Android Applications Development using Kotlin

Ahmed Abdallah

Course Contents

- Kotlin what, when and why ?
- Kotlin syntax
- Android Environment Installation
- First Android App
- Android App Anatomy
- MVC Design Pattern
- Accessing Device Storage
- Networking and APIs
- Project Structure
- SOLID Principles

FAQs

- Should I have development background ?
- How sessions work ?
- What is next ?
- What is the objective ?

About Me

- Name: Ahmed Abdallah (a.k.a. Yossef)
- Title: Software Security Consultant
- Experience: 12 years
- Languages: Java, Kotlin, Swift, Python, JS, Typescript, Perl, PHP,...
- Platforms: Desktop, Web, iOS, Android, Cloud, IoT
- Security: Penetration Testing and Ethical Hacking

Kotlin Overview

Kotiln

- Kotlin is a programming language introduced by JetBrains, the official designer of the most intelligent Java IDE, named Intellij IDEA. This is a strongly statically typed language that runs on JVM.
- In 2017, Google announced Kotlin is an official language for android development. Kotlin is an open source programming language that combines object-oriented programming and functional features into a unique platform.

Why Kotlin

- Promoted by Google
- Easy Language
- Concise, you can do more with less code
- Smaller runtime and better performance
- Modern Language, easy to learn and to use

Kotlin Usage

- Kotlin is a high level programming language
- Kotlin code is compiled to either:
  - Byte code for JVM runtime environment
  - ES5 (JavaScript Compatible code)

Kotlin Usage

- Kotlin is famous for developing Android Applications, however much more can be done with Kotlin

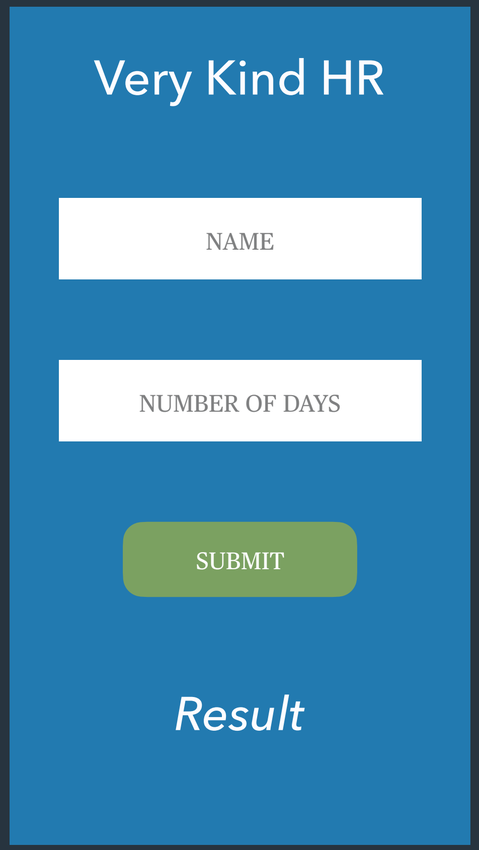
Installation

1. Java 8 installation: as kotlin uses JVM we need to install Java first
2. IDE Installation: You can choose between eclipse, netbeans, IntelliJ (Android Studio) or even your favorite editor along with kotlin command line compiler
3. Configure your tools
4. Write your first application

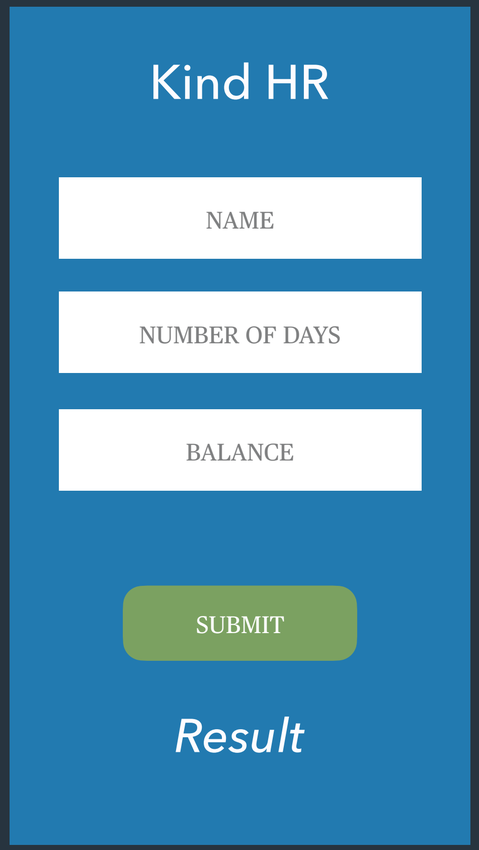
First Kotlin App

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

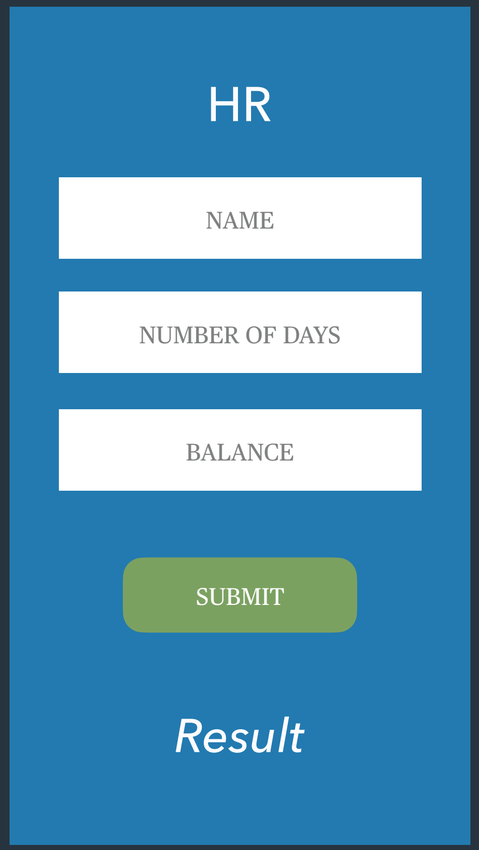
Demo



Demo



Demo



Basics of Kotlin

Variables

> To define a variable in Kotlin we use the keyword “Var”

```
var variableName: DataType
```

Constant

> To define a variable in Kotlin we use the keyword “val”

```
val constantName: DataType
```

Data Types - Numbers

Type	Size
Double	64
Float	32
Long	64
Int	32
Short	16
Byte	8

Data Types - Char

> Char: A datatype that represents a single character

```
val letter: Char     // defining a variable
letter = 'A'         // Assigning a value to it
                println("$letter")
```

Data Types - String

> String: A datatype that represent a list of characters

```
val name: String     // defining a variable
name = ”Ahmed“       // Assigning a value to it
println(“$name")
```

Data Types - Boolean

> Boolean: A datatype that can hold only one of two values (true/false)

```
val flag: Boolean     // defining a variable
flag = false          // Assigning a value to it
println("$flag")
```

Data Types - Array

> Arrays are a collection of homogeneous data

```
val numbers: IntArray = intArrayOf(1, 2, 3, 4, 5)
println("List starts with ${numbers[0]}")
```

Data Types - Collections

> Kotlin has two types of collection:

> **Immutable collection** it is a fixed list, map or set that cannot be changed (constant values)

> **Mutable collection** it is changeable (contains variables)

Data Types - Collections

```
val numbers: MutableList<Int> = mutableListOf(1, 2, 3) //mutable List

val numbers: List<Int> = listOf(1, 2, 3) //immutable List
```

Loops - For

```
val items = listOf(1, 2, 3, 4)
for (i in items) println("values of the array"+i)
```

Loops - For

```
val items = listOf(1, 22, 83, 4)
for ((index, value) in items.withIndex()) {
 println("the element at $index is $value")
}
```

Loops - While

```
var x:Int = 0
while(x< = 10) {
    println(x)
x++}
```

Loops - Control

- **Continue:** skips the rest of the current iteration and go to the next iteration
- **Break:** stops the entire loop and exit to execute what is after the loop

Kotlin Functions - Lambda Function

```
fun sum(numbers:List<Int>):Int{
var sum = 0
numbers.forEach {num -> sum+=num}
return sum
}
```

Conditions

```
If (condition) {
/ / do something }
else {
/ / do another thing
}
```

When

```
when (expression) {
Val1 -> // do something
Val2 -> // do another thing
}
```

Kotlin Functions

```
fun sayHello(){
    println("Hello")
}
fun main(args : Array<String>){
   sayHello()
}
```

Kotlin Functions

```
fun sayHelloName(name: String){
    println(“Hello, ${name}“)
}
```

Kotlin Functions

```
fun sayHelloName(name: String):String{
    return “Hello, ${name}“
}
```

Kotlin Functions

```
fun sum(num1:Int, num2:Int):Int{
    return num1 + num2
}
```

Kotlin Functions - vararg

```
fun sum(vararg numbers:Int):Int{
var sum = 0
for (num in numbers) sum += num
return sum
}
```

Kotlin Functions - Default Values

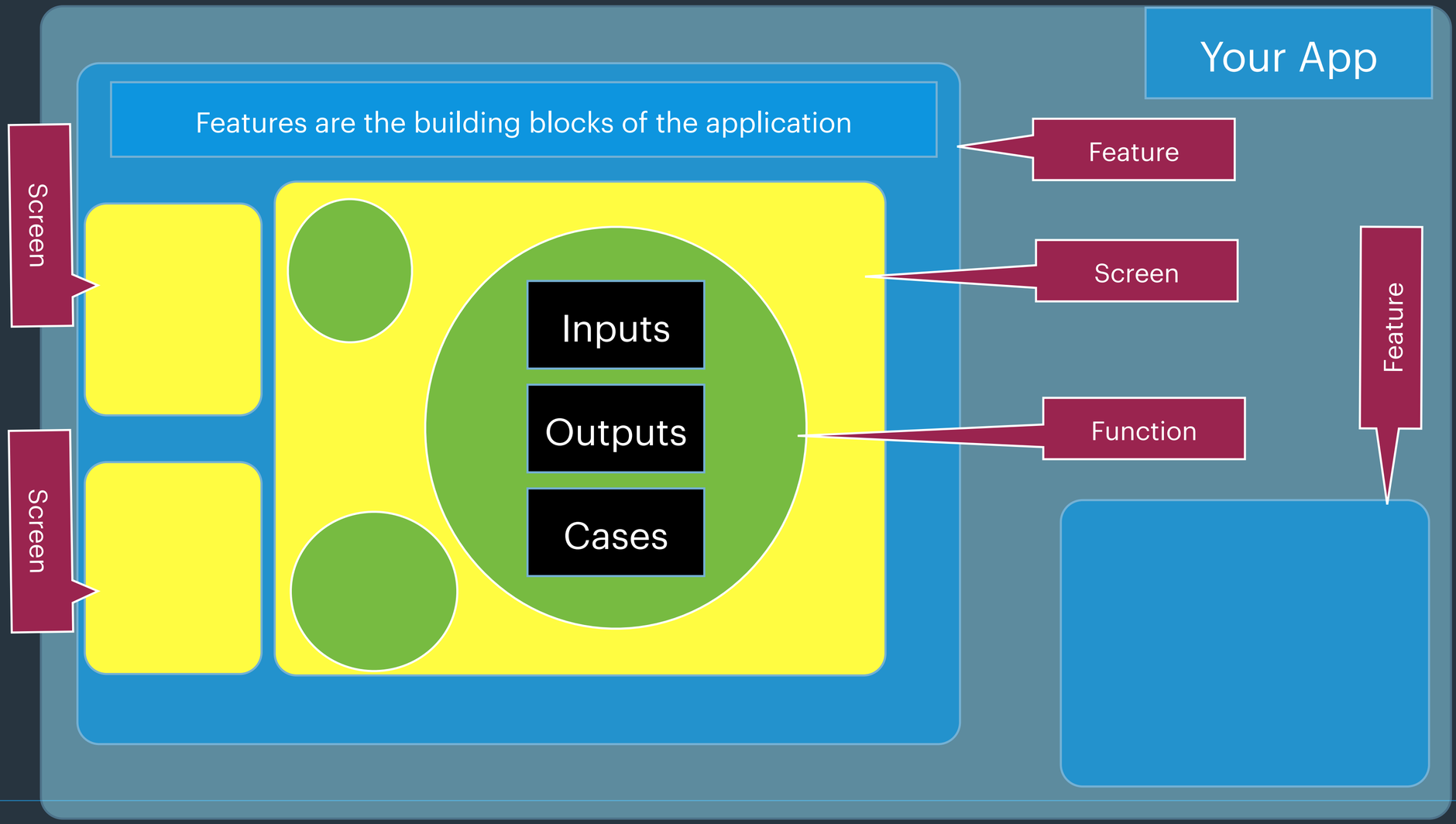
```
fun main(args: Array<String>) {
test()
test(50,”NO”)
}

fun test(num:Int= 10, str: String =“OK”){
print("Number is: $num and String is: $str")
}
```

Kotlin Functions - Inline Function

```
fun main(args: Array<String>){
/ /lambda function
val sum = {num1: Int, num2: Int -> num1 + num2}
println("10+5: ${sum(10,5)}")
}
```

Android App Anatomy

App Componentes



App Componentes

SCREEN (ACTIVITY)

Layout

Activity Class

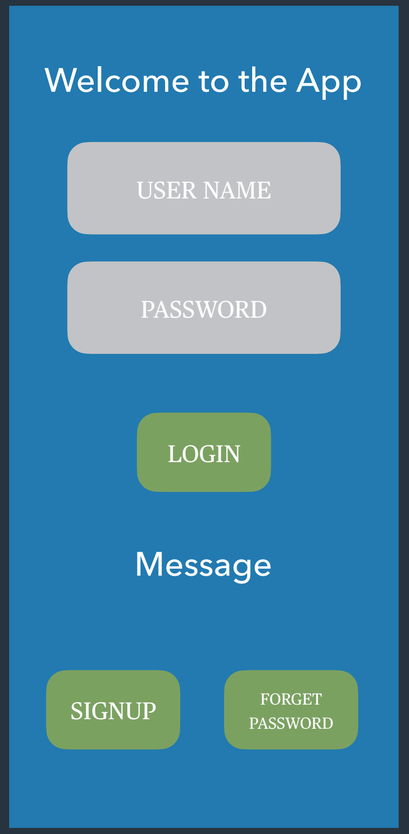
I/O Components

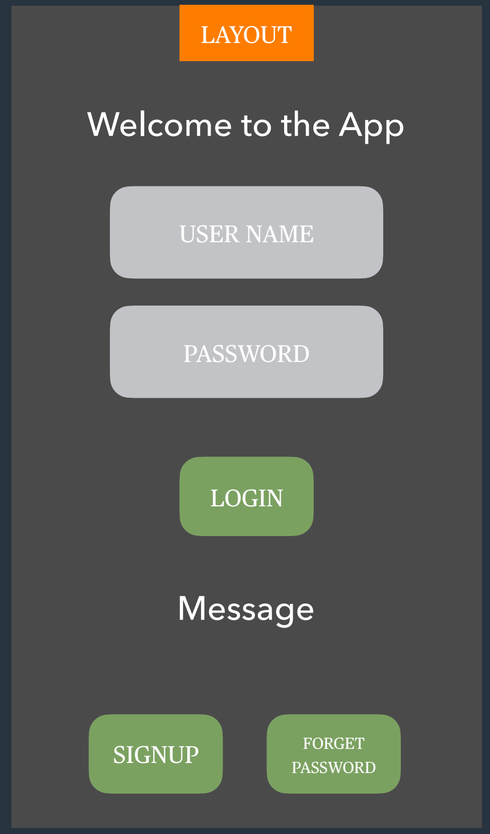
Attributes

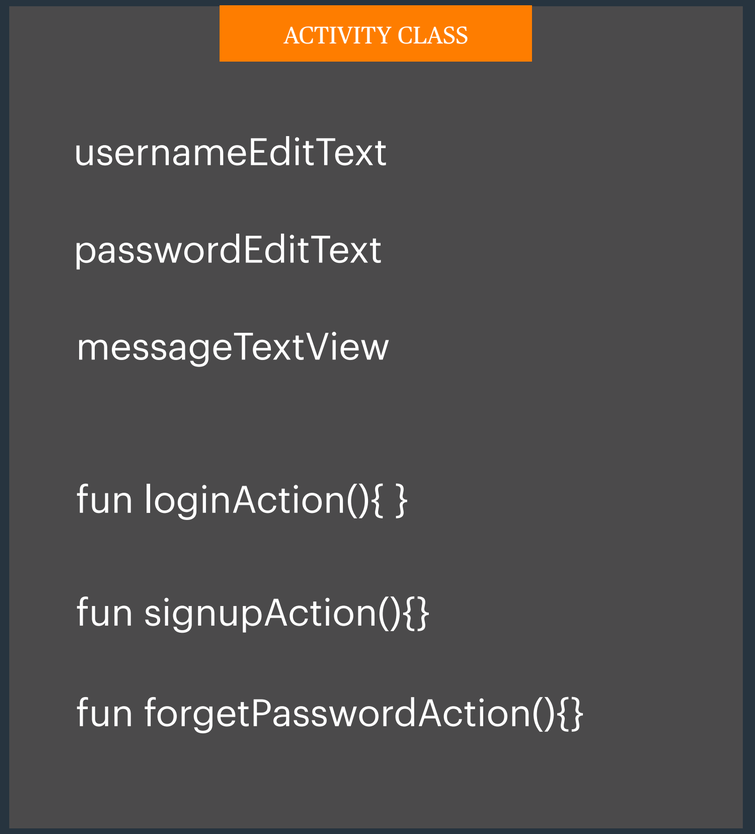
Event Sources

Functions

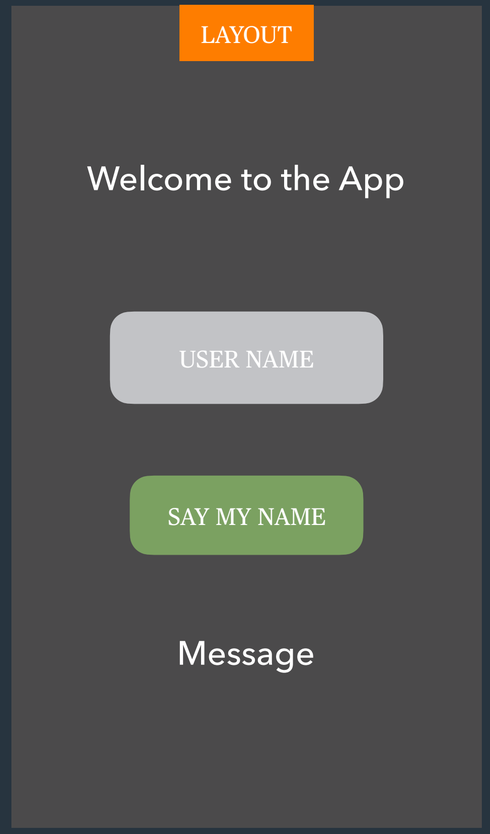
Login Example

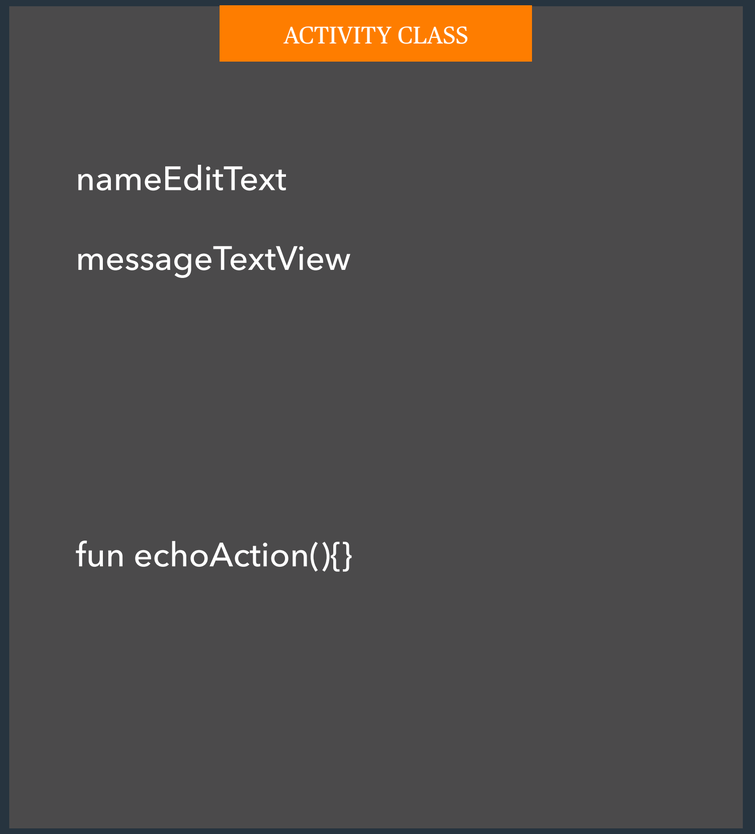




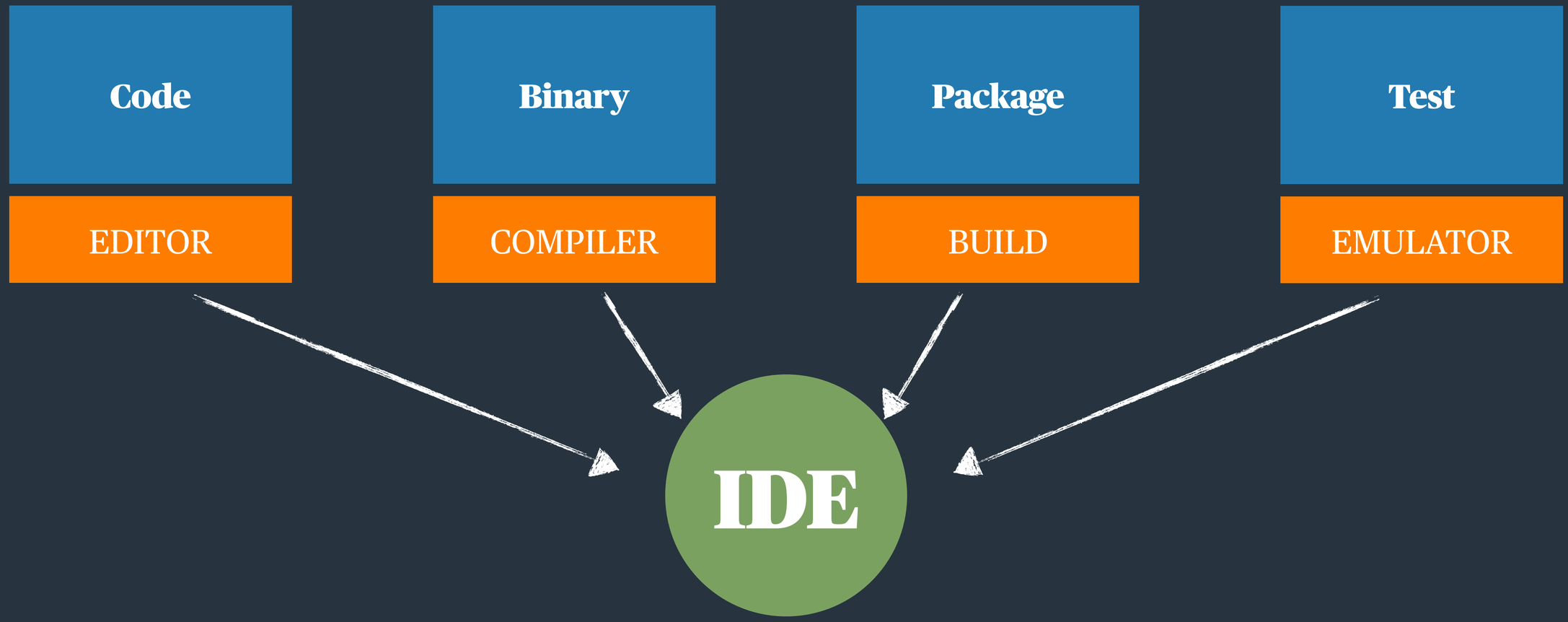


Echo Example





Development Tools



Integrated Development Environment (Android Studio)

Required Actions

- Install JDK
- Install Android Studio
- Install SDK(s)
- Create AVD(s)