Scala Abstract Class and Trait:

What is Abstraction?

It is a process where we only provide the functionality to the user and **hide all the complex implementation** from user

In Scala we can achieve abstraction in two ways

- Abstract Class
- Trait

Abstract Class

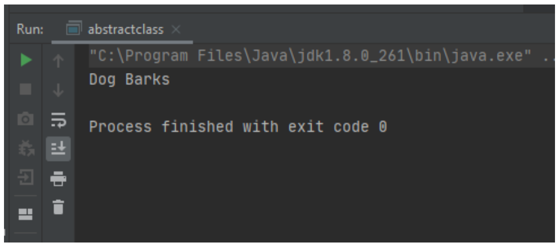
A class which is declared with **abstract keyword** is known as abstract class. And it can have abstract methods and non-abstract methods as well.

Example

```
abstract class animals{
  def sound()
}

class dog extends animals{
  def sound(){
println("Dog Barks")
  }
}

object  abstractclass{
  def main(args: Array[String]){
    var h = new dog()
h.sound()
  }
}
```

In this example we have made class animals as abstract and made a function in it as sound(), but not defined in it and then defined it in another class then made its object and called the function

Output



```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ..
Dog Barks

Process finished with exit code 0
```

NOTE: If we make a function in abstract class and not define it, compiler will show the error

Trait

It is a **collection** of abstract and non-abstract methods. We can make trait that can have all abstract methods or some abstract and some non-abstract methods.

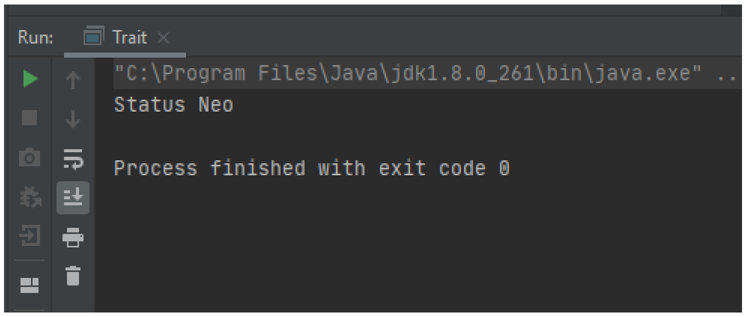
We can also consider traits as **interfaces** is java

A variable that is declared either by using **val or var keyword** in a trait get internally implemented in the class that implements the trait. Any variable which is declared by using val or var but **not initialized is considered abstract**.

Trait Example

```
trait class1{
  def print()
}

class class2 extends class1{
  def print(){
println("Status Neo")
  }
}

object Trait{
  def main(args:Array[String]){
val x = new class2()
x.print()
  }
}
```

Output



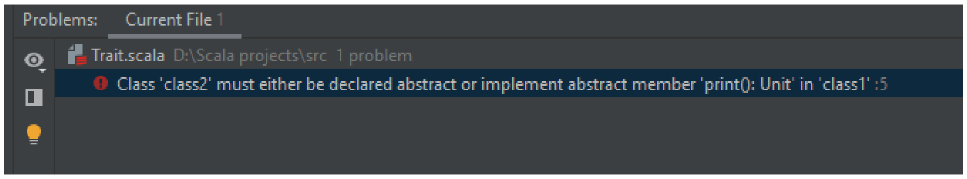
```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ..
Status Neo

Process finished with exit code 0
```

It better understandable from the above example, but there is one condition if a class extends a trait but does not implement the function declared in that trait, then it must be declared as abstract. Otherwise it will give the error

Example

```
trait class1{
  def print()
}

class class2 extends class1{
  def print1(){ //here we have not defined print()
println("Status Neo")
  }
}

object Trait{
  def main(args:Array[String]){
val x = new class2()
x.print()
  }
}
```

Output



```
Trait.scala D:\Scala projects\src 1 problem
Class 'class2' must either be declared abstract or implement abstract member 'print(): Unit' in 'class1' :5
```

NOTE: We can treat trait as abstract class alsoi.e we can also define methods in trait just like the abstract class. In Scala, trait is almost same as abstract class except that it can't have constructor. You can't extend multiple abstract classes but can extend multiple traits.

Scala Access Modifier:

Three types have been defined here

a. No modifier
b. Protected
c. Private

No modifier is the same as the public in java. In scala, we don't use the Public keyword anymore as it takes public as a default.
Here ***Protected Access Modifier*** can be accessed in Class, Subclass, Companion while in Private within a class only.

Example With private Access Modifier

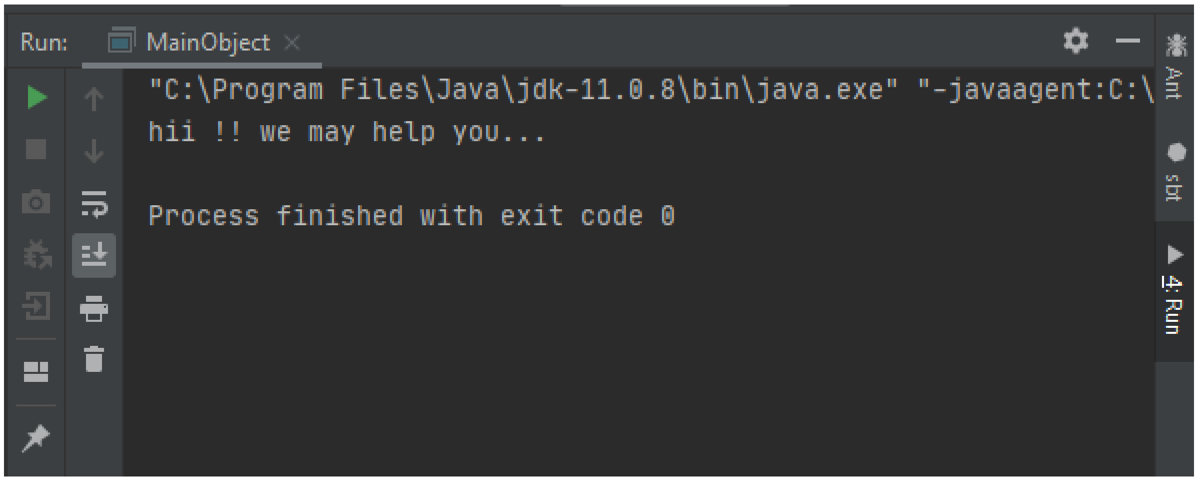
```
class Statusneo{
  private var a:String = "hii!! WE are here to help you..."
  def show(){
println(a)
  }
}
object MainObject{
  def main(args:Array[String]){
    var p = new Statusneo()
p.a = "hii !! we may help you..."
p.show()
  }
}
```

It throws the Error because of the private variable which we are trying to access it. Now with private access modifier.

ERROR: variable a in class Statusneo cannot be accessed as a member of Statusneo from object MainObjectp.a = "hii !! we may help you..."

```
class Statusneo{
   var a:String = "hii!! WE are here to help you..."
  def show(){
println(a)
  }
}
object MainObject{
  def main(args:Array[String]){
    var p = new Statusneo()
p.a= "hii !! we may help you..."
p.show()
  }
}
```

Output



```
"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\
hii !! we may help you...

Process finished with exit code 0
```

Scala Arrays:

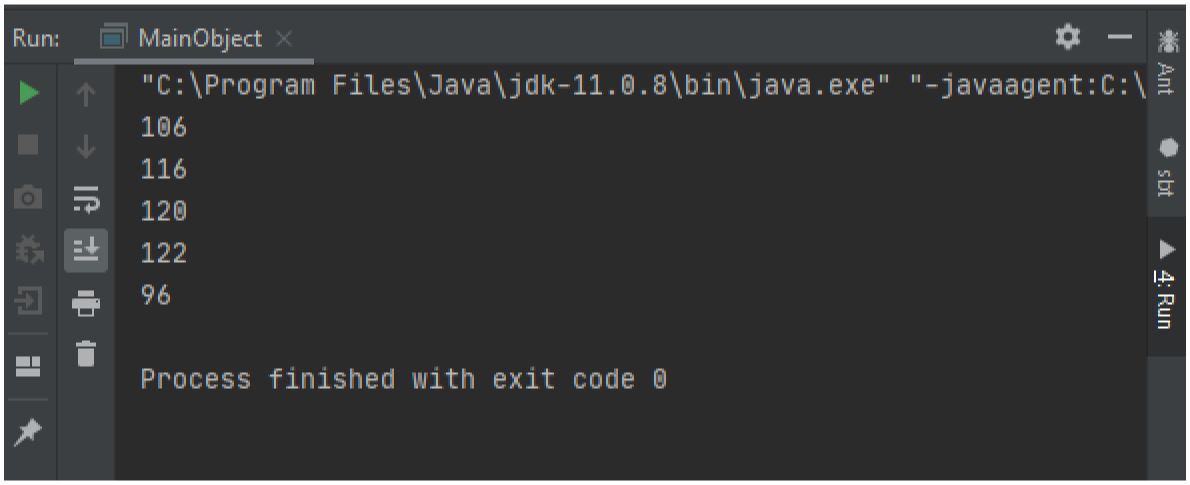
In Arrays data is stored in a sequential manner which is also known as contiguous memory Space. We can easily navigate the array. Scala arrays can be generic.
Arrays can be two-dimensional, three Dimensional i.e Multi-Dimensional array.
An array can be passed through a function.

Lets take an Example

Case-1 : Single Dimensional array
Var/valarr = Array(1,2........)

```
class MY_Array{
  var arr= Array(106,116,120,122,96)
  def Display(){
    for(a<-arr)
println(a)
  }
}
object MainObject{
  def main(args:Array[String]){
    var a = new MY_Array()
a.Display()
  }
}
```

Output

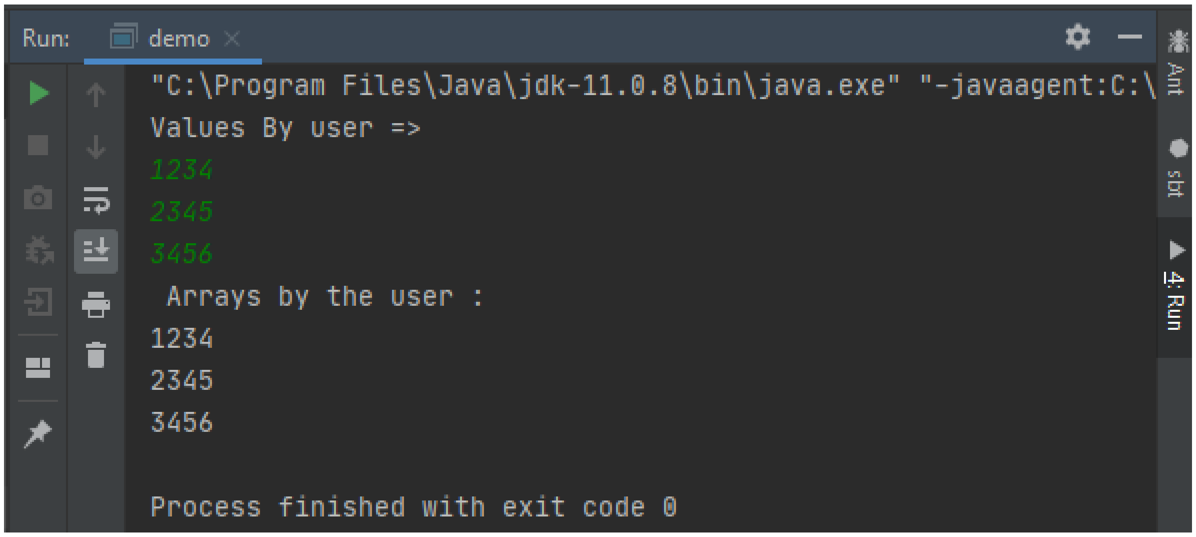


```
"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\
106
116
120
122
96

Process finished with exit code 0
```

Case-2 : Let’s take Array value by user Input

```
object demo {
  def main(args:Array[String]) {
valarr = new Array[Int](3)
println("Values By user => ")
    for (i<- 0 to 2) {
arr(i) = scala.io.StdIn.readInt()
    }
println(" Arrays by the user : ")
arr.foreach(println)
  }
}
```

Output



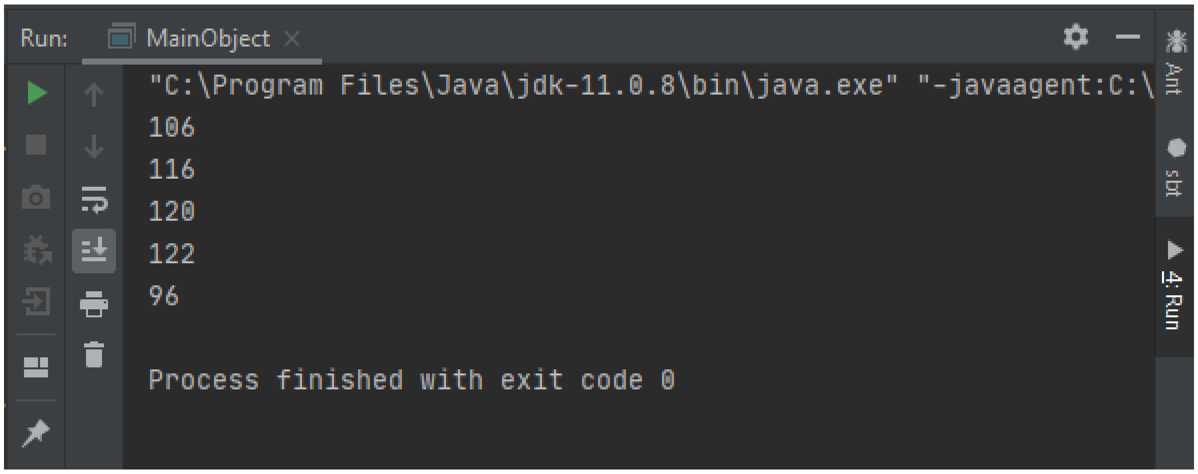
```
"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\
Values By user =>
1234
2345
3456
 Arrays by the user :
1234
2345
3456

Process finished with exit code 0
```

Case-3: Array Navigation through Iteration by For and Foreach Loop.

For-Loop:

```
class MY_Array{
  var arr= Array(106,116,120,122,96)
  def Display(){
    for(a<-arr)
println(a)
  }
}
object MainObject{
  def main(args:Array[String]){
    var a = new MY_Array()
a.Display()
  }
}
```

Output

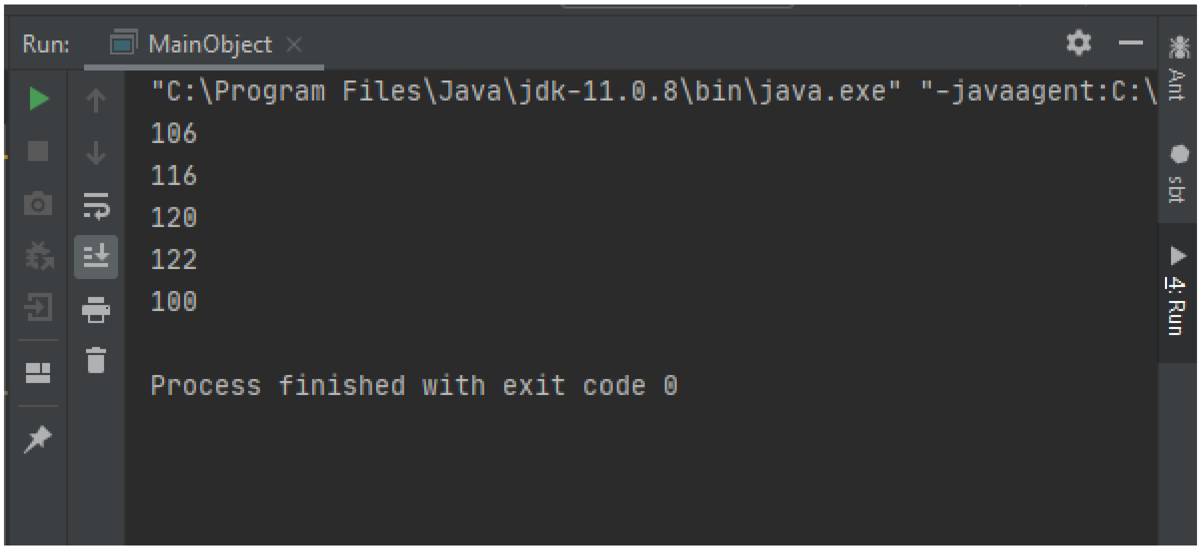


```
"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\
106
116
120
122
96

Process finished with exit code 0
```

Foreach-Loop:

```
class MY_Array{
  var arr= Array(106,116,120,122,100)
  def Display(){
arr.foreach((element:Int)=>println(element))
  }
}
object MainObject{
  def main(args:Array[String]){
    var a = new MY_Array()
a.Display()
  }
}
```

Output



```
"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\
106
116
120
122
100

Process finished with exit code 0
```

Multlidimentional Array:

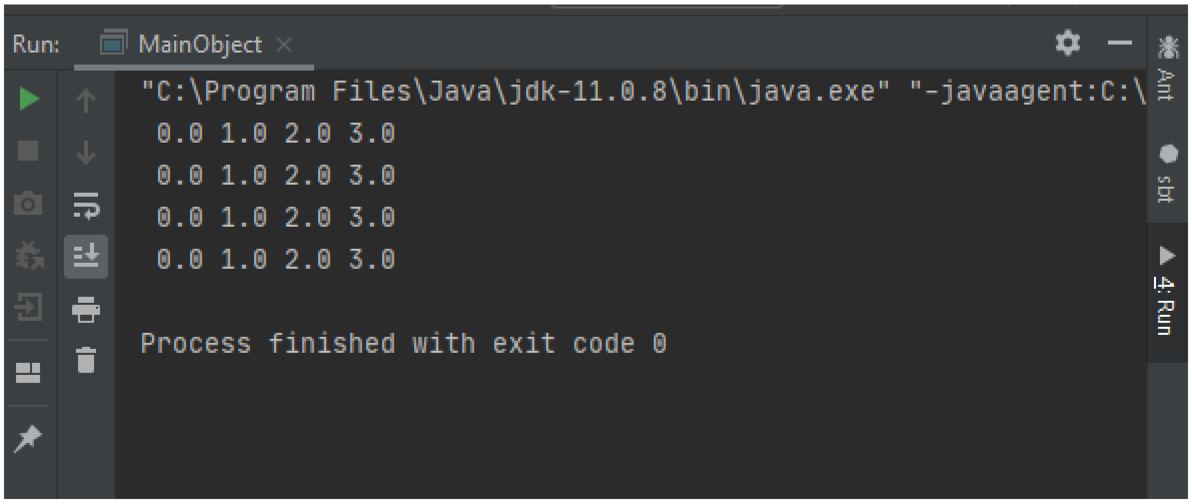
Here Array is stored in the Matrix Form. We can Add Arrays. We can have multiple dimensional of an array by “Array of array” will see in the below Examples.

Either we can use ofDim() Builtin function OR we can use Array of an array to have Multlidimentional Array.

Let’s try an Example by Using ofDim() =>

```
import Array._

object MainObject {
  def main(args: Array[String]) {
    var Multi_Dimension = ofDim[Double](4,4)

    for(x <- 0 to 3) {
      for( y <- 0 to 3) {
Multi_Dimension(x)(y) = y;
      }
    }
    for (x <- 0 to 3) {
      for ( y <- 0 to 3) {
print(" " + Multi_Dimension(x)(y));
      }
println();
    }
  }
}
```

Output

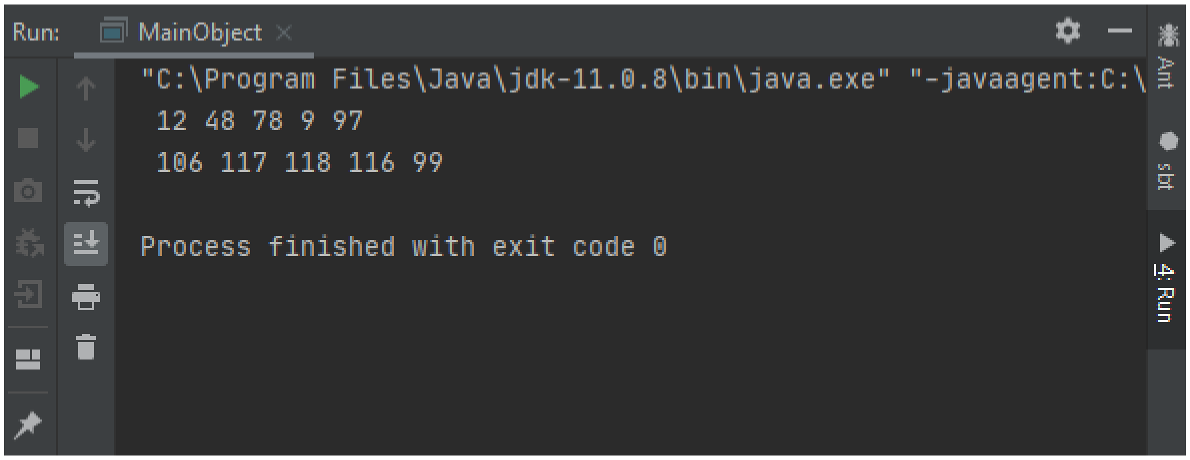


```
"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\
 0.0 1.0 2.0 3.0
 0.0 1.0 2.0 3.0
 0.0 1.0 2.0 3.0
 0.0 1.0 2.0 3.0

Process finished with exit code 0
```

Let’s take an Example by Using Array Of Array =>

```
class Dimentional_Array{

  //Array of Array
  var a = Array(Array(12,48,78,9,97), Array(106,117,118,116,99))

  def Arrays(){
    for(i<- 0 to 1){
      for(j<- 0 to 4){
print(" " + a(i)(j))
      }
println() // space after a complete row of array list
    }
  }
}
object MainObject{
  def main(args:Array[String]){
    var a = new Dimentional_Array()
a.Arrays()
  }
}
```

Output



```
"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\
 12 48 78 9 97
 106 117 118 116 99

Process finished with exit code 0
```

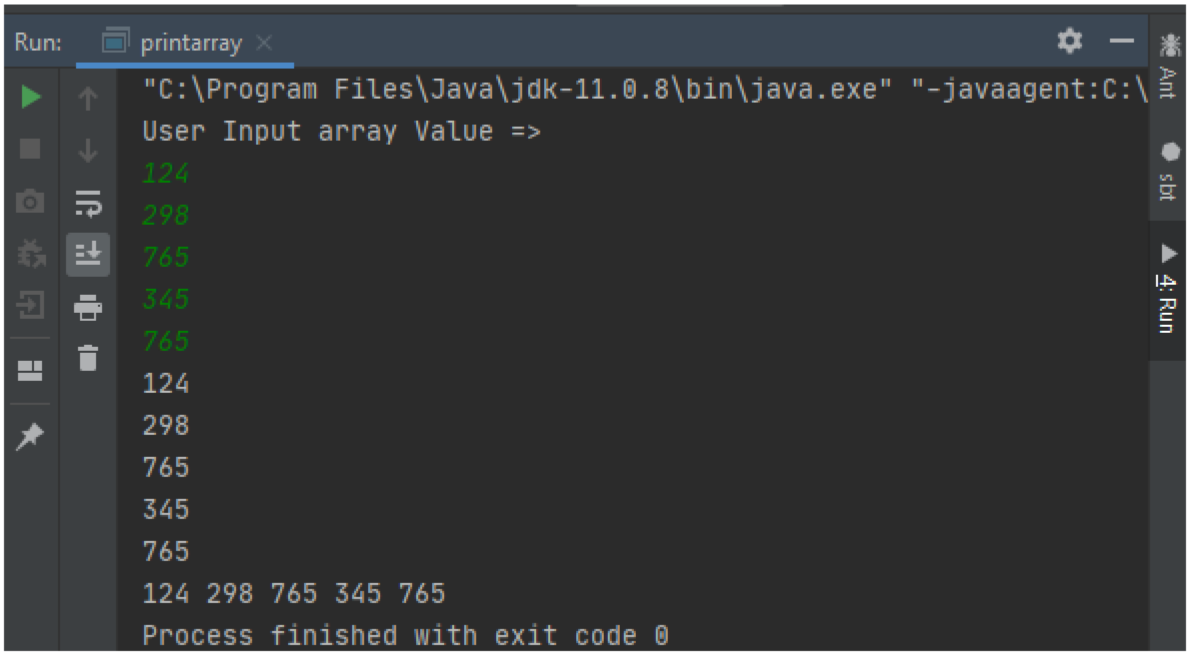
Here we can store arrays value by our choice as compare to ofdim Method in the above example until we take user input arrays value. We can do a lot of things using array-like

Concatenation, Adding up two Two Dimensional arrays much more. The more you explore the more you gain.

User Input Array Value:

```
object printarray {
  def main(args:Array[String]) {
valarr = new Array[Int](5)
println("User Input array Value => ")
    for (i<- 0 to 4) {
arr(i) = scala.io.StdIn.readInt()
    }
arr.foreach(println)

    for(a<-arr)
print(a + " ")
  }
}
```

Output



```
"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\
User Input array Value =>
124
298
765
345
765
124
298
765
345
765
124 298 765 345 765
Process finished with exit code 0
```

Scala String:

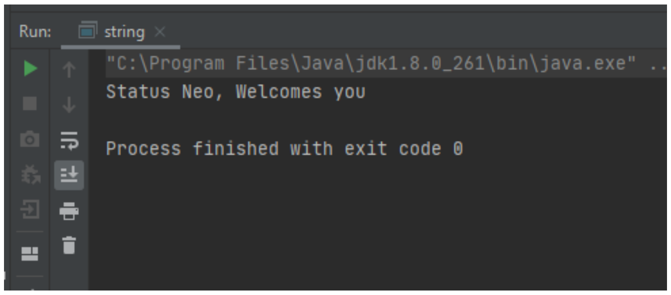
String is combination of characters. We can say it is a type of data structure which uses index and linear approach to store data. It is also immutable just like java which means we cannot change the original string but we can manipulate it by applying different methods

Let’s understand string with a basic example

```
class basic{
  var x = "Status Neo, Welcomes you"
  def print(){
println(x)
  }
}

object string{
  def main(args:Array[String]){
    var a = new basic()
a.print()
  }
}
```

here we have just made a class basic and defined a string x and then printed it by making an object of that class

Output



```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ..
Status Neo, Welcomes you

Process finished with exit code 0
```

Scala String Methods

There are many methods using which we can manipulate the string some of them are

- equals()
- compareTo()
- concat()
- substring()

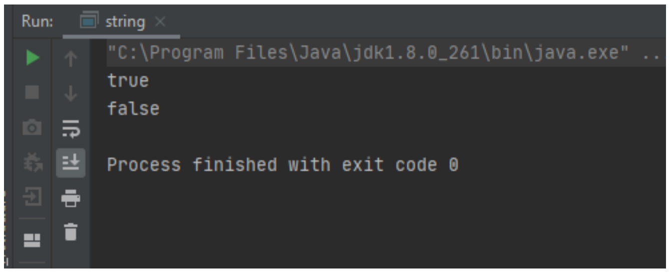
Scala String equals() Method

This method compares two strings and depending on the condition whether it is equal or not returns true or false

Example

```
class equal{
  var s1 = "welcome to status neo"
  var s2 = "status neo"
  var s3 = "status neo"
  def print(){
println(s2.equals(s3))
println(s1.equals(s2))
  }
}

object string{
  def main(args:Array[String]){
    var a = new equal()
a.print()
  }
}
```

Output



```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
true
false

Process finished with exit code 0
```

Scala compareTo() Method

This method compares the given string with current string lexicographically and then returns accordingly

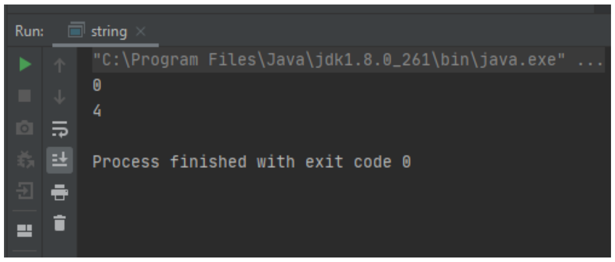
If given string is greater than current string, it returns positive number (difference of character value). If given string is less than current string, it returns negative number and if give string is equal to current string, it returns 0.

- s1 > s2 => positive number
- s1 < s2s2 => negative number
- s1 == s2 => 0

Example

```
class compare{
  var s1 = "welcome to status neo"
  var s2 = "status neo"
  var s3 = "status neo"
  def print(){
println(s2.compareTo(s3))
println(s1.compareTo(s2))
  }
}

object string{
  def main(args:Array[String]){
    var a = new equal()
a.print()
  }
}
```

Output


```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
0
4

Process finished with exit code 0
```

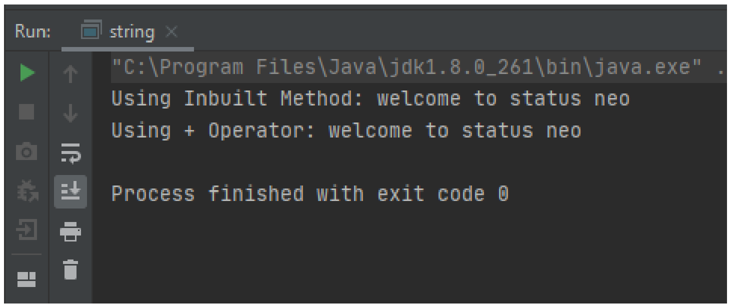
Scala String concat() Method

we can either use + operator or we can also use in built method **concat()**, to concatenate two strings

Example

```
class concat{
  var s1 = "welcome to"
  var s2 = " status neo"
  var s3= s1+s2    //using + operator
  def print(){
println("Using Inbuilt Method: " +s1.concat(s2))  //using inbuilt method
println("Using + Operator: "+ s3)
  }
}

object string{
  def main(args:Array[String]){
    var a = new equal()
a.print()
  }
}
```

Output


```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" .
Using Inbuilt Method: welcome to status neo
Using + Operator: welcome to status neo

Process finished with exit code 0
```

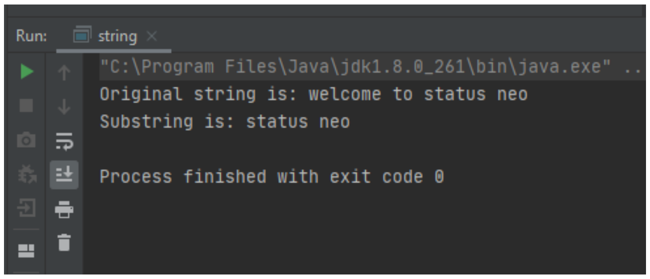
Scala substring() Method

This method is used to get a substring from a given string by providing the start and end index of the string as argument

Example

```
class substring{
  var s1 = "welcome to status neo"
  def print(){
println("Original string is: "+"welcome to status neo")
println("Substring is: "+ s1.substring(11,21))
  }
}

object string{
  def main(args:Array[String]){
    var a = new equal()
a.print()
  }
}
```

Output



```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ..
Original string is: welcome to status neo
Substring is: status neo

Process finished with exit code 0
```