Singleton Objects: Standalone

In this lesson, you will learn how to write a singleton standalone object.

We'll cover the following

Standalone Objects

Standalone objects are singleton objects which are not attached to any class, i.e., they don't have the same name as any class. Hence, they can literally *stand alone*.

Up until now, we haven't been able to execute our `ChecksumAccumulator` program. This is where standalone objects come in. They can be used for defining the entry point of a Scala application, an executable program. To use a standalone object for this purpose, you need to create a `main` method that takes a single parameter of type `Array[String]` (an array of strings). Now all you have to do when you want to execute your program is run the standalone object.

Let's define a standalone object for our `ChecksumAccumulator` program so it can be used as an application.

```
import scala.collection.mutable

class ChecksumAccumulator {
  private var sum = 0
  def add(b: Byte) = sum += b
  def checksum() = ~(sum & 0xFF) + 1
}

//companion object of ChecksumAccumulator

object ChecksumAccumulator {
  private val cache = mutable.Map.empty[String, Int]
  def calculate(s: String): Int =
    if (cache.contains(s))
      cache(s)
    else {
      val acc = new ChecksumAccumulator
```

```
      for (c <- s)
        acc.add(c.toByte)
      val cs = acc.checksum()
      cache += (s -> cs)
      cs
    }
}

//standalone object that acts as an entry point for the ChecksumAccumulator application

import ChecksumAccumulator.calculate

object EntryApplication {
  def main(args: Array[String]) = {
    for(arg<-args)
      println(arg + ": " + calculate(arg))
  }
}
```

On **line 28**, of the code widget above, we are importing the `calculate` method of the `ChecksumAccumulator` program. Singleton objects can be used anywhere in the program by using `import` followed by the name of the object and the member you wish to use.

On **line 30**, we start defining our standalone object, `EntryApplication`, which has a single member, a `main` method. `main` is taking a single parameter `args` of type `Array[String]`. It takes each string in the array and prints its checksum by passing it to the `calculate` method.

Let's run the program and see what happens.

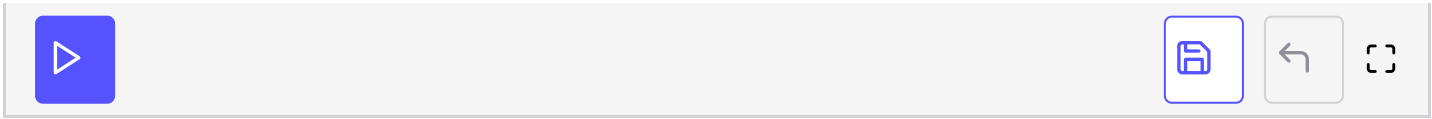
> The code below will give a warning...ignore it for now.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
import scala.collection.mutable
import ChecksumAccumulator.calculate

class ChecksumAccumulator {
  private var sum = 0
  def add(b: Byte) = sum += b
  def checksum() = ~(sum & 0xFF) + 1
}

//companion object of ChecksumAccumulator
object ChecksumAccumulator {
  private val cache = mutable.Map.empty[String, Int]
  def calculate(s: String): Int =
    if (cache.contains(s))
```

```
    if (cache.contains(s))
      cache(s)
    else {
      val acc = new ChecksumAccumulator
      for (c <- s)
        acc.add(c.toByte)
      val cs = acc.checksum()
      cache += (s -> cs)
      cs
    }
}

//standalone object that acts as an entry point for the ChecksumAccumulator application
object EntryApplication {
  def main(args: Array[String]) = {
    for(arg<-args)
      println(arg + ": " + calculate(arg))
  }
}

val inputArray = Array("insert", "anything", "here")
EntryApplication.main(inputArray)
```



The output of the above code gives us the checksum of **insert**, **anything**, and **here**.

Try running `EntryApplication` with your own array and see how the output changes.

Warning

While the above code gives the correct output, it also gives a warning. This is because we are calling the `main` method. A standalone object which acts as an application should not be called using `EntryApplication.main(inputArray)`. This is because, as in most programming languages, `main` does not need to be called as it is automatically called when running the script it is contained in.

Suppose your `ChecksumAccumulator` class and object are written in a script named **main.scala** and `EntryApplication` is written in a separate script with the name **test.scala**. Traditionally, you would run the Scala script using the following command.

```
scalac main.scala test.scala
```

`scalac` is a Scala compiler. You use this to run `main.scala` and `test.scala`. Running the command above will result in `.class` files for each class and singleton object in the given `.scala` files.

After this, all you would have to do is write the string whose checksum you want to be calculated on the command line. The string should be preceded by the standalone object which calls the `main` method. In our case, that object is `EntryApplication`.

```
scala EntryApplication insert anything here
```

The files you would run, would have the following structure.

main.scala test.scala	All code files are copied to end of the page...

Conclusion

The purpose of this lesson, along with the previous two lessons, was to give you a taste of the structure of larger Scala programs and how they work; how Scala provides singleton objects to make our lives easier when writing lengthy programs.

With standalone objects, our discussion on objects and classes comes to an end. This is only the tip of the iceberg when it comes to object-oriented programming in Scala. Let's wrap up this chapter with a quiz to test what you have learned so far.

Code Files Content !!!

⤋ ⤋

```
------------------------------------------------------------------------
|  main.scala [1]
------------------------------------------------------------------------


import scala.collection.mutable
```

```
class ChecksumAccumulator {
  private var sum = 0
  def add(b: Byte) = sum += b
  def checksum() = ~(sum & 0xFF) + 1
}

// Companion object of ChecksumAccumulator
object ChecksumAccumulator {
  private val cache = mutable.Map.empty[String, Int]
  def calculate(s: String): Int =
    if (cache.contains(s))
      cache(s)
    else {
      val acc = new ChecksumAccumulator
      for (c <- s)
        acc.add(c.toByte)
      val cs = acc.checksum()
      cache += (s -> cs)
      cs
    }
}


-------------------------------------------------------------------------
|  test.scala [1]
-------------------------------------------------------------------------


import ChecksumAccumulator.calculate

//standalone object that acts as an entry point for the ChecksumAccumulator application
object EntryApplication {
  def main(args: Array[String]) = {
    for(arg<-args)
      println(arg + ": " + calculate(arg))
  }
}




*************************************************************************
```