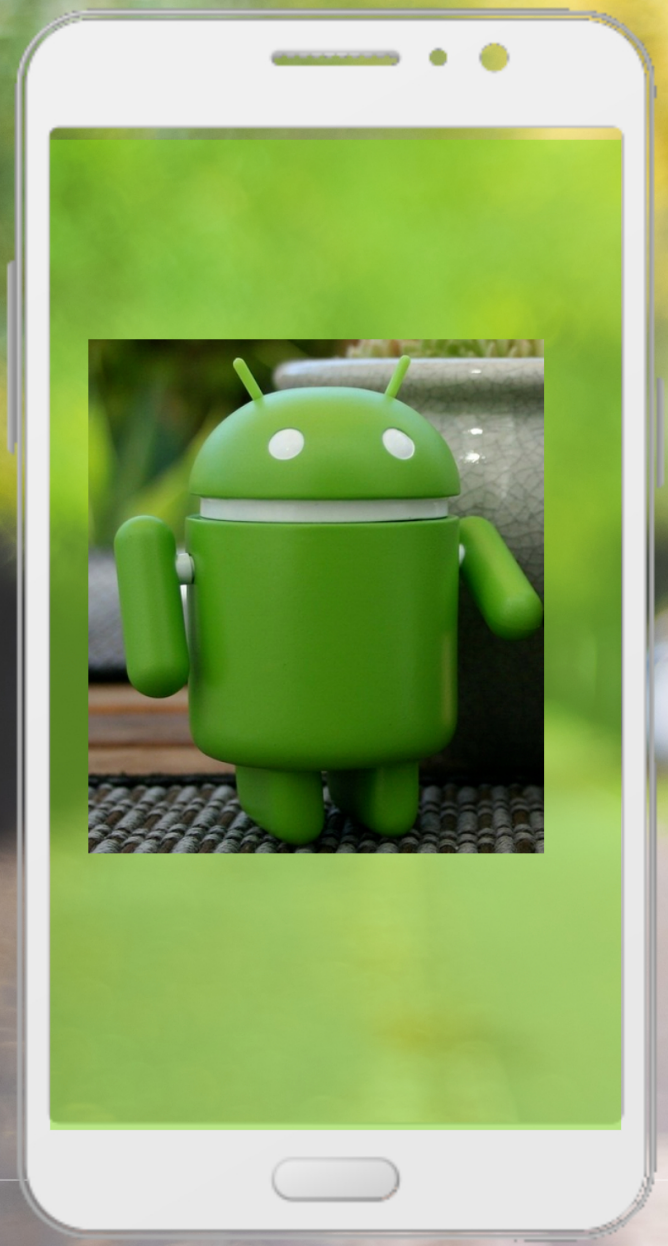
Learn Simply: Android Mobile Application Development using Kotlin



- Idea
- Design
- Code

DR. IYAD ABU DOUSH

Learn Simply: Android Mobile Application Development using Kotlin

Dr. Iyad Abu Doush

This book is for sale at http://leanpub.com/mobilekotlin

This version was published on 2020-05-09

Contents

Preface

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Kotlin Programming

Kotlin is a cross-platform general-purpose programming language that is announced by Google in 2019 to be the preferred language for Android application development. The language has been included in Android studio since 2017 as an alternative to the standard Java compiler.

In this part of the book we will learn how to use Kotlin as a general purpose programming language. After that, we will learn how to use Kotlin with Android to develop mobile applications.

Several online resources are available to help you in learning how to program in Kotlin:

a) Kotlin official website
https://kotlinlang.org/

b) Kotlin Online IDE
https://play.kotlinlang.org/

If you want to start programing Kotlin you can use IntelliJ IDEA IDE available from Jetbrains[1].

The following are the Kotlin basic syntax and rules:

- No semicolon ; at the end of each line of code.
- No need to define type of variables, you can just assign values immediately.
- You can use Java code and Kotlin code in the same file.
- No *new* keyword when defining objects.
- No primitive types as all the types are classes.
- No static members can be defined in the class.

Hello World

Each Kotlin program must have a **main** function which is the starting point for execution. The following is the simplest coding example in Kotlin. This function will print *Hello World* in the screen.

```
1  //main fun
2  fun main(){
3      print("Hello World\n")
4  }
```

Comments

In order to add comments (i.e., a text that will not be executed in the code) to the Kotlin you can use a single line comment // or the multi-line comment

[1] https://www.jetbrains.com/idea

```
// This is a single line comment
/*
This is
a multi-line comment
*/
```

Variables and Constants

In Kotlin you can define variables using **var**, but there is no need to define the type of the variable as the language support type inference (loosely typed).

```
var name="Welcome"
print(name + "\n")
```

You can define a constant (i.e., the value cannot be changed) in Kotlin using **val**.

```
val name   = "Iyad" // String
val age  =25        //integer
val GPA  = 4.8      // floating point
```

Convert dataType

In some cases you may need to convert from a string to integer to apply mathematical operations. Kotlin allow you to convert between different types using several built-in methods like **toDouble()** or **toInt()**.

```
var strSalary:String= "500.5"
var doubleSalary:Double = strSalary.toDouble()
println(doubleSalary)

var strAge:String="15"
var intAge:Int = strAge.toInt()
println(intAge)

println(doubleSalary+intAge)
```

```
500.5
15
515.5
```

Math operations

The mathematical operators found in other programming language are available for you in Kotlin +, -, *, and /.

```
var n2:Double= 3.5
var n1:Double = 5.5
var sum=n1+n2
println(sum)
println("Sum = "+sum)
println("Sum = $sum")
```

9.0
Sum = 9.0
Sum = 9.0

Logical operators

The logical operators which are used to connect between two or more logical terms are available in Kotlin >, <, >=, <=, ==, !=, And &&, OR ||, and NOT !.

```
fun main(){
    println(3<2)
    println(2==2)
    println(3>1 || 1==0)
    println(! (3>1) )
    println(3>1 && 1==3)
}
```

false
true
true
false
false

If statement

The if statement is used as a control structure which allows you to skip or execute a block of code based on a logical condition. If the condition is true then the statement is executed, if it is false then we skip the statement. Kotlin has three different structures for the if statement.

Simple if

The simple if has a statement or block of statements that will be executed if the condition is true. The following example compare the grade with different values and print the equivalent letter grade.

```
fun main(){


    val grade = 75

    if (grade>= 90){
        println("A")
    }

    if(grade in 80..89){
        println("B")
    }

    if(grade in 70..79){
        println("C")
    }

    if (grade<70){
        println("Fail")
    }

}
```

C

If ... else

Kotlin provide if..else structure, the if part is executed if the condition is true and the else part will be executed if the condition is false.

The following example check if the age is larger than or equal 18 then it will print “You could apply for the job” and it will print “You can not apply for the job” if the age is less than 18.

```
fun main(){

    val age = 25

    if(age >=18){
        println("You could apply for the job")
    }
    else{
        println("You can not apply for the job")
    }
}
```

You could apply for the job

Nested If . . else

The if statement can have more than one condition in the same block of code. In this case we call the structure nested if. Note that only one of the statements will be executed which is the one that matches the condition.

```
fun main(){


    val grade = 65

    if(grade>= 90){
        println("A")
    }
    else if(grade in 80..89){
        println("B")
    }
    else if(grade in 70..79){
        println("C")
    }
    else{
        println("Fail")
    }
}
```

Fail

When (Similar to switch statement)

The when statement in Kotlin allows to execute block of code based on a specific value that equal one of different values. It is similar somehow to the nested if as it expects only one block to be executed which is the one that matches the value.

```
fun  main(){

val foodID= 10

when(foodID){
        1 -> {
            print("You got Sandwich")
            print("You got Salat")
        }

        10 ->{
            println(" You got burger")
        }

        else ->{
            println("You did not order anything")
        }
    }
}
```

You got burger

For loop

In order to execute a block of code several times we use the for loop, or other kinds of loops. The loop has three parts initialization, increment or decrement and a stop condition. The default increment in Kotlin is 1.

The following code will print the numbers from 1 to 5.

```
fun main(){
    for (i in 1..5)
        println(i)
}
```

1
2
3
4
5

Another example with increment by 2 and decrement by 2.

```
fun main(){

    println("========== Increment========== ")
    for (i in 0..10 step 2){
        println("Number is $i")
    }

    println("========== Decrement========== ")
    for (i in  10 downTo 0 step 2){
        println("Number is $i")
    }
}
```

}
========== Increment==========
Number is 0
Number is 2
Number is 4
Number is 6
Number is 8
Number is 10
========== Decrement==========
Number is 10
Number is 8
Number is 6
Number is 4
Number is 2
Number is 0

While and Do..While loop

Another types of loops are the while loop which check the condition before executing the loop and the do..while which checks the condition at the end of the loop.

```
fun main(){

fun main(){

    println("While Loop")

    var i=4
    while (i<=10){
        println("Counter $i")
        i += 2
    }

    println("Do While Loop")
    i= 4
    do{
        println("Counter $i")
        i += 2
    }while (i<=10)
}
```

While Loop
Counter 4
Counter 6
Counter 8
Counter 10
Do While Loop
Counter 4
Counter 6
Counter 8
Counter 10

String type

A String data type is used to store a sequence of characters. We use the double quotation to enclose that sequence. We can access the character in the String by using the angle brackets and the index of the character (e.g., allMessage[0]). The String is dealt with as an object which means that you can access the built-in methods inside the String by using the name of the object followed by the dot operator (e.g., allMessage.toLowerCase()). In the following example we used several built-in methods from the String class which are:

- *toLowerCase()* : convert the alphabet in the string into lower case letters.
- *toUpperCase()* : convert the alphabet in the string into upper case letters.

- *trim()*: remove the all the space before and after the string (if any).
- *split(str)*: split the string into chunk of sub-strings based on the given separator and return the result as a list.
- *contains(str)*: return true if the string has the given sub-string.

Note that in the code we use $ followed by the string variable name to refer to the variable value. For example, "$name,$message" means that we show the values of two Strings: name and message.

```
fun main(){

    val message= " Welcome to Planet"
    val name = "Iyad"
    val allMessage = "$name,$message"
    println(allMessage[0])             // first character
    println(allMessage.toLowerCase()) // convert to lowercase
    println(allMessage.toUpperCase()) // convert to uppercase
    println(message.trim())          // remove space
    println(message)
    val listOfTokens = message.trim().split(" ")  // split into words

    for (token in listOfTokens){    // check all the words
        if (!token.contains("to") && !token.contains("is")){
                println("token: $token")
            }// end if
    }// end for
}
```

I
I
iyad, welcome to planet
IYAD, WELCOME TO PLANET
Welcome to Planet
Welcome to Planet
token: Welcome
token: Planet

Null safety

In Kotlin you cannot give a null value to a variable unless you provide ? after the type.

```
var str:String
str= null // error


var str:String? // Tell program to exit if the value is null
str= null
println(str)
str="Iyad"
print(str)
```

The **!!** operator can be used to tell the the compiler that the variable is not null and if it is null, throw a null pointer exception.

```
var str:String?
str= null
println(str!!)
str="Iyad"
print(str)
```

The previous code will generate the following exception:
Exception in thread “main” kotlin.KotlinNullPointerException

The following is another example

```
var str:String?
str= "Hello"
println(str!!)
str="Iyad"
print(str)
```

Hello
Iyad

Arrays

Arrays are sequence of values used to save several values using one name. We define the number of elements in the array when we define the array. Each value in the array can be accessed by using the name of the array and the location of the value we want to access (e.g., listOfCars[1]).

```
fun main(){

    print("Enter number of Cars: ")
    val maxSize = readLine()!!.toInt()     // read the array size
    //Write into Array
    var listOfCars:Array<String> = Array(maxSize){""} // define the array type and s\
ize
    for(i in 0 until  maxSize){
        print("Enter Car model $i:")
        listOfCars[i] = readLine()!!.toString() // reading and saving the value into\
 the array
    }

    println("Your cars Are ")
    for(i in 0 until maxSize){
        println("Car $i:  ${listOfCars[i]}")
    }
}
```

Enter number of Cars: 3
Enter Car model 0:Toyota
Enter Car model 1:BMW
Enter Car model 2:Nissan
Your cars Are
Car 0: Toyota
Car 1: BMW
Car 2: Nissan

Array List (Linked list)

Arrays are fixed and cannot have more elements than its given size. Array list is a dynamic array that can have more elements during the execution of your program.

The following code define arraylist of Strings. It takes the pets names until the user enters quit.

```
import java.util.*

fun main(){
    //Write App in Array
    var listOfPets= LinkedList<String>()

    do{
        print("Enter Pet name  or exit to quit:")
        val petName =readLine()!!.toString()
        if(petName != "quit"){
            listOfPets.add(petName)
        }

    }while (petName!="quit")

    println("Your pets Are - using Index")
    for(i in 0 until listOfPets.size){
        println("Pet $i:  ${listOfPets[i]}")
    }

    println("Your pets Are - using Object")
    for(pet in  listOfPets ){
        println("Pet:  $pet")
    }

}
```

Enter Pet name or quit to exit:Soso
Enter Pet name or quit to exit:Mimi
Enter Pet name or quit to exit:quit
Your pets Are - using Index
Pet 0: Soso
Pet 1: Mimi
Your pets Are - using Object
Pet: Soso
Pet: Mimi

HashMap

You can have pair of key and value using a data structure called hashmap.

```
import java.util.*

fun main(){
    //defining hashmap of interger key and string value
    var listOfUsers = HashMap<Int,String>()
    listOfUsers[50]= "Iyad"
    listOfUsers[120]= "Sarah"
    listOfUsers[12]= "Ahmed"
    listOfUsers[5]= "Bayan"

    listOfUsers.put(50,"Jamal" )

    for (key in listOfUsers.keys)
        println("$key: ${listOfUsers[key]}")
    }
```

50: Jamal
5: Bayan
120: Sarah
12: Ahmed

Function

A function or method is a block of code that is defnined to be called by other codes. It can take one or more input called parameter and it can return an output.

The following Kotlin code define a function to add two numbers. It takes two numbers and sum them, then returns the result. In the second function, which is called *displayInfo*, the keyword vararg tells the compiler that this function can have a variable number of arguments to be passed to the function.

```
fun addNumbers(x:Double=0.0,y:Double=0.0):Double{
    return x+y
}

fun displayInfo( vararg  names:String){
    for(name in names){
        println(name)
    }
}


fun main(){
```

```
    var returnAdd= addNumbers( 3.0, 4.0)
    println("returnAdd: $returnAdd")

    returnAdd =  addNumbers(y=10.0)
    println("returnAdd: $returnAdd")

    returnAdd =  addNumbers(10.0)
    println("returnAdd: $returnAdd")


    displayInfo( names = *arrayOf("Sarah","Iyad"))
}
```

returnAdd: 7.0
returnAdd: 10.0
returnAdd: 10.0
Sarah
Iyad

Classes

A class is a blueprint for the code which have data members and methods (functions). We define a class to create an object from that class. Kotlin have similar access modifiers to Java: private, public and protected.
In the following code we define a class called classLamp which has one data member *isOn* and three methods *turnOn, turnOff, displayLightStatus*. Then we use the class in main by defining two objects light1 and light2.

```
classLamp {
    // property (data member)
    private var isOn: Boolean = false
    // member function
    fun turnOn() {
        isOn = true
    }
    // member function
    fun turnOff() {
        isOn = false
    }
    fun displayLightStatus(lamp: String) {
        if (isOn == true)
            println("$lamp lamp is on.")
```

```
        else
            println("$lamp lamp is off.")
    }
}
fun main(args: Array<String>) {
    val light1 = Lamp() // create light1 object of Lamp class
    val light2 = Lamp() // create light2 object of Lamp class
    light1.turnOn()
    light2.turnOff()
    light1.displayLightStatus("l1")
    light2.displayLightStatus("l2")
}
```

l1 lamp is on.
l2 lamp is off.

A constructor is a method that is automatically called when we create an object from the class. In Kotlin, the primary constructor is part of the class header. Here's an example, note that the block of code surrounded by parentheses is the primary constructor: (val firstName: String, var age: Int).

```
class Person(val firstName: String, var age: Int) {
    // class body
}
```

The following is another example:

```
fun main(args: Array<String>) {
    val person1 = Person("Joe", 25)
    println("First Name = ${person1.firstName}")
    println("Age = ${person1.age}")
}
class Person(val firstName: String, var age: Int) {
}
```

When you run the program, the output will be:

First Name = Joe
Age = 25

The primary constructor has a constrained syntax, and cannot contain any code.
You can have initialization code that can do more than initializing properties (data members) using an initializer block. It is prefixed with init keyword.

```
fun main(args: Array<String>) {
    val person1 = Person("joe", 25)
}
class Person(fName: String, personAge: Int) {
    val firstName: String
    var age: Int
    // initializer block
    init {
        firstName = fName.capitalize()
        age = personAge
        println("First Name = $firstName")
        println("Age = $age")
    }
}
```

When you run the program, the output will be:

FirstName = Joe
Age = 25

Another example to create class Car with a constructor and create an instance in main.

```
class Car(type:String,model:Int,price:Double){
init{
    println("Type:$type")
    println("model:$model")
    println("price:$price")
    }
}

fun main(){
    var car1=Car("BMW",2015,12000.500)
}
```

Type:BMW
model:2015
price:12000.5

Another example which defines the data members outside the class constrcutor:

```
class Car(){
    var type:String?=null
    var model:Int?=null
    var price:Double?=null

    constructor(typeV:String,modelV:Int,priceV:Double) : this() {
        this.type = typeV
        this.model=modelV
        this.price=priceV
        println("Type:$type")
        println("model:$model")
        println("price:$price")
    }
    fun GetPrice(): Double? {
        return this.price!!
    }
}

fun main(){
    var car1=Car("BMW",2015,12000.500)
    car1.GetPrice()
}
```

Class Inheritance

Inheritance is one of the common relationships that we can have between classes. In this relation we have a parent (superclass) and a child (sub-class). The child will inherit all public and protected data members and functions from the parent.

In Kotlin, by default all the classes are final (non-inheritable). To allow a class to be inherited by others, you must mark it with the open modifier. Note that the child class has the responsibility to initialize the parent class. If the child class has a primary constructor, then it must initialize the parent class in the class header with the parameters passed to its primary constructor.

In the following code the parent class is BankAccount and the sub-class is SavingsAccount. The *open* keyword at the beginning of the parent class means that we can have children of this class. In the child class we use colon followed by the parent class name to indicate the the class we inherit the public and protected data members and functions from.

```
open class BankAccount(val accountNumber: String, val accountName: String) {
var balance : Double = 0.0

    fun depositeMoney(amount: Double): Boolean {
        if(amount >0) {
            balance += amount
            return true
        }
        else {
            return false
        }
    }

    fun withdrawMoney(amount: Double): Boolean {
        if(amount >balance) {
            return false
        }
        else {
            balance -= amount
            return true
        }
    }
}// end BankAccount class

class SavingsAccount (accountNumber: String, accountName: String, val interestRate: \
Double) :
    BankAccount(accountNumber, accountName) {

    fun depositInterest() {
        val interest = balance * interestRate / 100
        this.depositeMoney(interest);
    }
}// end SavingsAccount class

fun main(args: Array<String>) {
    // Create a Savings Account with 6% interest rate
    val savingsAccount = SavingsAccount("64524627", "Rajeev Kumar Singh", 6.0)

    savingsAccount.depositeMoney(1000.0)

    savingsAccount.depositInterest()

    println("Current Balance = ${savingsAccount.balance}")
```

```
44 }
```

```
Current Balance = 1060.0
```

Android and Kotlin

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

First Mobile Application

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Layout Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

ConstraintLayout

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

LinearLayout

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

RelativeLayout

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

GridLayout

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Views (Widgets)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Pick SpongeBob Character

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Data-Driven Containers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Vocabulary game (using ArrayList)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Vocabulary game (using File)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Intents

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Implicit Intent Example - Opening a URL

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Explicit Intent Example - Restaurant App using ListView

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design and Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Activity Life Cycle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Understanding the Lifecycle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

An App. that Shows a Message on each Stage of the Activity Lifecycle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Purpose of the Lifecycle Phases

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

How to save the application state?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Stopwatch application

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

RESTful Web API

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

REST and HTTP

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

JavaScript Object Notation (JSON)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

RESTful Web API and Web services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

ToDo list application using Web API

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Cats images application using Web API

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Action Bar

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Simple Action bar app.

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

SQLite Database

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

What is a database?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Where is the Data?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

How to talk to the database?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

My Notes Application

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Creating the Database

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Update the created database

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Adding a note when clicking the button

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Cursors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

View on-device files with Device File Explorer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Cloud Database - Firebase

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

RESTful Web Services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Mobile Backend as a Service

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Firebase Realtime Database

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

How Firebase store data?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Firebase for Android

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

A simple chat application using Firebase

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Localization

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Internationalized using strings.xml

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Game switchboard with two languages

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Location

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Pick city application

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Idea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

Coding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.

About the Author

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/mobilekotlin.