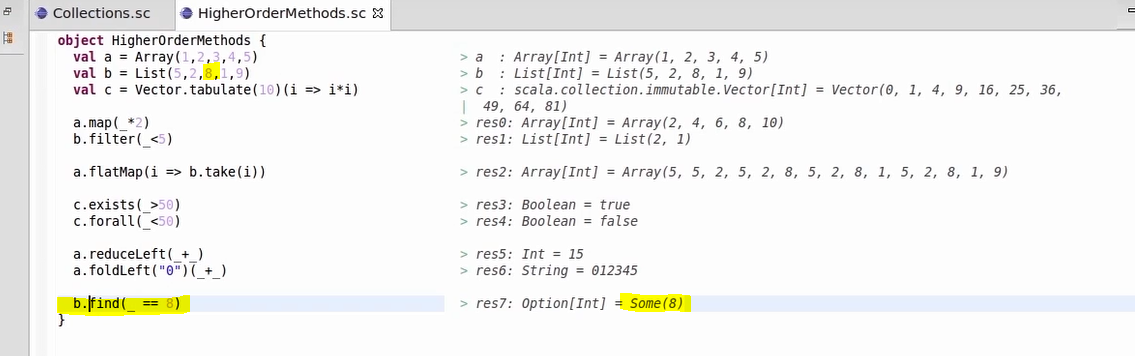
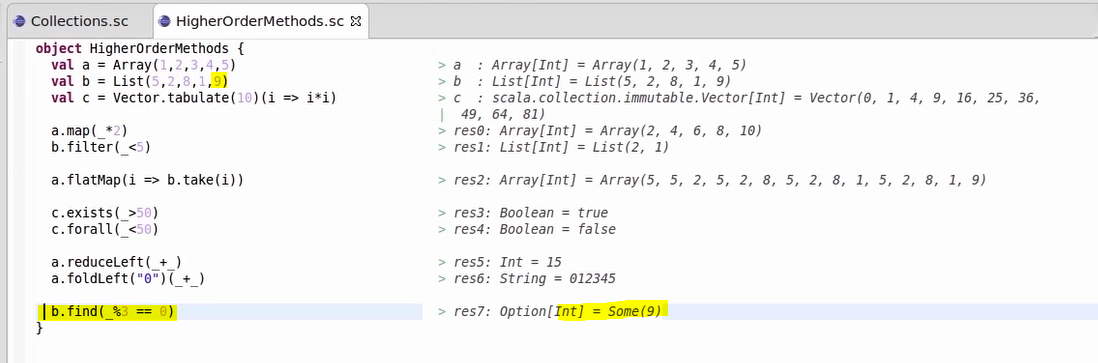
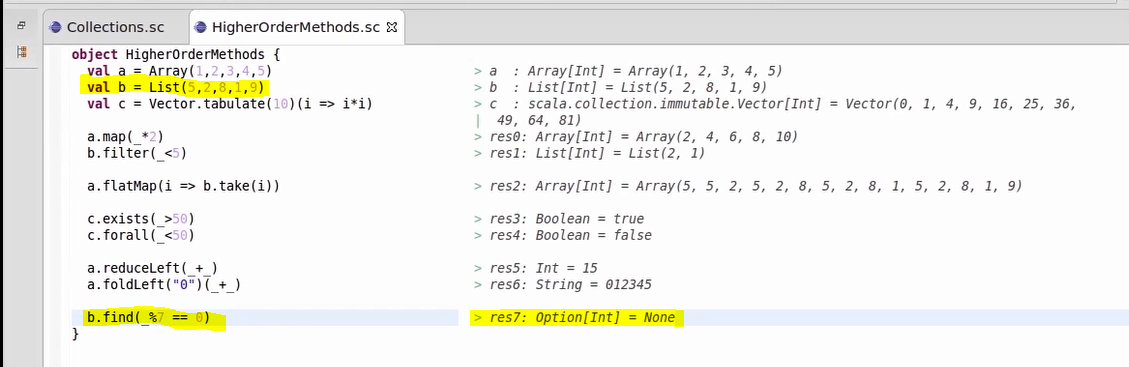
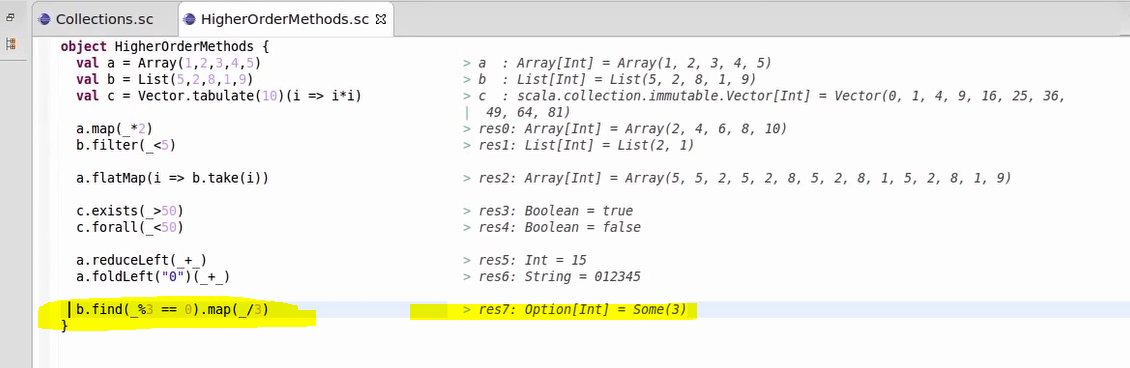
**Lecture 23: Find an Option.**

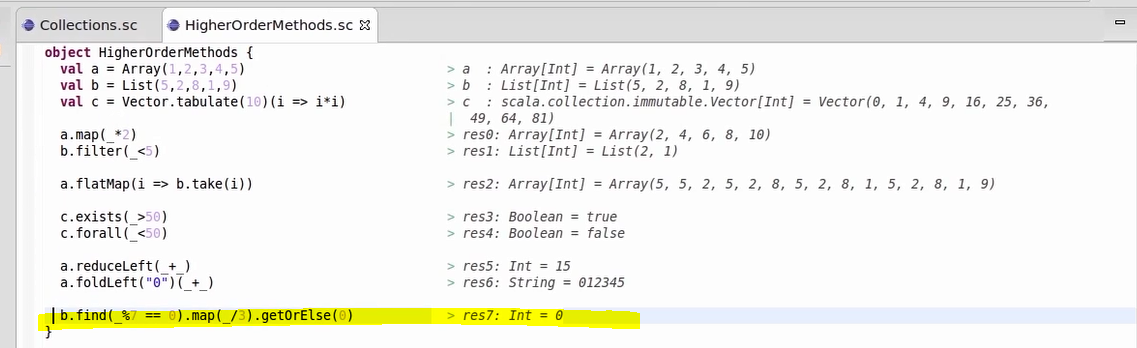






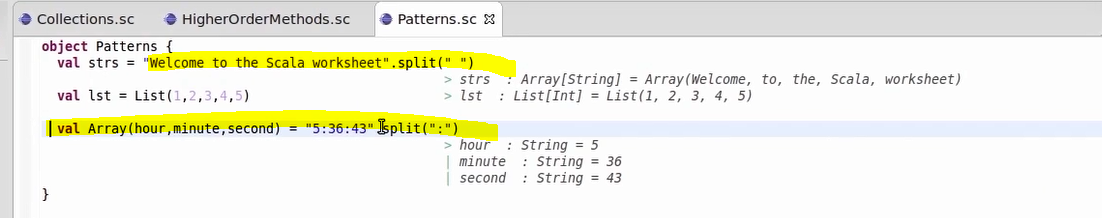


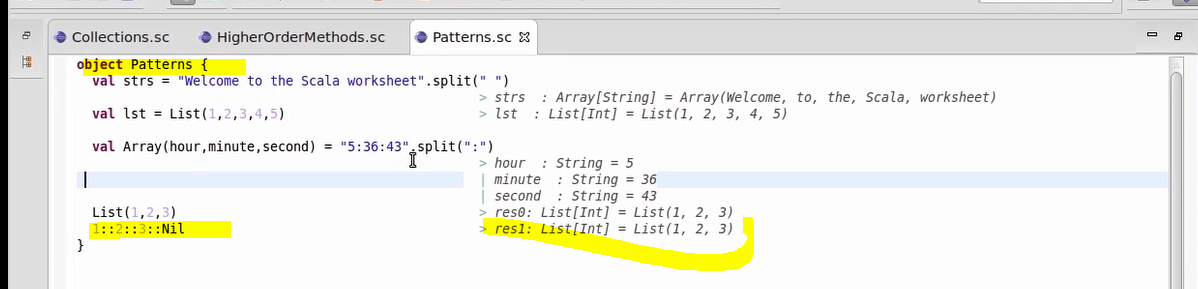
We got some value from find function, we are then dividing it by 3 using map function. If the result is present then you get else you get none.



If value is present then print else get O from getOrElse(0) method.

**Lecture 24: Pattern Matching.**

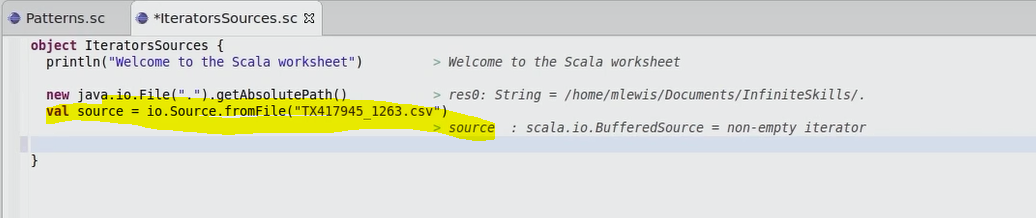




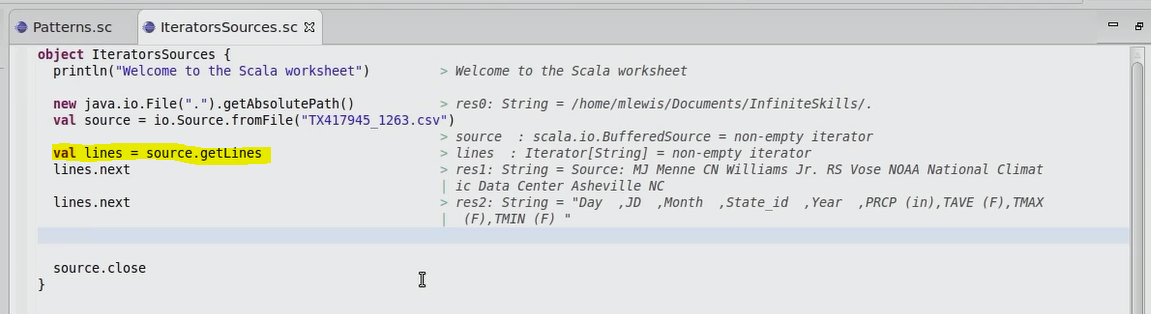
We can build list using cons and list is returned.

Cons can be used in pattern matching also.

**Lecture 24: Iterators and read file in scala**

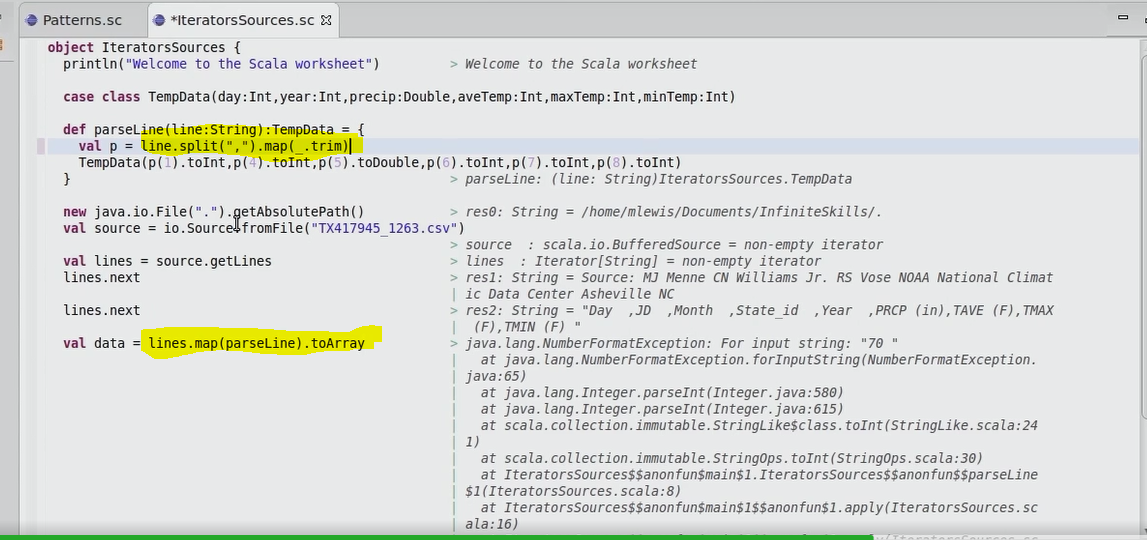


Java.io.file is just to know the directory where it is being pulled from. You can ignore it if you want.



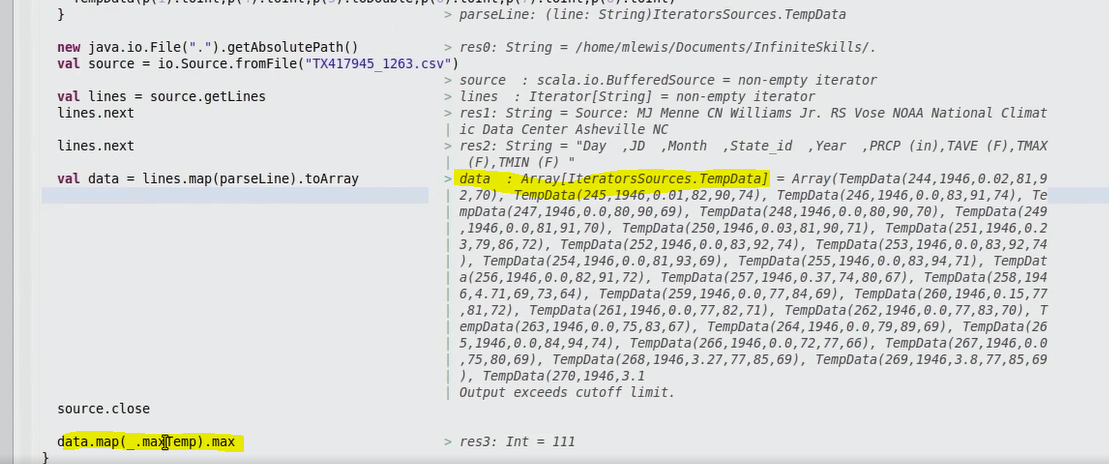
Reading line in Scala from the above program. You have to close the file as you opened it.

**Lecture 26: Iterators and read file in scala**



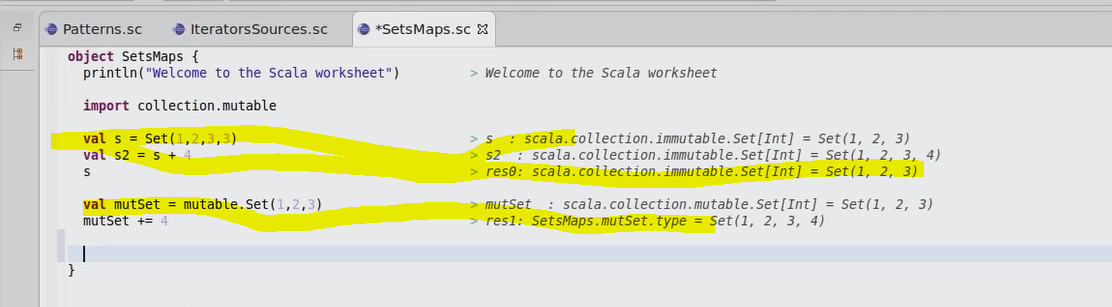
You are reading each and every line and then splitting. Splitted value is then passed to map function as an array which is inturns trim it again.

Lines.map(parseLine).toArray we got class object which is converted using toArray method.



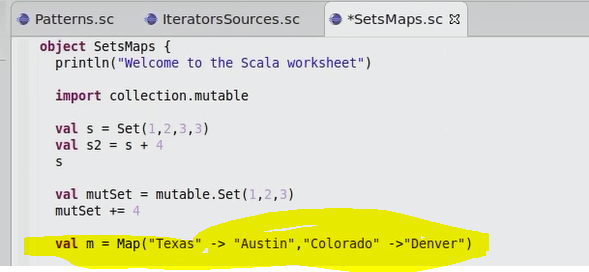
Mapping the array in each element with maxtemp and finding the max value.

**Lecture 27: Sets and Maps**



**By default set is immutable, you have to import mutable to make them writable. See above screenshot both import and then use method to make mutable.**

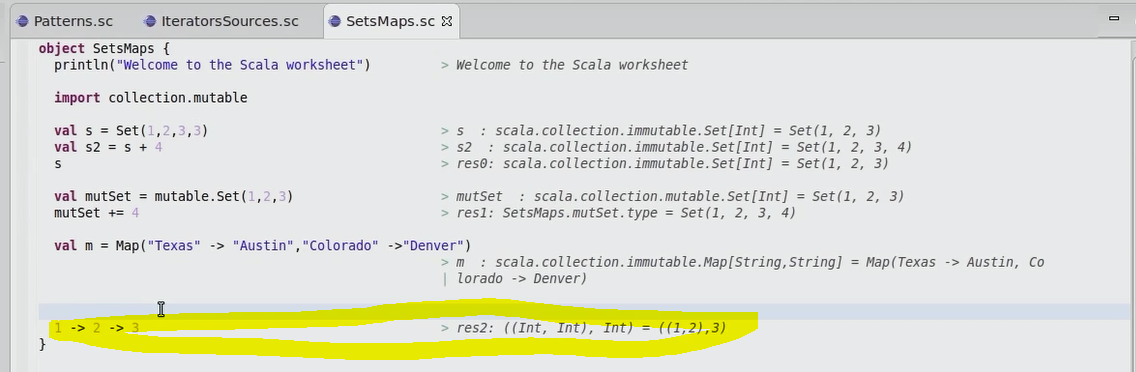
* **Makes tuples in scala . you can use it anywehere**

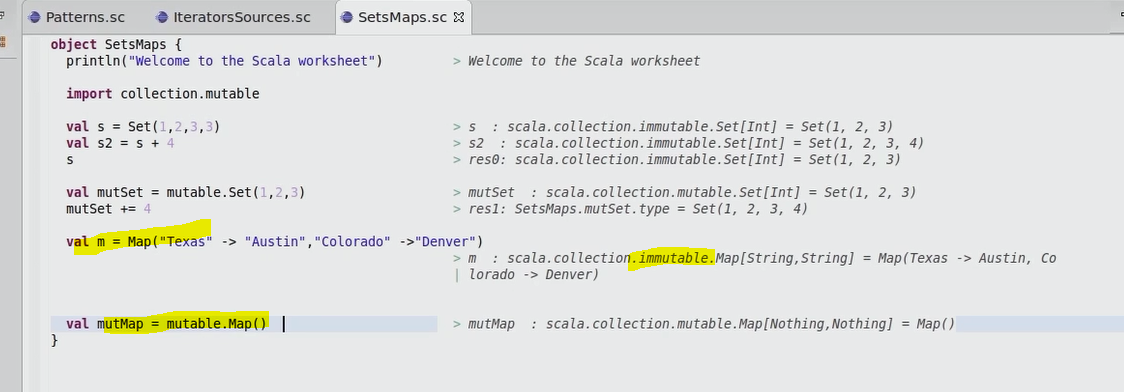


**It always tuple for 2 elements ->**

**So in below case , first it will return tuple for 2 and then result of tuple with another elements as again tuple**

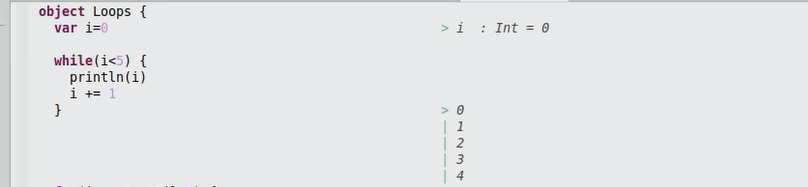
**1 ->2 ->3 tuple(tuple (1 and 2),3)**

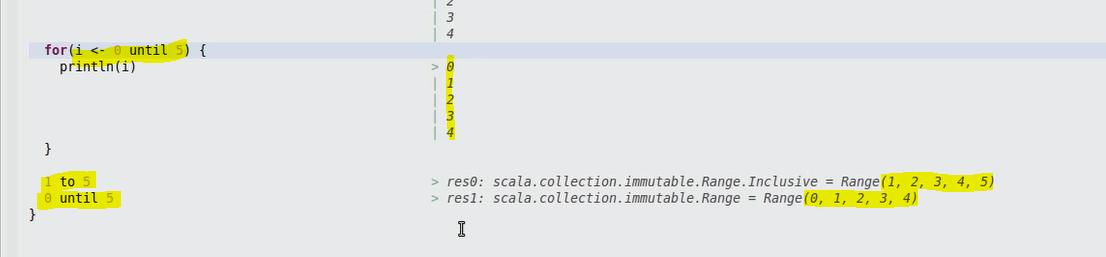


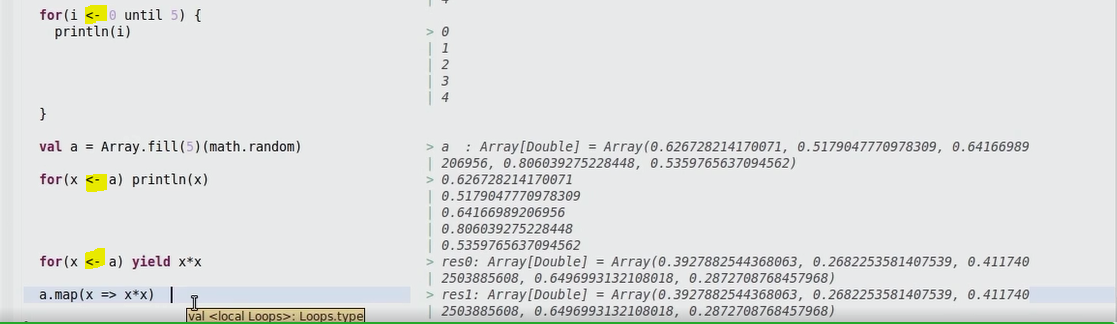


**By default, map is also immutable and to make it mutable use the mutable package.**

**Lecture 28: Loops**

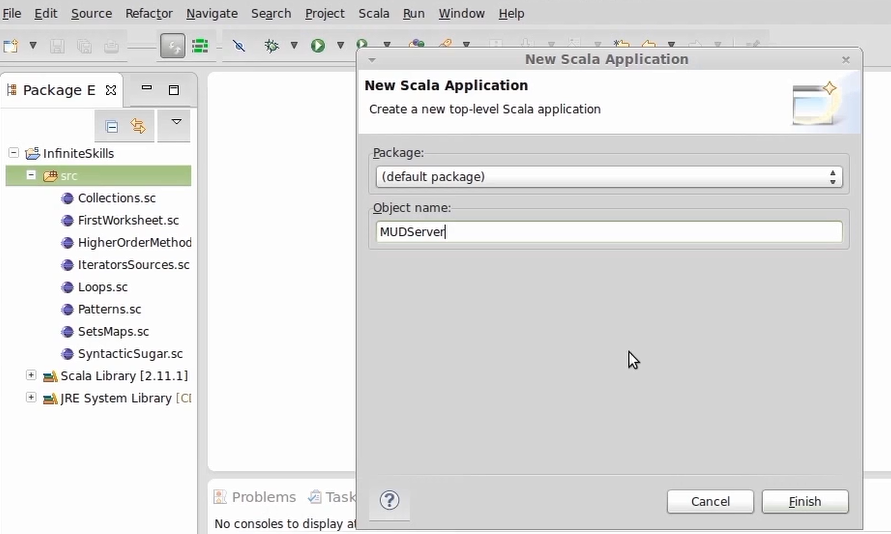


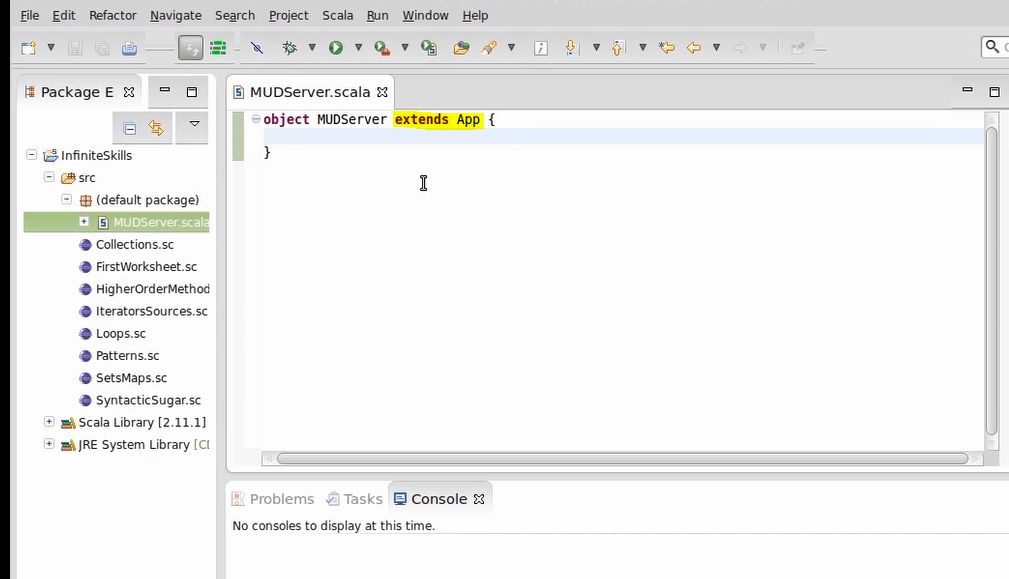




**Lecture 32-Building an APP**

**Right click on the project to create new class which is an app.**





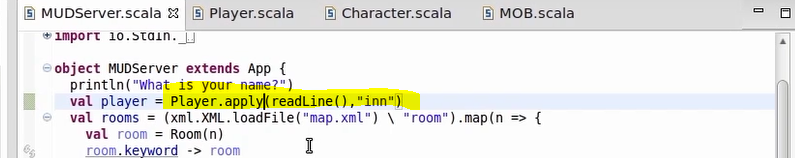
**If you extend app, you don’t need to mention static void main class.**

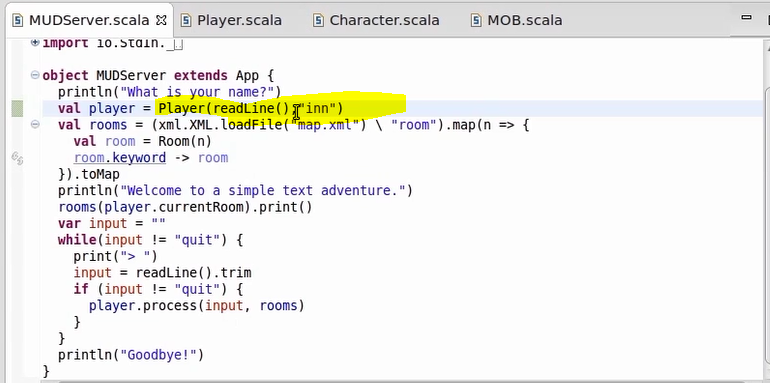
**\*\*\*\* companion objects -🡪 apply method like**

**Array(1,2,3) is equal to Array.apply(1,2,3)**

**Player is companion object which has apply method. Scala will take apply by default by () symbols.**

**Scala treats all functions as objects so player is an object i.e. companion object in this case.**





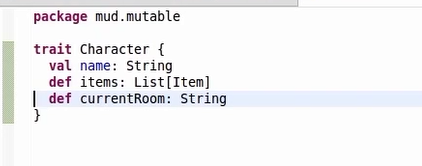
**You don’t need to mention apply, it takes by default. We are calling functions as objects which indirectly has apply method. Apply method present in Player class with companion player object.**

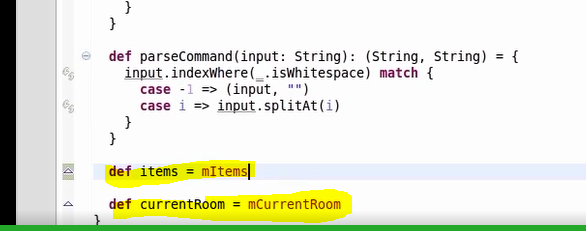


**Traits cannot take arguments. It may change in new versions. Also classes extends traits not implements.**

**In java 8, you can add default implementations to abstract methods inside interfaces.**

**Class can accept arguments but class definition will not have arguments but in scala we have it.**





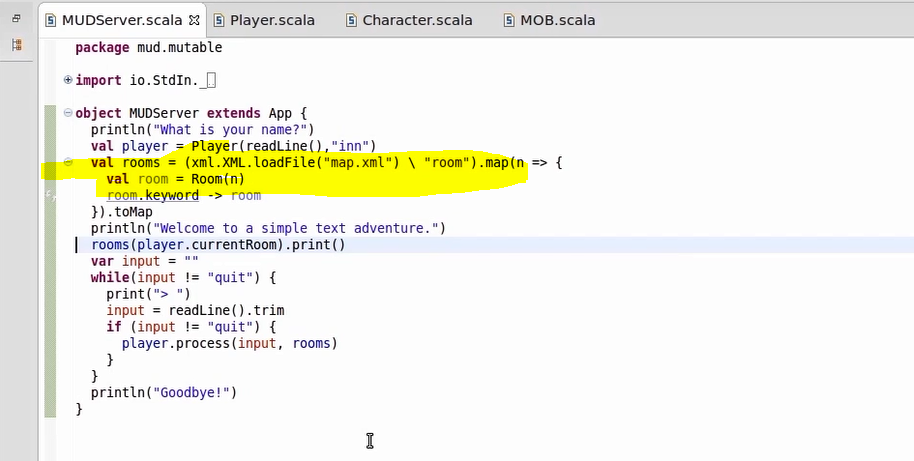
**Values are overridden in the bottom for scala.**

**In scala, you can override values, methods, type declarations but in java only methods.**

**Class can take arguments.**

**What is primary constructor?**

**Lecture 34: Imperative search**



**Room is also companion object. This code uses xml and we will look at it later point of time.**

