Kotlin - Inheritance

In this chapter, we will learn about inheritance. By definition, we all know that inheritance means accruing some properties of the mother class into the child class. In Kotlin, the base class is named as “Any”, which is the super class of the ‘any’ default class declared in Kotlin. Like all other OOPS, Kotlin also provides this functionality using one keyword known as **“:”**.

Everything in Kotlin is by default final, hence, we need to use the keyword “open” in front of the class declaration to make it allowable to inherit. Take a look at the following example of inheritance.

Live Demo

```
import java.util.Arrays

open class ABC {
   fun think () {
      print("Hey!! i am thiking ")
   }
}
class BCD: ABC(){ // inheritence happend using default constructor
}

fun main(args: Array<String>) {
   var  a = BCD()
   a.think()
}
```

The above piece of code will yield the following output in the browser.

```
Hey!! i am thiking
```

Now, what if we want to override the think() method in the child class. Then, we need to consider the following example where we are creating two classes and override one of its function into the child class.

Live Demo

```
import java.util.Arrays

open class ABC {
   open fun think () {
      print("Hey!! i am thinking ")
   }
```

```
}
class BCD: ABC() { // inheritance happens using default constructor
   override fun think() {
      print("I Am from Child")
   }
}
fun main(args: Array<String>) {
   var  a = BCD()
   a.think()
}
```

The above piece of code will call the child class inherited method and it will yield the following output in the browser. Like Java, Kotlin too doesn't allow multiple inheritances.

```
I Am from Child
```