Classes: Constructors, Auxiliary Constructors and Access Rules in Scala

By

Sardar Muhammad Sulaman

Classes in Scala

- A class is a blueprint for objects
- It defines characteristics of an object
- Characteristics are:
    - Attributes (Field or Properties)
    - Behaviors (Methods or Operations)
- Example: Car Class

<u>Attributes:</u>

- Year
- Model
- Color
- Engine etc.

<u>Behaviors:</u>

- on() or off()
- chngGears()
- Accelerate()
- Brake() etc.

Contd.

- Class declaration is like Java
- Instantiation of a class is also same like Java
- In scala we create variables either using *val* or *var*
- Using *val we get a read only variable (Immutable)*
- Example:

```
//Declaration of a class
class Hello
{
def  greet() = println("Hello Everyone!")
}

//Class instantiation
val object = new Hello()
object.greet()
```

Constructors

- They are different than Java
- In Scala *Primary constructor* is the body of class, and its parameter list comes after the class name
- Example:

//We will print the hello message on instantiation

```
class Hello(message: String)
{
Println(“Welcome”)
def  greet() = println(message)
}
val object = new Hello(“Hello Everyone!”)
object.greet()
```

// The parameter looks like a field, but it is not. The Compiler automatically creates and that is val (Immutable) with public access.

Auxiliary Constructors

- Class body is the Primary Constructor
- We can add Auxiliary Constructors
- An auxiliary constructor must call (on 1st line of its body) either another auxiliary constructor or the primary constructor
- They are created by defining methods named “this”
- Example: // add another message for aux. constructor

```
class Hello(message1: String, message2: String)
{
def this(message: String) = this(message, “ ”) // Calling primary cons.
def  greet() = println(message1 + message2)
}
val object = new Hello(“Hello”)
object.greet()
```

Access Rules in Scala

- Public:
  - Public is Scala's default access level
  - Members can be accessed from outside
- Private:
  - Private is similar to Java
  - Member labeled *private* only visible inside the class or object
  - It used in *inner classes*, which is different from Java
- Protected:
  - A bit more restrictive than Java
  - A protected member is only accessible from *subclass of the class* in which the member is defined (In Java it is accessible with in a package)

Thanks