. The index of an array starts at 0 and ends at 1 less than the actual size of the array.

Create an Array using arrayOf() and arrayOf<Datatype>() functions

```
package com.nareshtech.myfirstapp

fun main(){
   var a = arrayOf(1,2,3,4,5,6,7,8,9,10)
   // get the size of an array
   println("the size of the array is ${a.size}")

   // Modify the values based on the index
   a[5] = 34

   // set method to change a value
   a.set(5, 43)

   // access the idividual elements through indices
   println(a[5]) // this prints the value present on the
sixth index of a


   // traverse the array and print all the values
   for(i in a){
       print("$i ")
   }

   println()
   println(a.get(5))

}
```

Using Array() constructor & Lambdas

```
import java.util.Scanner

fun main()
{
   println("Enter the size of the array")
   val s:Scanner = Scanner(System.`in`)
   val l = s.nextInt()

   var a = Array<Int>(l,{i -> i*2})

   for(i in a){
       print("$i ")
   }
}
```