Scala Case Classes and Case Object

Scala case classes are just regular classes which are immutable by default and decomposable through pattern matching.

It uses equal method to compare instance structurally. It does not use new keyword to instantiate object.

All the parameters listed in the case class are public and immutable by default.

Syntax

```
case class className(parameters)
```

Scala Case Class Example

```
case class CaseClass(a:Int, b:Int)


object MainObject{

   def main(args:Array[String]){

       var c =  CaseClass(10,10)      // Creating object of case class

       println("a = "+c.a)              // Accessing elements of case class

       println("b = "+c.b)

   }

}
```

Output:

```
a = 10

b = 10
```

Case classes support pattern matching. So, you can use that in patterns. Following is the example of case classes and pattern.

Scala Case Class and Pattern Matching Example

A case class which has no arguments is declared as case object instead of case class. case object is serializeable by default.

```
trait SuperTrait

case class CaseClass1(a:Int,b:Int) extends SuperTrait

case class CaseClass2(a:Int) extends SuperTrait         // Case class

case object CaseObject extends SuperTrait               // Case object

object MainObject{

   def main(args:Array[String]){

      callCase(CaseClass1(10,10))

      callCase(CaseClass2(10))

      callCase(CaseObject)

   }

   def callCase(f:SuperTrait) = f match{

      case CaseClass1(f,g)=>println("a = "+f+" b ="+g)

      case CaseClass2(f)=>println("a = "+f)

      case CaseObject=>println("No Argument")

   }

}
```

Output:

```
a = 10 b =10

a = 10

No Argument
```