

Closures

Introduction

- ❖ When defining a function, as in the case of defining a lambda expression, we can refer variables other than the function parameters.

...

```
(x:Int) => x + sum
```

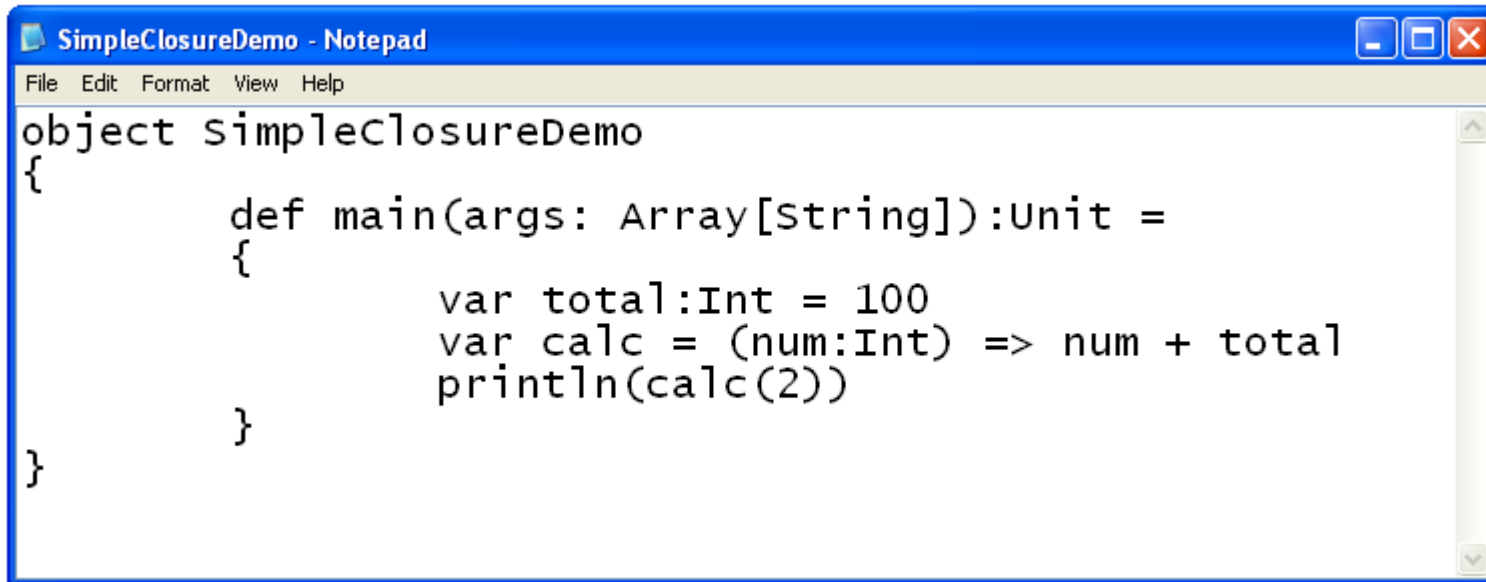
...

- ❖ The other variables must be available within the scope of the defined function.

What is a Closure?

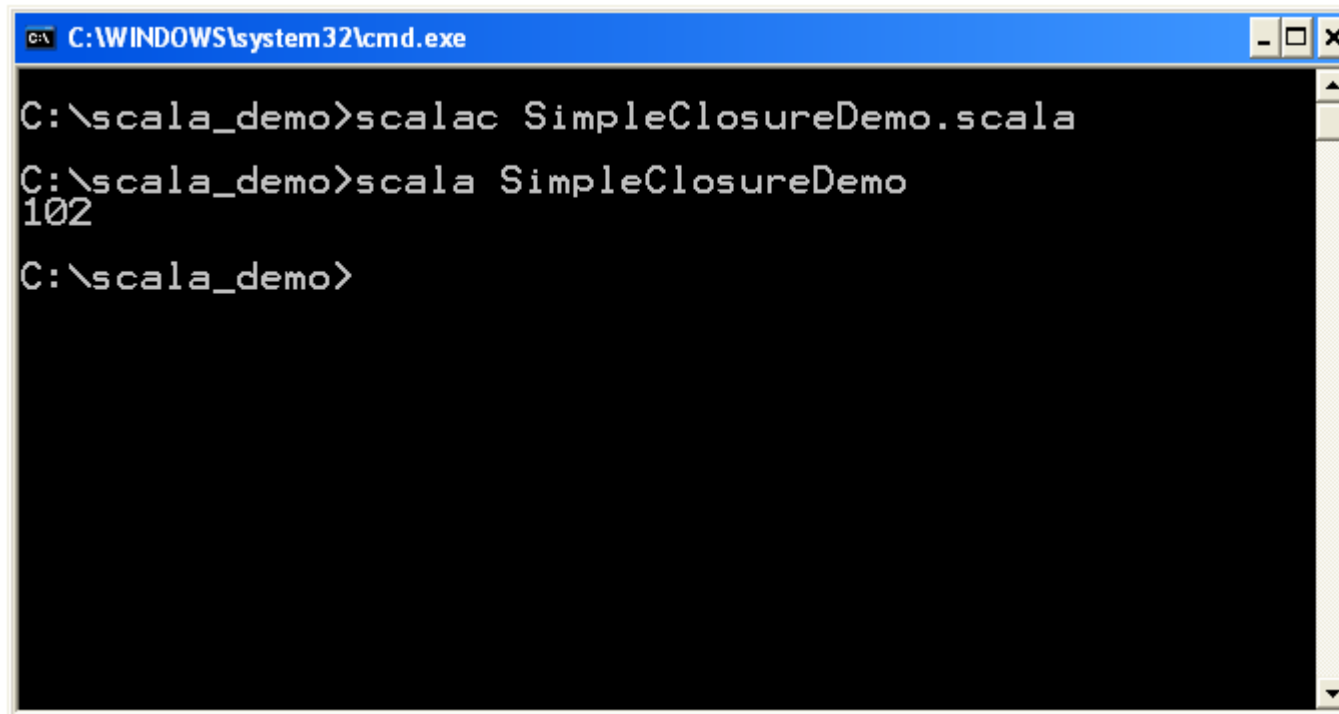
- ❖ The closure is an inner function that has access to its outer function scope. When defining a function, at runtime we get an object.
- ❖ Each function call is actually an invocation of the object that represents the function.
- ❖ When we define a function that uses variables from its outer scope the object we get is a closure.

What is a Closure?



```
SimpleClosureDemo - Notepad
File Edit Format View Help
object SimpleClosureDemo
{
    def main(args: Array[String]):Unit =
    {
        var total:Int = 100
        var calc = (num:Int) => num + total
        println(calc(2))
    }
}
```

What is a Closure?

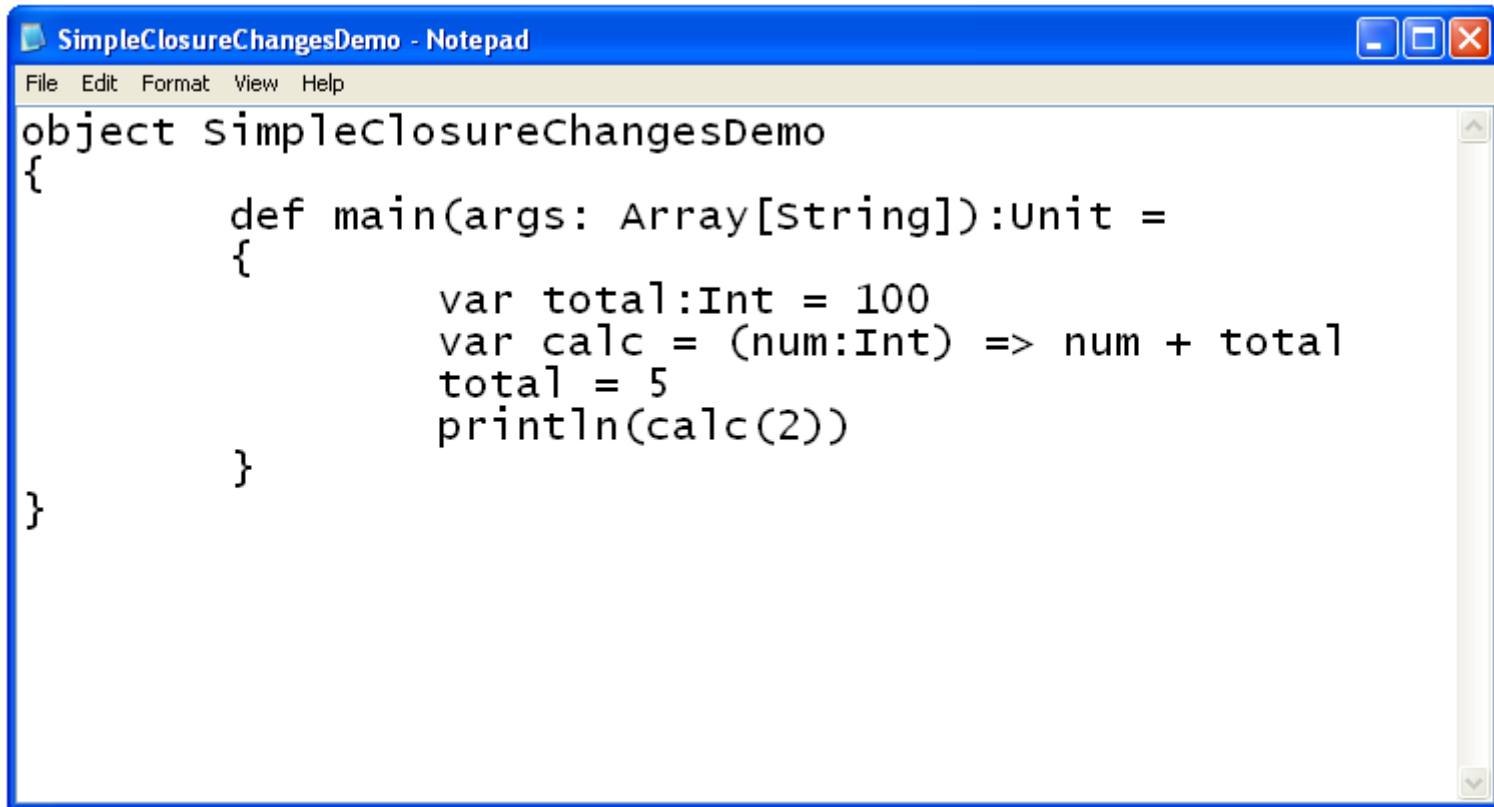


```
C:\WINDOWS\system32\cmd.exe
C:\scala_demo>scalac SimpleClosureDemo.scala
C:\scala_demo>scala SimpleClosureDemo
102
C:\scala_demo>
```

Changes

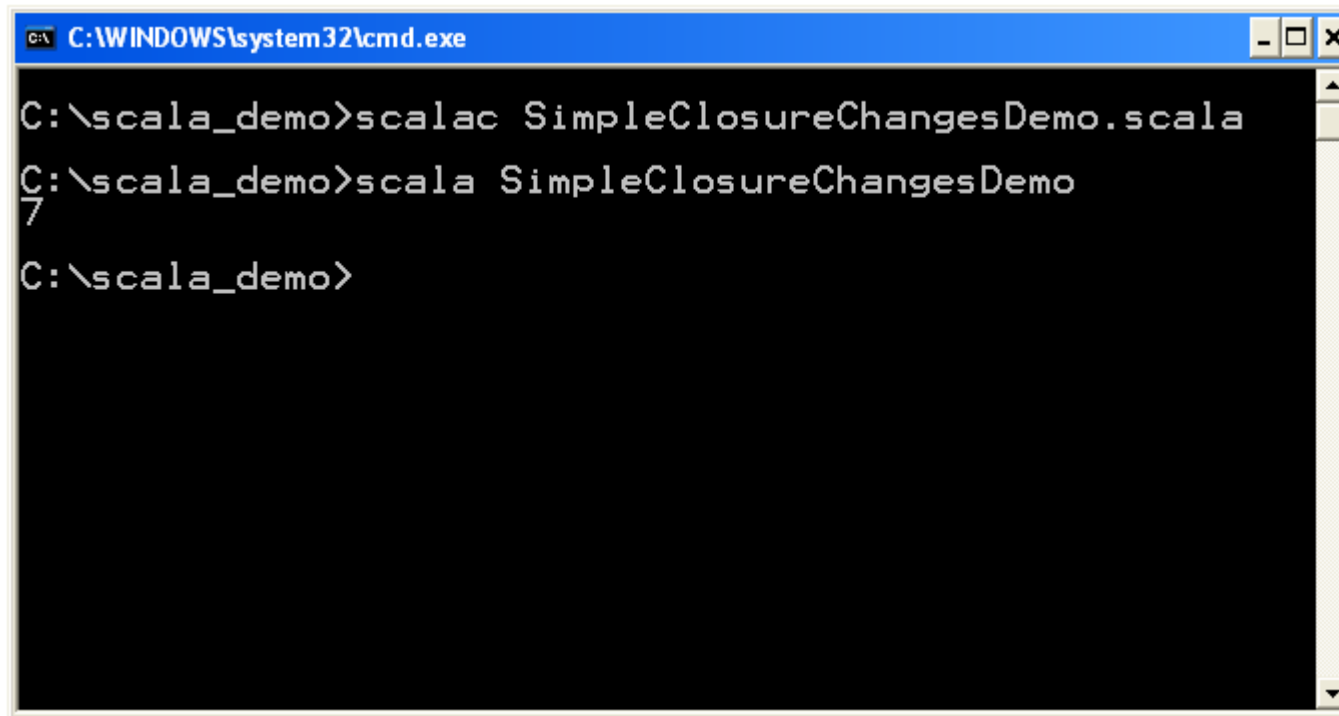
- ❖ Once a closure was created, the closure continues to see changes in those variables the closure uses from its outer scope.
- ❖ The closures in Scala capture the variables themselves. They don't capture their values.

Changes



```
SimpleClosureChangesDemo - Notepad
File Edit Format View Help
object SimpleClosureChangesDemo
{
    def main(args: Array[String]):Unit =
    {
        var total:Int = 100
        var calc = (num:Int) => num + total
        total = 5
        println(calc(2))
    }
}
```

Changes

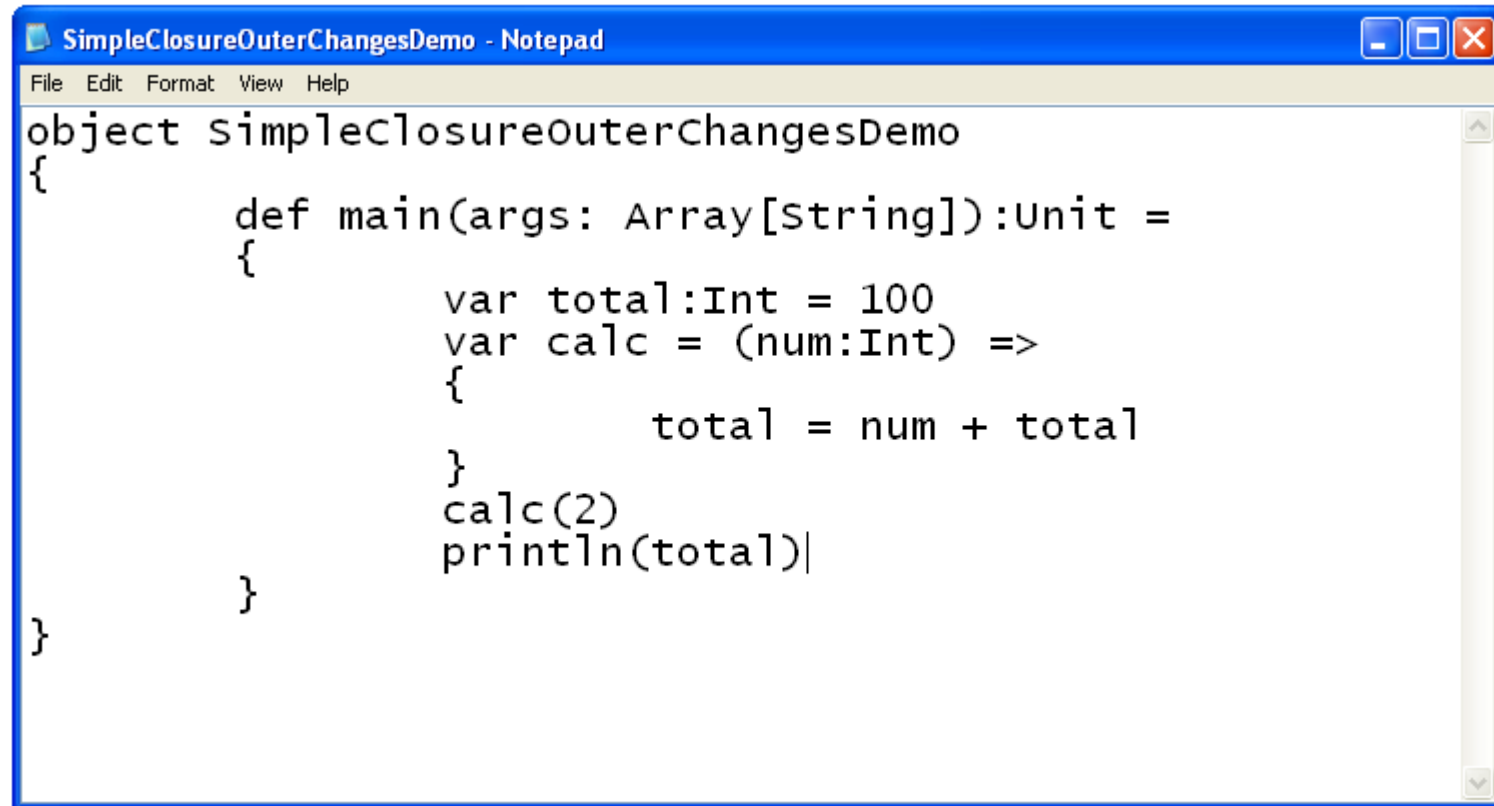


```
C:\WINDOWS\system32\cmd.exe
C:\scala_demo>scalac SimpleClosureChangesDemo.scala
C:\scala_demo>scala SimpleClosureChangesDemo
7
C:\scala_demo>
```


Changes

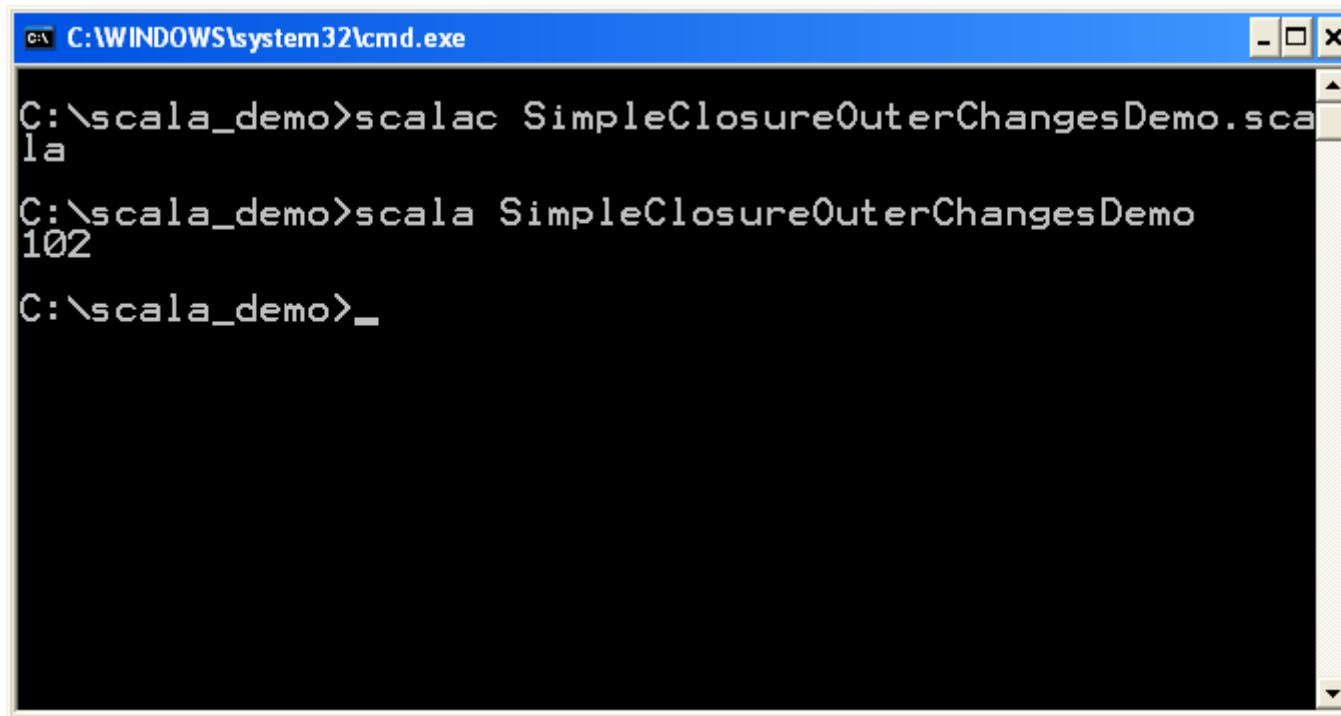
- ❖ Changes made by a closure to a captured variable are visible outside the closure.

Changes



```
SimpleClosureOuterChangesDemo - Notepad
File Edit Format View Help
object simpleClosureOuterChangesDemo
{
    def main(args: Array[String]):Unit =
    {
        var total:Int = 100
        var calc = (num:Int) =>
        {
            total = num + total
        }
        calc(2)
        println(total)
    }
}
```

Changes



```
C:\WINDOWS\system32\cmd.exe
C:\scala_demo>scalac SimpleClosureOuterChangesDemo.scala
C:\scala_demo>scala SimpleClosureOuterChangesDemo
102
C:\scala_demo>_
```