**Lazy val**

The difference between them is, that a val is executed when it is defined whereas a lazy val is executed when it is accessed the first time.

var x = { println("x"); 15 }

lazy val y = { println("y"); x+1 }

println("-----")

x = 17

println("y is: " + y)

Output of above code is:

x

-----

y

y is: 18

**Objects**

* Objects are single instances of their own definitions. You can think of them as singletons of their own classes.
* You can define objects with the object keyword.

object IdFactory {

private var counter = 0

def create(): Int = {

counter += 1

counter

}

}

* You can access an object by referring to its name.

val newId: Int = IdFactory.create()

println(newId) // 1

val newerId: Int = IdFactory.create()

println(newerId) // 2

**Singleton Objects**

* An object is a class that has exactly one instance. It is created lazily when it is referenced, like a lazy val.
* As a top-level value, an object is a singleton.
* As a member of an enclosing class or as a local value, it behaves exactly like a lazy val.

**Companion objects**

* An object with the same name as a class is called a companion object.
* Conversely, the class is the object’s companion class.
* A companion class or object can access the private members of its companion.
* static members in Java are modeled as ordinary members of a companion object in Scala.
* When using a companion object from Java code, the members will be defined in a companion class with a static modifier. This is called static forwarding. It occurs even if you haven’t defined a companion class yourself.

(factory methods???)

**Traits**

* Traits are used to share interfaces and fields between classes.
* They are similar to Java 8’s interfaces.
* Classes and objects can extend traits but traits cannot be instantiated and therefore have no parameters.
* Use the extends keyword to extend a trait. Then implement any abstract members of the trait using the override keyword
* **Subtyping**: Where a given trait is required, a subtype of the trait can be used instead.