Kotlin - Sealed Class

In this chapter, we will learn about another class type called “Sealed” class. This type of class is used to represent a restricted class hierarchy. Sealed allows the developers to maintain a data type of a predefined type. To make a sealed class, we need to use the keyword “sealed” as a modifier of that class. A sealed class can have its own subclass but all those subclasses need to be declared inside the same Kotlin file along with the sealed class. In the following example, we will see how to use a sealed class.

Live Demo

```
sealed class MyExample {
   class OP1 : MyExample() // MyExmaple class can be of two types only
   class OP2 : MyExample()
}
fun main(args: Array<String>) {
   val obj: MyExample = MyExample.OP2()

   val output = when (obj) { // defining the object of the class depending on the ir
      is MyExample.OP1 -> "Option One has been chosen"
      is MyExample.OP2 -> "option Two has been chosen"
   }

   println(output)
}
```

In the above example, we have one sealed class named “MyExample”, which can be of two types only - one is “OP1” and another one is “OP2”. In the main class, we are creating an object in our class and assigning its type at runtime. Now, as this “MyExample” class is sealed, we can apply the “when ” clause at runtime to implement the final output.

In sealed class, we need not use any unnecessary “else” statement to complex out the code. The above piece of code will yield the following output in the browser.

```
option Two has been chosen
```