Constructor

**Constructor**  
A class in Kotlin can have a primary constructor and one or more secondary constructors.

**Primary Constructor**  
The primary constructor is a part of the class header, and it goes after the class name and optional type parameters

1. Primary constructor can have zero arguments

class Demo constructor(){  
  
}

1. If the primary constructor does not have any annotations or visibility modifiers, the constructor keyword can be omitted

fun main(){  
 *print*("constructor key Omitted")  
}  
class Demo (userName:String){  
}

1. The primary constructor cannot contain any code. Initialization code can be placed in initializer blocks prefixed with the init keyword

fun main(){  
 val demoInstance=Demo("Vinay T Shetty")  
}  
class Demo (userName:String){  
 val displayname=userName  
 init {  
 *println*("DisplayName= ${displayname}")  
 }  
}

1. Primary constructor parameters can be used in the initializer blocks. They can also be used in property initializers declared in the class body

fun main(){  
 val demoInstance=Demo("Vinay T Shetty")  
 demoInstance.displayuserName()  
}  
class Demo (userName:String){  
 val displayname=userName  
 init {  
 *println*("DisplayName= ${displayname}")  
 }  
 fun displayuserName(){  
 *println*("Display Username=${displayname}")  
 }  
}

1. Primary Constructor Parameters are always **local Variable** to constructor.
   1. Primary Constructor values cannot be used directly inside the method.
2. fun main(){  
    val demoInstance=Demo("Vinay T Shetty")  
    demoInstance.displayuserName()  
   }  
   class Demo (userName:String){  
    fun displayuserName(){  
    *println*("Display Username=${displayname}")  
    }  
   }  
     
   e: D:\Tutorials\Projects\Kotlin\KotlinPrj\src\main\kotlin\Test.kt: (7, 38): Unresolved reference: displayname
3. Default Values can be provided to constructor

fun main(){  
 val demoInstance=Demo("Vinay T Shetty","\*\*\*\*\*")  
 demoInstance.dispalyValues()  
 }  
 class Demo (userName:String,passWord:String,authenticagted:Boolean=true){  
 val disUserName=userName  
 val dispassWord=passWord  
 val disAuthenicagted=authenticagted  
 fun dispalyValues(){  
 *print*("UserName=${disUserName}, Password=${dispassWord} , Authenicated= ${disAuthenicagted}")  
 }  
 }  
  
/\* UserName=Vinay T Shetty, Password=\*\*\*\*\* , Authenicated= true  
 Process finished with exit code 0  
 \*/

1. We ca provide both val and var values to the constructor parameter

fun main(){  
 val demoInstanceVal=DemoVal("Vinay T Shetty")  
 val demoInstanceVar=DemoVar("Vinay")  
}  
class DemoVal (val userName:String){}  
class DemoVar(var userName:String){}

1. init Block will be executed based on written order.

fun main() {  
 val demoInst=Demo("Vinay T Shetty")  
 demoInst.displayName()  
}  
  
class Demo(username: String) {  
 val dispUsername = username  
 init {  
 *println*("Frist Intialize Block")  
 }  
 init {  
 *println*("Second Intialize Block")  
 }  
 fun displayName(){  
 *println*("Name= ${dispUsername}")  
 }  
}  
/\*  
Frist Intialize Block  
Second Intialize Block  
Name= Vinay T Shetty  
\*/

**Secondary Constructor**Secondary constructor is used to initialize the class and introduce some extra logic.  
Kotlin may have one or more secondary constructors.

fun main() {  
 val test=Test("Vinay T Shetty")  
}  
  
class Test {  
 constructor(username: String) {  
 *println*("Username= ${username}")  
 }  
}

1. Constructor Overloading

Overloading of the Constructor is Valid.

fun main() {  
 val userName=Test("Vinay T Shetty")  
 val username\_password=Test("Vinay T Shetty","\*\*\*\*\*\*")  
 val username\_password\_email=Test("Vinay T Shetty","\*\*\*\*\*","vinay@gmail.com")  
}  
  
class Test {  
 constructor(username: String) {  
 *println*("Username= ${username}")  
 }  
 constructor(username: String,password:String){  
 *println*("Username= ${username}, Password= ${password}")  
 }  
 constructor(username: String,password:String,email:String){  
 *println*("Username= ${username}, Password= ${password} , Email= ${email}")  
 }  
}

1. Primary Constructor call is expected from the Secondary constructor using **this** keyword If the user has written a Primary constructor.

Reason is by default a primary constructor will be called by compiler If user has not written.If the user has written a primary constructor then user needs to call primary constructor.

fun main() {  
 val username\_password=Test("Vinay TS")  
}  
  
class Test (projectid:Int) {  
 constructor(username: String):this(100) {  
 *println*("Username= ${username}")  
 }  
}

1. Constructor Chaining(Calling Once Constructor from another)

fun main() {  
 val userName=Test("Vinay T Shetty")  
}  
  
class Test {  
 constructor(username: String) :this("Vinay T Shetty","\*\*\*\*\*\*"){  
 *println*("Username= ${username}")  
 }  
 constructor(username: String,password:String):this("Vinay T Shetty","\*\*\*\*\*","vinay@gmail.com"){  
 *println*("Username= ${username}, Password= ${password}")  
 }  
 constructor(username: String,password:String,email:String){  
 *println*("Username= ${username}, Password= ${password} , Email= ${email}")  
 }  
}

1. S
2. T
3. E
4. S
5. T
6. E
7. S
8. t

Points To Remember

Link <https://kotlinlang.org/docs/classes.html#constructors>