**Companion Object**

whenever we want to call the method or whenever we want to access the members of a class then we make the object of the class and with the help of that object, we access the members of the class. A typical Kotlin example of the same is

fun main() {  
val test=Test()  
 test.callme()  
}  
class Test{  
 fun callme(){  
 *print*("Call Me function called")  
 }  
}

**static Keyword**

In some languages like Java and C#, we use static keyword to declare the members of the class and use them without making any object i.e. just call them with the help of class name.

class Test{

public static void main(String[] args) {

    Person.userName();

    }

}

class Person{

    public static void userName(){

        System.out.println("Person UserName= Vinay T Shetty");

    }

}

// Person UserName= Vinay T Shetty

***There is nothing called static in Kotlin. So, in Kotlin, we use a companion object***

In Kotlin, if you want to write a function or any member of the class that can be called without having the instance of the class then you can write the same as a member of a companion object inside the class

So, by declaring the companion object, you can access the members of the class by class name only(which is without explicitly creating the instance of the class).

fun main() {  
 Test.userName()  
 *println*("Password = ${Test.password}")  
 *println*("Email Id = ${Test.emailId}")  
}  
  
class Test {  
 companion object credentialsDeclaration {  
 fun userName() {  
 *println*("Username= Vinay T Shetty")  
 }  
 val password = "\*\*\*\*\*"  
 var emailId = "VinayTShetty@gmail.com"  
 }  
}

1. User Defined name for the companion object can be omitted.  
   Declaration of only ‘companion object’ is valid.

class Test {  
 companion object {  
 fun userName() {  
 *println*("Username= Vinay T Shetty")  
 }  
 val password = "\*\*\*\*\*"  
 var emailId = "VinayTShetty@gmail.com"  
 }  
}