TutorialKart

Kotlin Class : Primary Constructor, Secondary Constructor, Init Block

Kotlin Classes & Constructors

Similar to Java, Kotlin also provides the concepts of Classes and Constructors. In addition to that Kotlin has two kinds of constructors: Primary and Secondary; and initialization blocks.

In this tutorial, we shall learn about Kotlin Class, **Kotlin Constructors** – Kotlin Primary Constructor, Kotlin Secondary Constructor, and Kotlin init block with examples.

Kotlin Class

A class is the base of object oriented programming.. A class is kind of a blue print for type of objects that belong to the class type.

Let us have a quick look into an example of a Kotlin Class to know the placement of **Kotlin Constructors**.

In the following example, we define a class named Person, with primary and secondary constructors, class variables and class methods.

example.kt

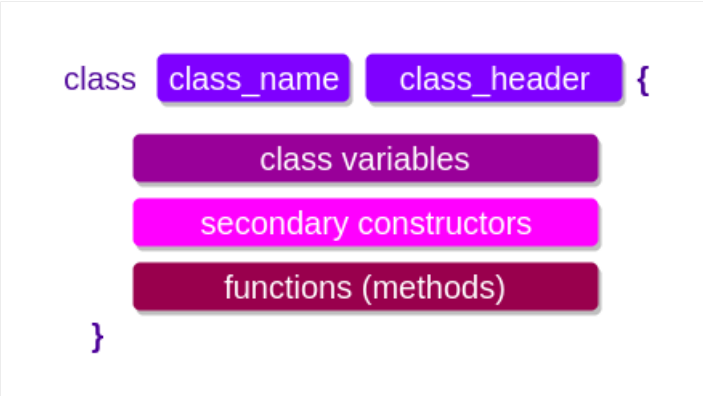
```
/**
* Kotlin Class Example and Kotlin Constructor Example
*/
class Person constructor(var name: String, var age: Int){
    // class variables
    var profession: String = "Not Mentioned"

    // secondary constructor
    constructor (name: String, age: Int, profession: String): this(name,age){
        this.profession = profession
    }

    // class method (or routine or function)
    fun printPersonDetails(){
        println("$name whose profession is $profession, is $age years old.")
    }
}
```

Structure of Kotlin Class

We shall look at the components that form a class in Kotlin



Structure of a class in kotlin

As mentioned in the picture above, a class has following three parts :

- **class** keyword followed by **class_name** class Person –**mandatory**
- **class_header –** Header of the class contains the type parameters and an implicit Kotlin Primary Constructorconstructor(var name: String, var age: Int) –**optional**
- **Body of Class** – contains **class variables**, Kotlin **Secondary Constructors** and **methods** of class. The body of the class is enclosed with curly braces after the header of class –**optional**

Kotlin Constructors

There are two types of Kotlin Constructors. They are Kotlin Primary Constructor and Kotlin Secondary Constructor.

Kotlin Primary Constructor

There could be only one **primary** constructor for a class in Kotlin. The primary constructor comes right after the class name in the header part of the class.

In the following example, we have defined a class with Primary Constructor is highlighted in the following Example :

example.kt

```
fun main(args: Array<String>){
    var person = Person("Suresh",25)
    person.printPersonDetails()
}

/**
* Kotlin Primary Constructor Example - Kotlin Class and Kotlin Constructor
```

```
* Kotlin Primary Constructor Example - Kotlin Class and Kotlin Constructor
*/
class Person constructor(var name: String, var age: Int){
    var profession: String = "Not Mentioned"

    constructor (name: String, age: Int, profession: String): this(name,age){
        this.profession = profession
    }

    fun printPersonDetails(){
        println("$name whose profession is $profession, is $age years old.")
    }
}
```

Output

```
Suresh whose profession is Not Mentioned, is 25 years old.
```

Visibility modifiers for Kotlin Primary Constructor

The default visibility on the primary constructor is public. However, the visibility can be changed to private, protected or internal. The syntax to change the visibility of Primary constructor using visibility modifier is

```
class Person visibility_modifier constructor (var name: String, var age: Int)
```

Kotlin Secondary Constructor

Constructors that are written inside the Body of Class are called Secondary constructors.

Secondary Constructor should call primary constructor using **this** keyword. From the example of Kotlin class already given, the secondary constructor is :

```
constructor (name: String, age: Int, profession: String): this(name,age){
    this.profession = profession
}
```

This secondary constructor takes three variables, but calls primary constructor using: this(name, age) to set the variables handled by the primary constructor.

In the following example, we have defined a secondary constructor.

example.kt

```
fun main(args: Array<String>){
    var person_1 = Person("Suresh",25, "Teaching")
    person_1.printPersonDetails()
}
```

```
}

/**
* Kotlin Secondary Constructor Example - Kotlin Classes and Kotlin Constructors
*/
class Person constructor(var name: String, var age: Int){
    var profession: String = "Not Mentioned"

    constructor (name: String, age: Int, profession: String): this(name,age){
        this.profession = profession
    }

    fun printPersonDetails(){
        println("$name whose profession is $profession, is $age years old.")
    }
}
```

Output

```
Suresh whose profession is Teaching, is 25 years old.
```

Visibility modifiers for Kotlin Secondary Constructor

The default visibility of secondary constructor is public. However, the visibility can be changed to private, protected or internal.

The syntax to provide visibility modifiers for Kotlin Secondary constructor is

```
visibility_modifier constructor (var name: String, var age: Int, var profession: String)

}
```

Kotlin init

If you observe the definition of primary constructor, there is no provision in the header to include some lines code for the primary constructor, except for the declaration of type variables. To fill this void, there is init block. But, care has to be taken that init block is run when the class variable is initialized. Hence, this init block is run for all the constructors irrespective of primary and secondary, and after the execution of primary constructor block. Init block is run with the context of primary constructor.

An example Kotlin program to demonstrate the working of Kotlin init block is given below :

example.kt

```
fun main(args: Array<String>){
    var person_0 = Person("Ranjan",21)
    person_0.printPersonDetails()
```

```
    var person_1 = Person("Suresh",25, "Teaching")
    person_1.printPersonDetails()
}

/**
* Kotlin Init Example
*/
class Person constructor(var name: String, var age: Int){
    var profession: String = "Not Mentioned"
    // initializer block is run during the initialization of class object, after executi
    init{
        println("$name's details are being held in this class object.")
    }

    constructor (name: String, age: Int, profession: String): this(name,age){
        this.profession = profession
    }

    fun printPersonDetails(){
        println("$name whose profession is $profession, is $age years old.")
    }
}
```

Output

```
Ranjan's details are being held in this class object.
Ranjan whose profession is Not Mentioned, is 21 years old.
Suresh's details are being held in this class object.
Suresh whose profession is Teaching, is 25 years old.
```

Conclusion

In this Kotlin Tutorial, we have learned the structure of a class in Kotlin with an example, also the types of Kotlin Constructors we have for a class: primary constructor and secondary constructor, and the role of init block in aiding primary constructor.

Kotlin Java

- Kotlin Tutorial

Getting Started

- Setup Kotlin(Java) Project
- Kotlin Example Program
- Convert Java to Kotlin
- Kotlin Main Function
- Kotlin Loops

- Kotlin For Loop
- Kotlin While, Do While Loops
- Kotlin Repeat
- Kotlin Ranges
- Kotlin When

Object Oriented Concepts

Classes

- **Kotlin - Class, Primary and Secondary Constructors**
- Kotlin Sealed Class
- Kotlin Data Class
- Kotlin Enum
- Kotlin - Extension Functions

Inheritance

- Kotlin Inheritance
- Kotlin Override Method of Super Class

Abstraction

- Kotlin Abstraction
- Kotlin Abstract Class
- Kotlin - Interfaces
- Kotlin Null Safety

Exception Handling

- Kotlin Try Catch
- Kotlin Throw Exception
- Kotlin Custom Exception

Fix Compilation Errors

- Kotlin - Variable must be initialized
- Kotlin - Primary Constructor call expected
- Kotlin - Null can not be a value of a non-null type String
- Kotlin - Cannot create an instance of an abstract class

Kotlin - String Operations

- Kotlin - Compare Strings
- Kotlin - Replace String
- Kotlin - Split String
- Kotlin - Split String to Lines
- Kotlin - String Capitalize

Kotlin - Functions

- Kotlin Function - Default Arguments
- Kotlin - Use Function

Kotlin Collections

Kotlin List

- Kotlin List
- Kotlin List forEach

Kotlin File Operations

- Kotlin - Create File
- Kotlin - Read File
- Kotlin - Read File as List of Lines
- Kotlin - Write to File
- Kotlin - Append Text to File
- Kotlin - Check if File Exists
- Kotlin - Copy a File to Other
- Kotlin - Iterate through all files in a directory
- Kotlin - Delete Recursively
- Kotlin - Get File Extension

Kotlin Interview Q/A

- Kotlin Interview Questions

Kotlin Android

- Kotlin Android Tutorial

Useful Resources

- How to Learn Programming