**I. Function**

* Trong kotlin sử dụng từ khoá “fun" để khai báo cho function

fun getSum(x : Int, y : Int) : Int{

return x+y

}

* Parameter truyền vao theo dạng ( name : type), các parameter được phân chia bởi dấu phẩy.
* default argument:

fun getSum(x : Int = 0, y : Int = 0) : Int{

….

}

* giá trị default values sẽ được sử dụng khi parameter tương ứng bị bỏ qua. Với cách này t sẽ giảm được số lượng lớn overload method so với java
* **fun** getSum(a : Int = 10, b : Int = 10) : Int = a+b
* getSum(12) // a = 12, b = 10 => 22
* khi override method thì 2 method đó sẽ cùng sử dụng 1 default argument, và ko khai báo lại ở method con nữa.

open class A{

open **fun** getSum(a : Int = 10, b : Int = 10) : Int = a+b

}

class B : A {

override fun getSum(a: Int, b: Int) : Int = a+b // không đựợc khai báo default argument nữa

}

* đối với single-expression function (function chỉ có 1 biểu thức duy nhất) thì ta có thể viết rút gọn như sau:

fun getSum(x : Int = 0, y : Int = 0) : Int = x + y

* Member Function : là những function được khai báo trong class
* Local function: Kotlin cho phép chúng ta khai báo 1 function ở trong 1 function khác

fun getSum(x : Int, y : Int) : Int{

func validate(){

if (x!= null && y!= null ) return x+y

}

validate()

}

**II. Lambdas**

* Kotlin function có thể được lưu trữ trong 1 biến và cũng có thể truyền đi như 1 paramater.

**2.1 Higher-Order  Functions**

* higher-order function là 1 function nhận paramater truyền vào là 1 function và return 1 function khác.

fun fetchApi(callback: () -> Unit){

// do fetchApi

onSuccess(){

callback()

}

}

=> fetchApi({() -> //do some update UI })

**2.2 Lambda expression**

* ví dụ :

max(strings, { a, b -> a.length < b.length })

* Lambda expression syntax:

val sum = { x: Int, y: Int -> x + y }

* 1 Lambda Expression luôn luôn được bao quanh bởi dấu ngoặc nhọn {} .
* Trong kotlin , nếu parameter cuối cùng của function là 1 function  thì sẽ được viết như sau:

fetchApi(){ () ->

//do some update UI

}