Aleksandar Karamfilov

CTO @ Pragmatic.bg

Scala awesomeness for the QA fellow

Handy Scala language features

JVM Languages

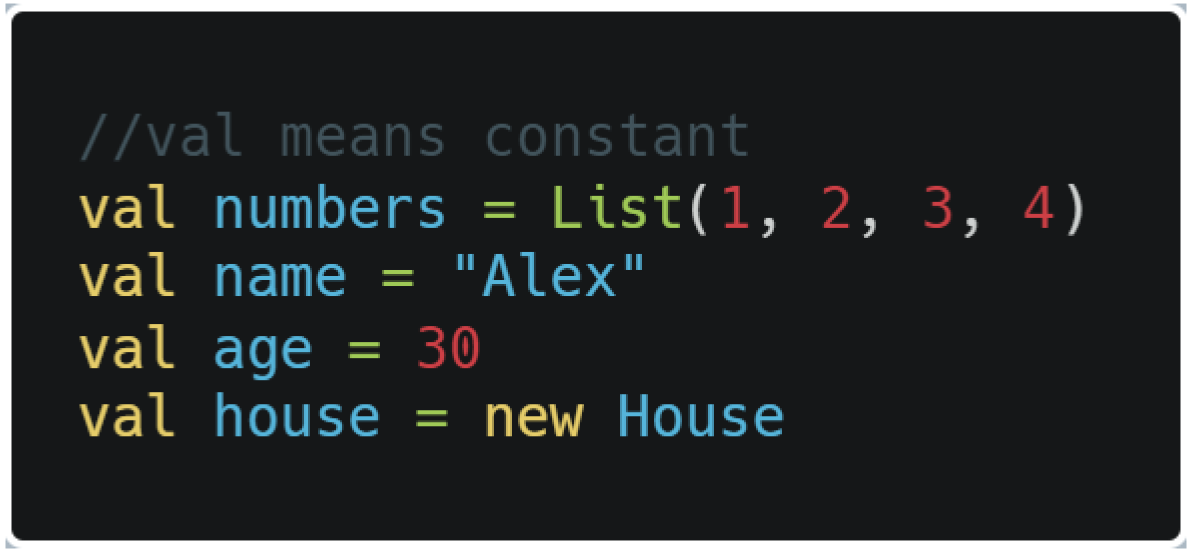
- Java
- Scala
- Kotlin
- Groovy
- Clojure
- Jython
- …

Scala language

Scala stands for “scalable language”.

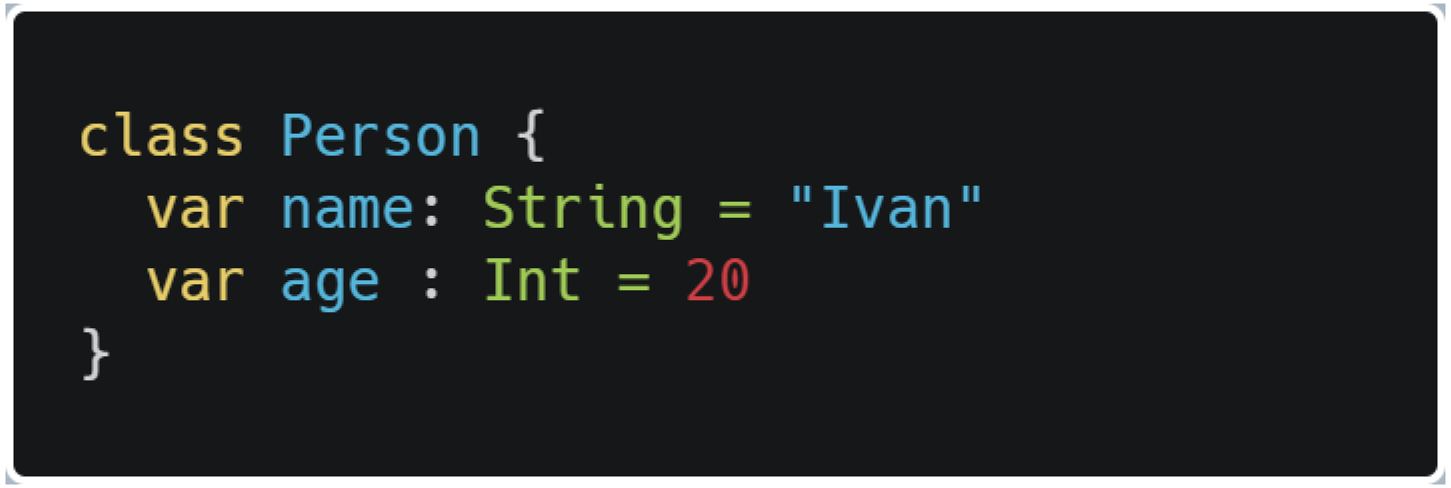
Scala's a statically typed language which combines two important programming paradigms, namely object-oriented and functional programming.

Embrace the immutability (val)



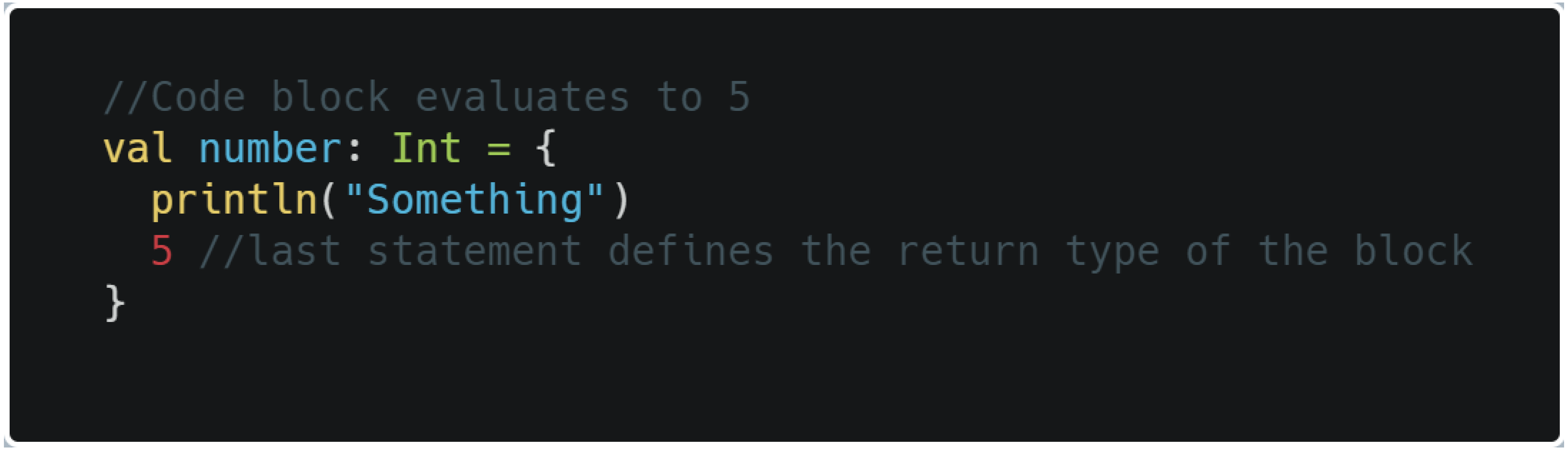
```
//val means constant
val numbers = List(1, 2, 3, 4)
val name = "Alex"
val age = 30
val house = new House
```

!Embrace the immutability (var)



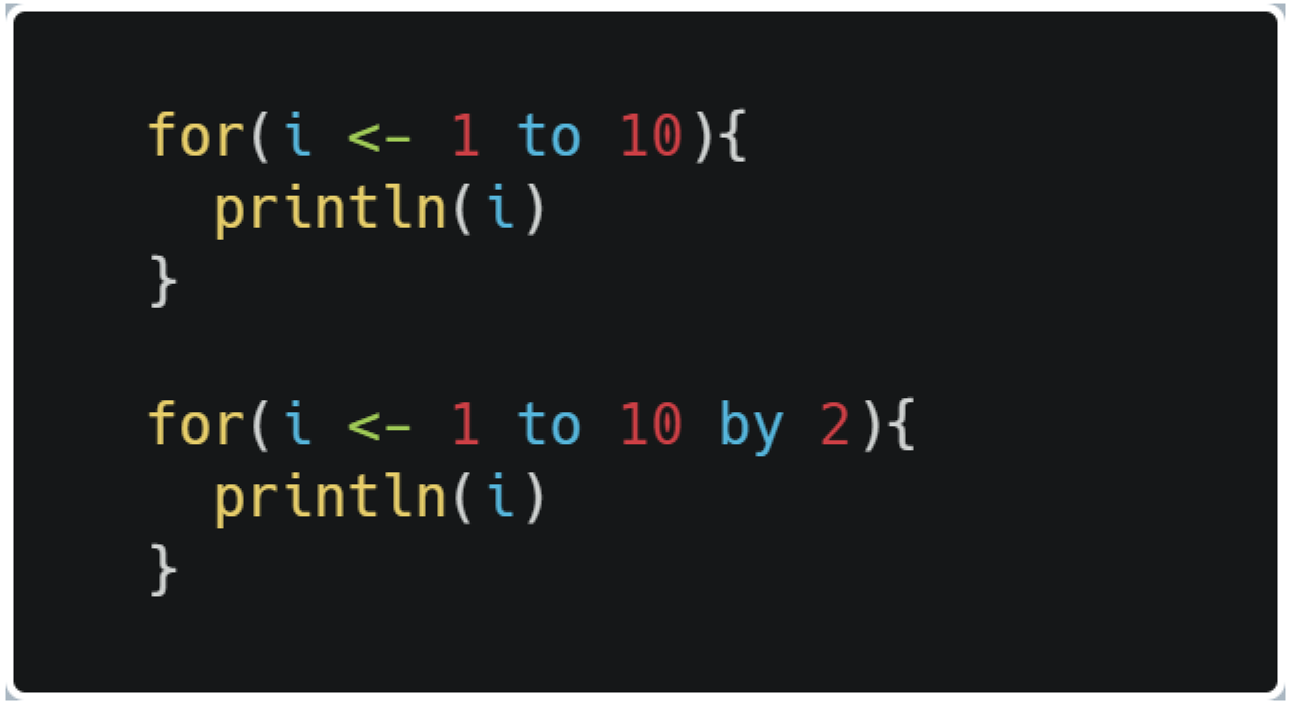
```
class Person {
  var name: String = "Ivan"
  var age : Int = 20
}
```

Code blocks

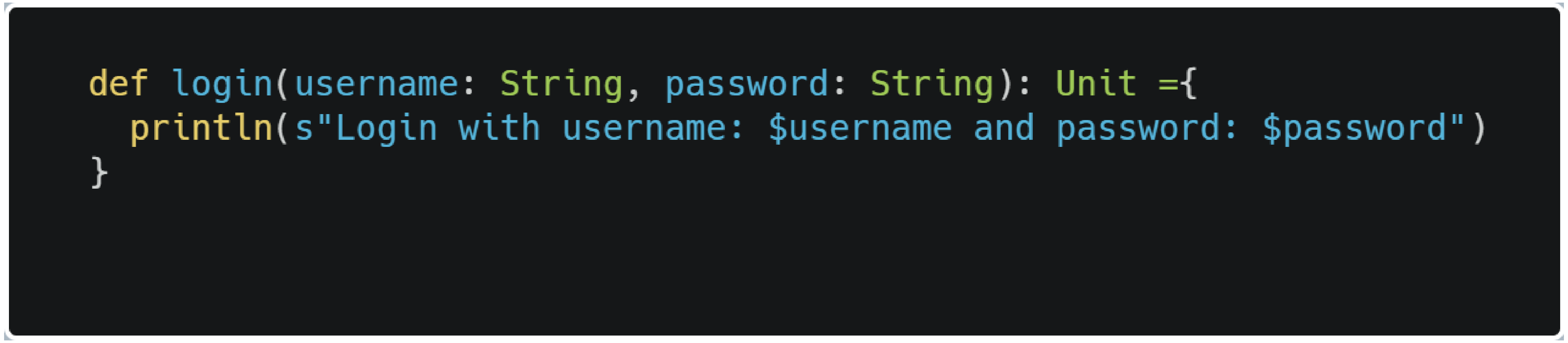


```kotlin
//Code block evaluates to 5
val number: Int = {
  println("Something")
  5 //last statement defines the return type of the block
}
```

For loop

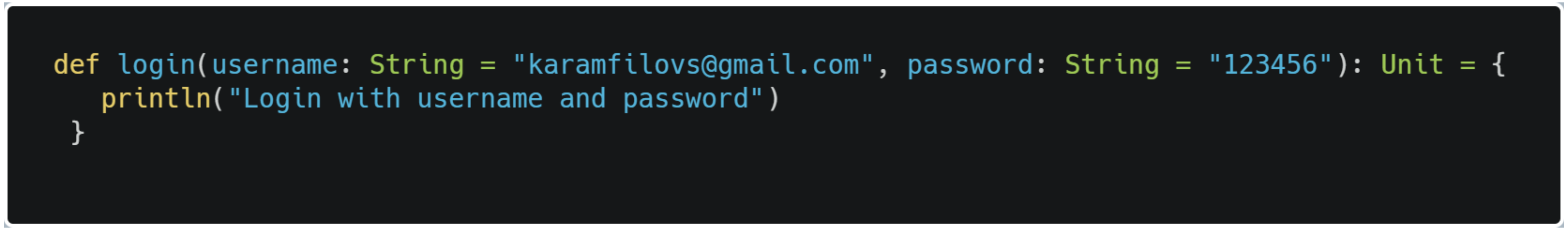

```
for(i <- 1 to 10){
  println(i)
}

for(i <- 1 to 10 by 2){
  println(i)
}
```

Methods in Scala (def)



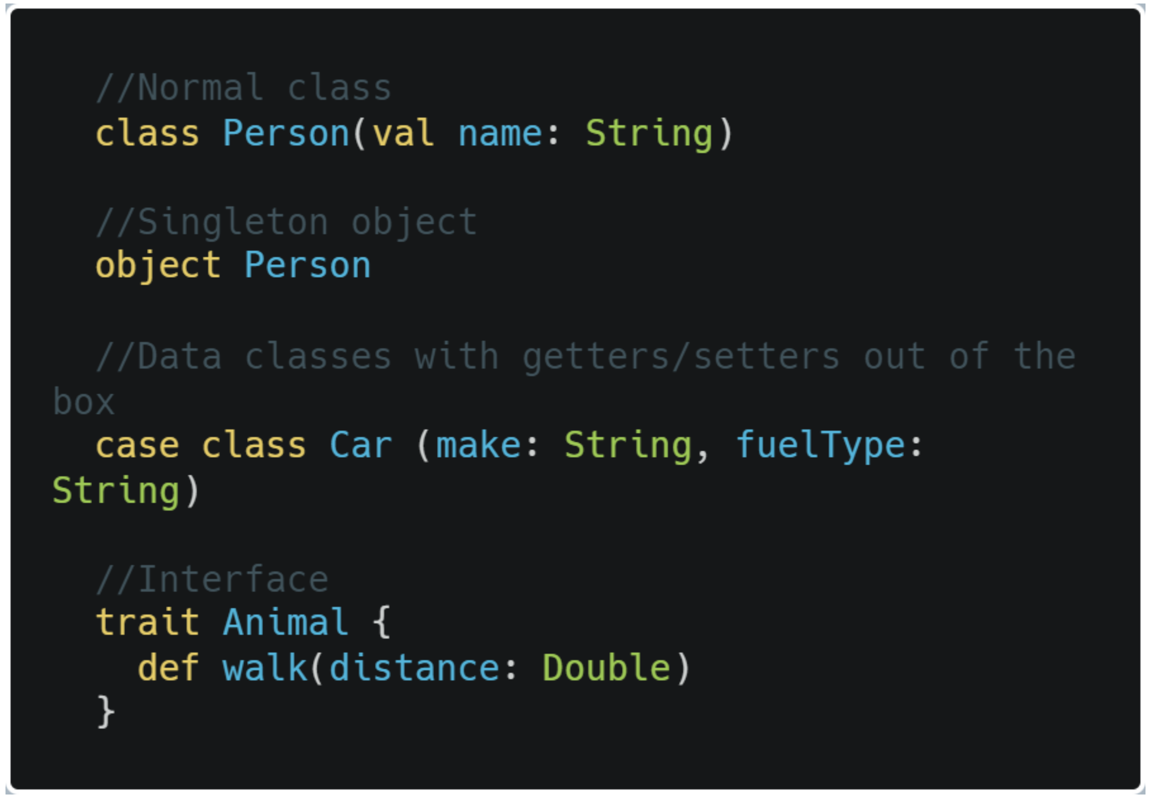
```scala
def login(username: String, password: String): Unit ={
  println(s"Login with username: $username and password: $password")
}
```

Methods in Scala (default args)



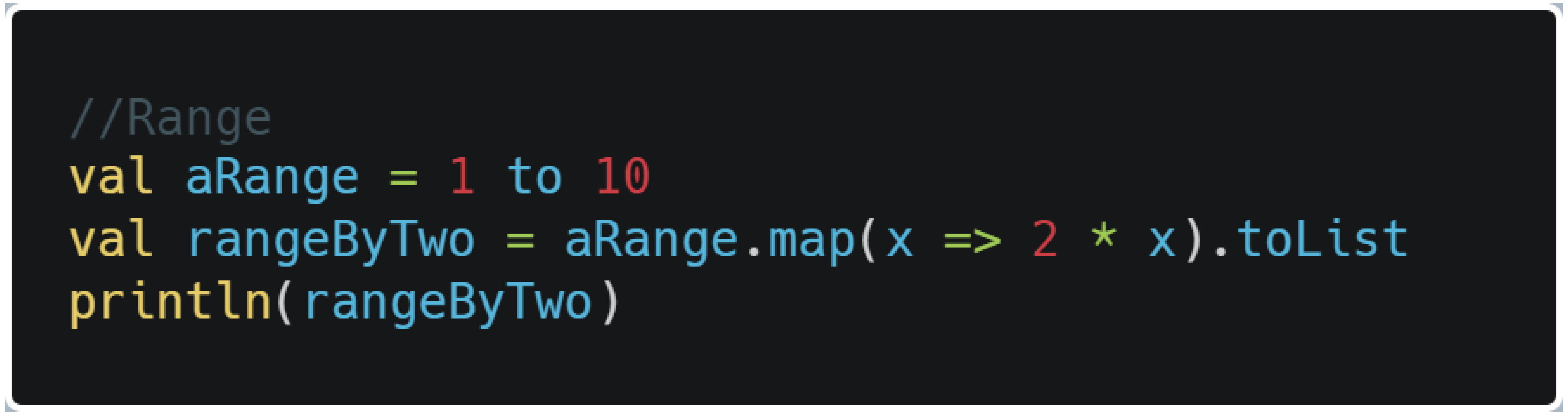
```scala
def login(username: String = "karamfilovs@gmail.com", password: String = "123456"): Unit = {
   println("Login with username and password")
 }
```

Classes, Objects, Traits



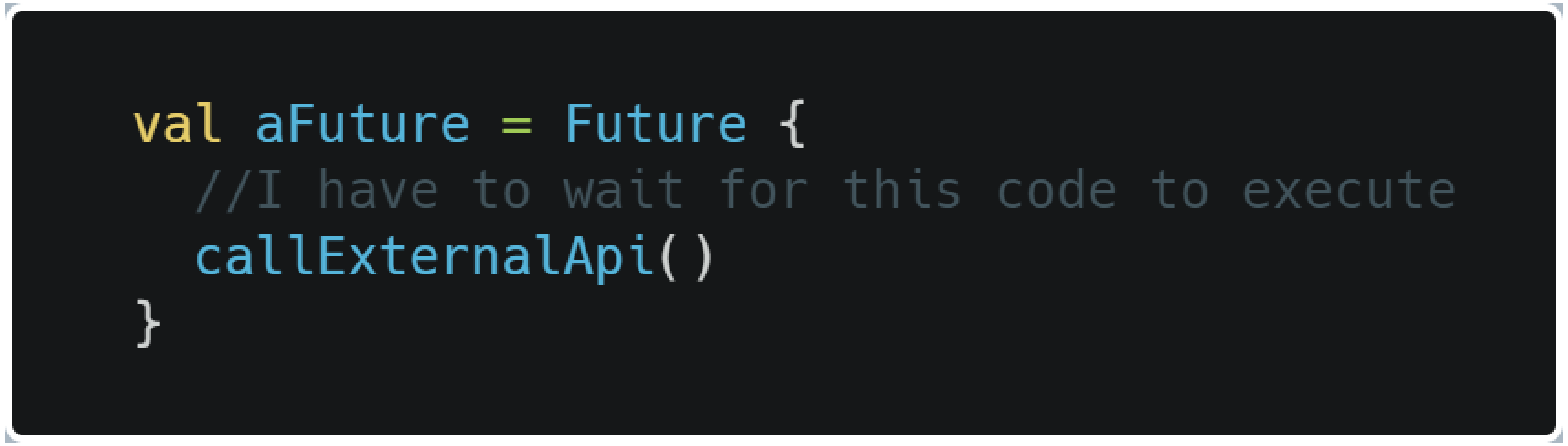
```
  //Normal class
  class Person(val name: String)

  //Singleton object
  object Person

  //Data classes with getters/setters out of the
box
  case class Car (make: String, fuelType:
String)

  //Interface
  trait Animal {
    def walk(distance: Double)
  }
```

Range



```
//Range
val aRange = 1 to 10
val rangeByTwo = aRange.map(x => 2 * x).toList
println(rangeByTwo)
```

Future

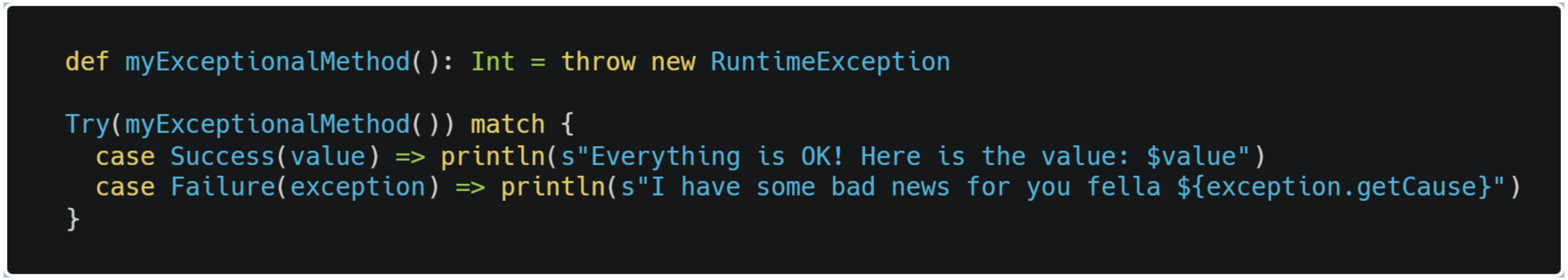


```
val aFuture = Future {
  //I have to wait for this code to execute
  callExternalApi()
}
```

Lazy

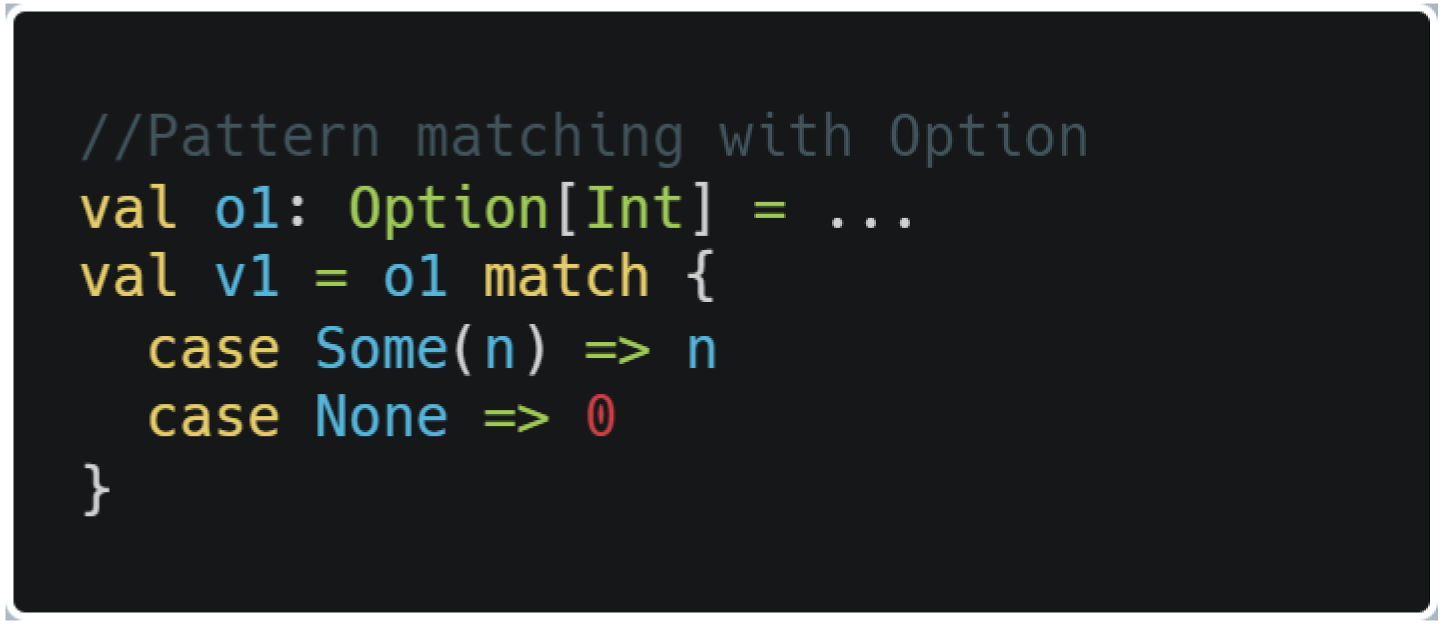
```scala
//The values wont be initialized until the first use
lazy val clientAPI = new ClientAPI
lazy val courseAPI = new CourseAPI
lazy val bankAccountAPI = new BankAccountAPI
```

Try [Pattern matching]



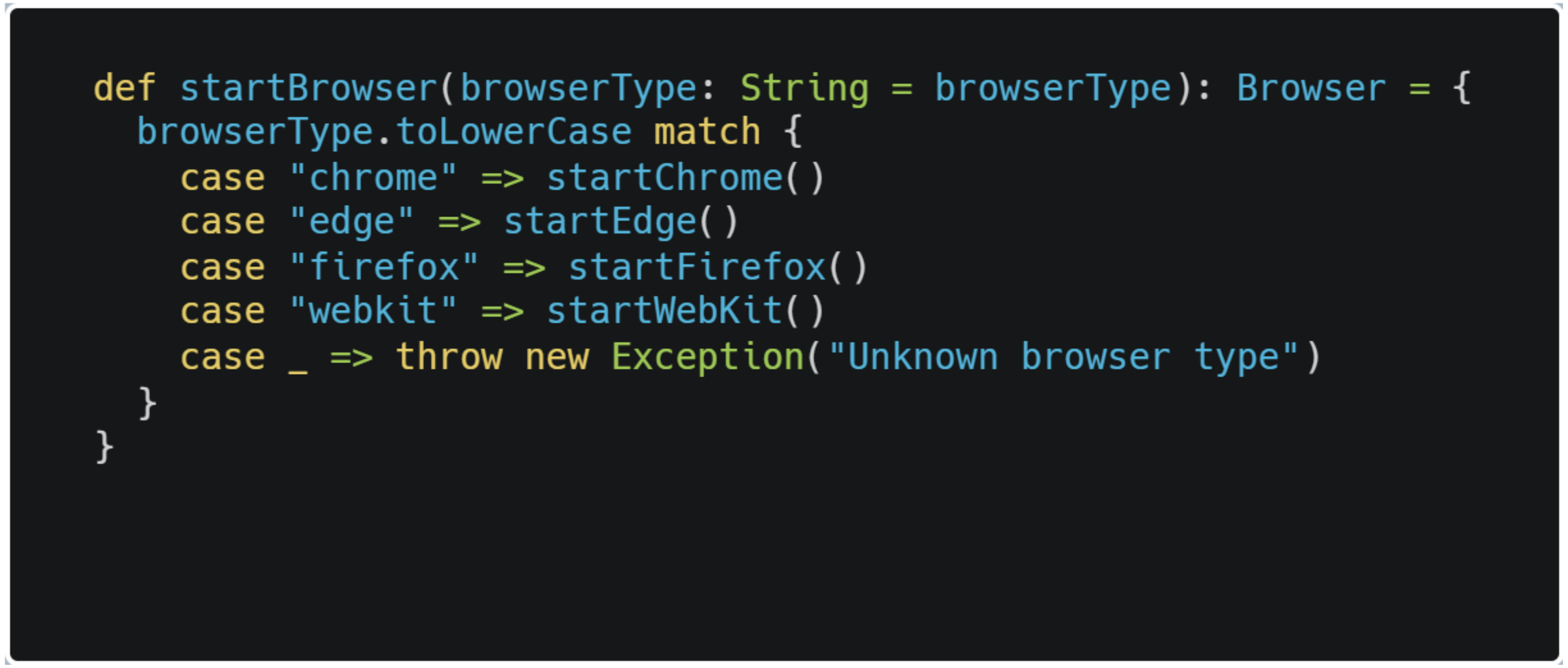
```scala
def myExceptionalMethod(): Int = throw new RuntimeException

Try(myExceptionalMethod()) match {
  case Success(value) => println(s"Everything is OK! Here is the value: $value")
  case Failure(exception) => println(s"I have some bad news for you fella ${exception.getCause}")
}
```

Option [Pattern matching]



```
//Pattern matching with Option
val o1: Option[Int] = ...
val v1 = o1 match {
  case Some(n) => n
  case None => 0
}
```

Pattern matching by value



```
def startBrowser(browserType: String = browserType): Browser = {
  browserType.toLowerCase match {
    case "chrome" => startChrome()
    case "edge" => startEdge()
    case "firefox" => startFirefox()
    case "webkit" => startWebKit()
    case _ => throw new Exception("Unknown browser type")
  }
}
```

Unpack case class with match

```
case class Person(firstName: String, lastName: String)
private val alex = Person("Alex", "Karamfilov")

val fullName = alex match {
    case Person(fn, ln) => s"$fn $ln"
    case _ => ""
}
```

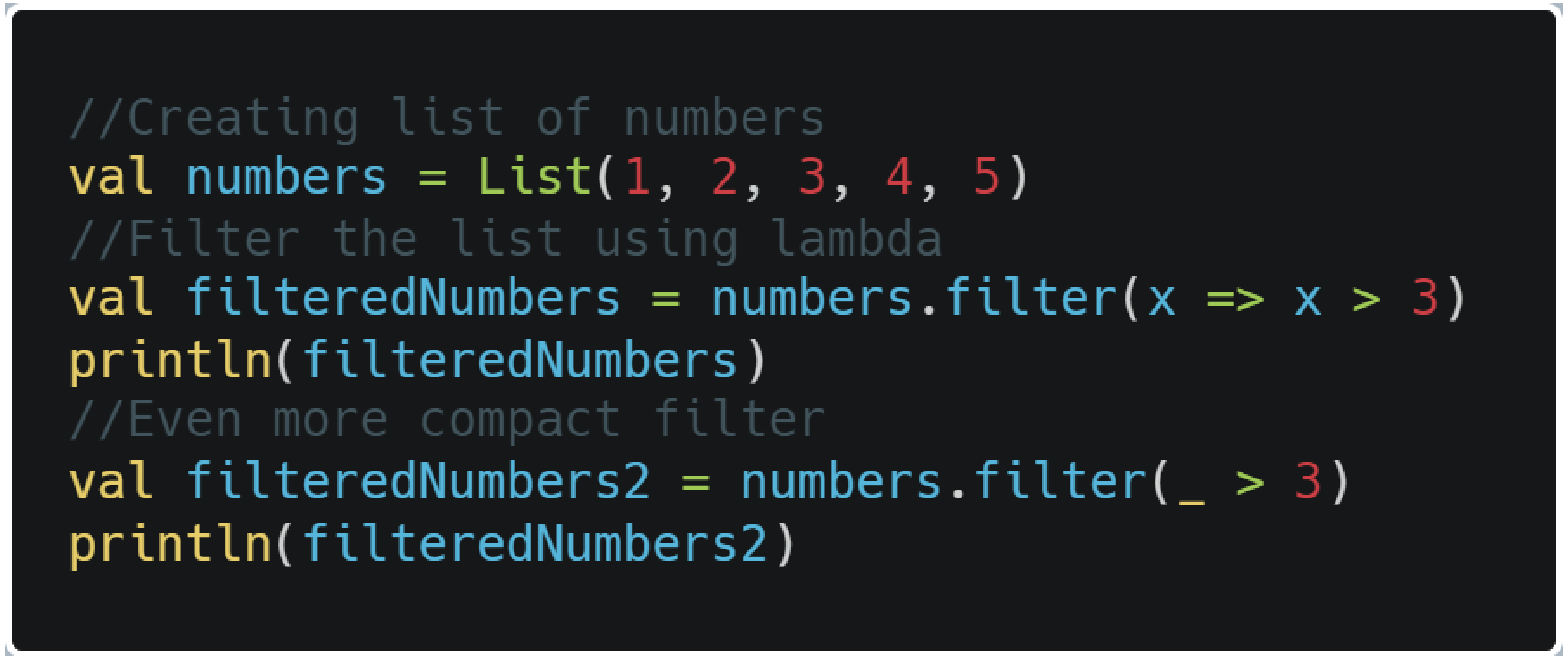
Apply (black magic)

```scala
class Person (name: String) {
  def apply (age: Int): Unit = println(s"I have aged with $age")
}

val bob = new Person("Bob")
bob.apply(42)
bob(43)
```

For comprehension

```
val listOfPassedAssertsInSucceededTests: List[Int] =
  for {
    result <- results
    if result.succeeded
  } yield (result.successfulAsserts)
val passedAssertsInSucceededTests: Int = listOfPassedAssertsInSucceededTests.sum
```

Filters



```
//Creating list of numbers
val numbers = List(1, 2, 3, 4, 5)
//Filter the list using lambda
val filteredNumbers = numbers.filter(x => x > 3)
println(filteredNumbers)
//Even more compact filter
val filteredNumbers2 = numbers.filter(_ > 3)
println(filteredNumbers2)
```

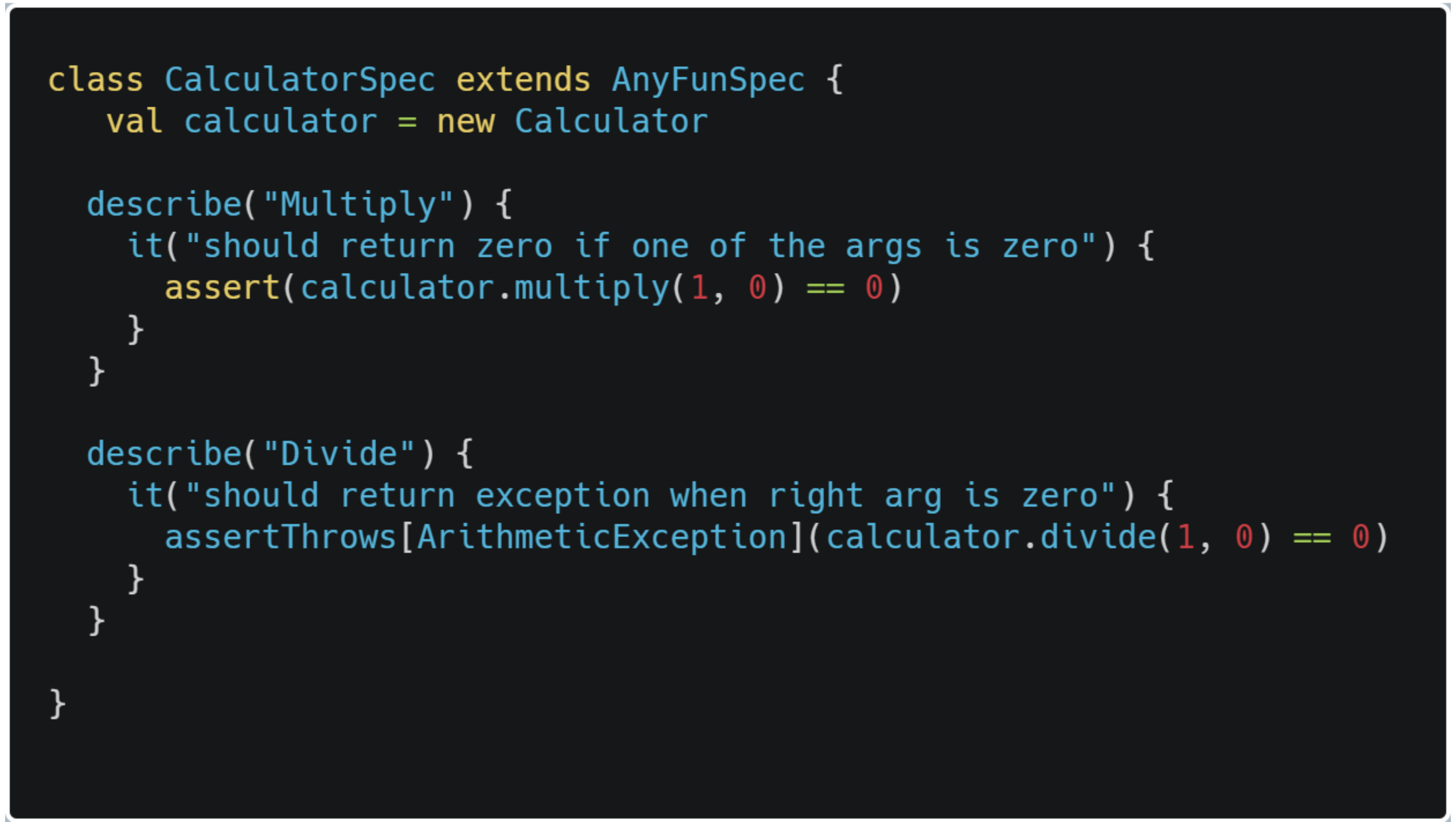
Test styles [PlaySpec]

```
"Item" should {
  "be created successfully with only mandatory fields" in {
    val coffee = Item("Coffee", 10, "кг")
    val response = api.itemAPI createItem coffee
    response.status mustBe 200
    response.body must include("Артикула е създаден успешно!")
  }
}
```

Test styles [AnyWordSpec]

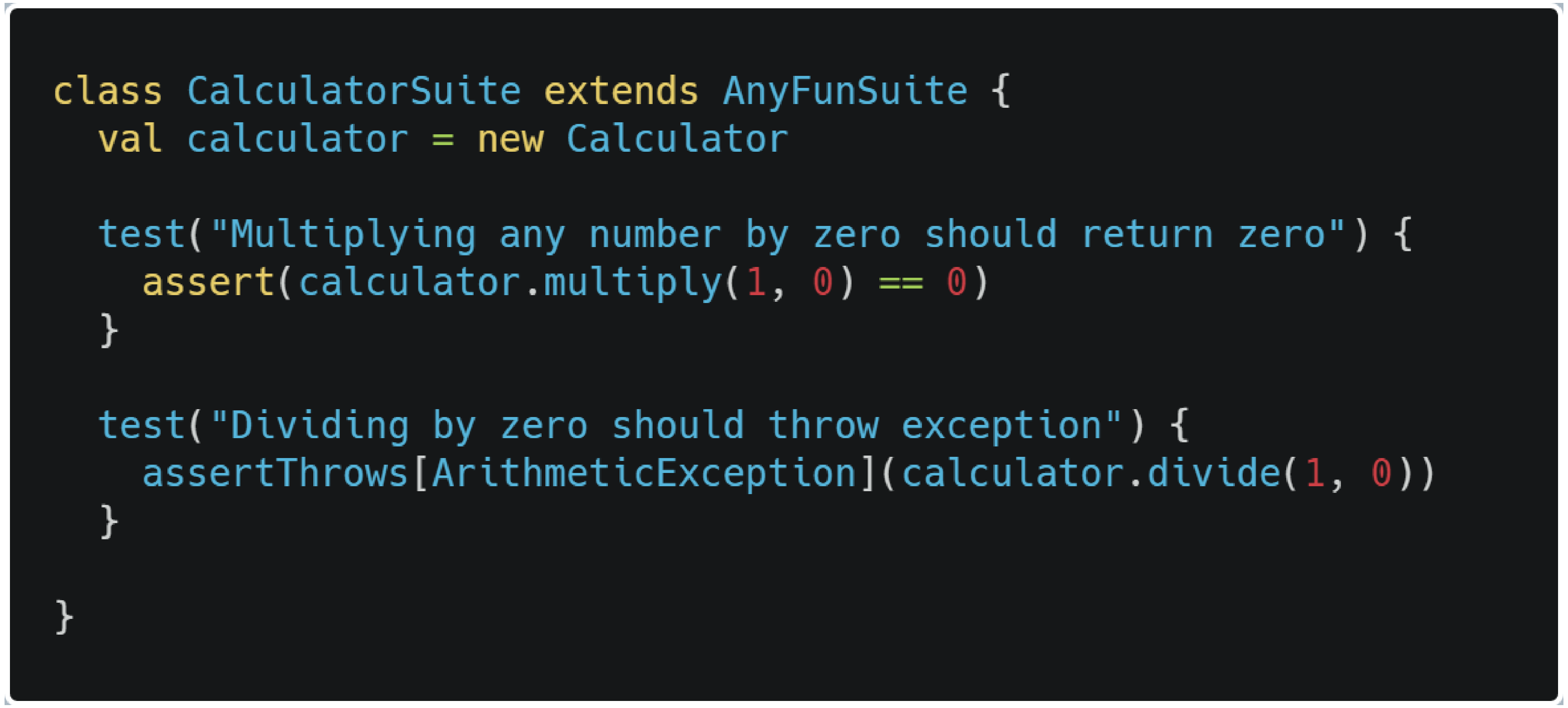
```scala
class CoursesSpec extends AnyWordSpec with should.Matchers {

  "Course" should {
    "be added for valid name" in {
      val courseResp = Apis.courseAPI createCourse "Test Automation"
      assert(courseResp.equals("Test Automation"))
      courseResp shouldBe "Test Automation2" //With should matchers
    }
  }
```

Test styles [AnyFunSpec]



```
class CalculatorSpec extends AnyFunSpec {
   val calculator = new Calculator

  describe("Multiply") {
    it("should return zero if one of the args is zero") {
      assert(calculator.multiply(1, 0) == 0)
    }
  }

  describe("Divide") {
    it("should return exception when right arg is zero") {
      assertThrows[ArithmeticException](calculator.divide(1, 0) == 0)
    }
  }

}
```

Test styles [AnyFunSuite]

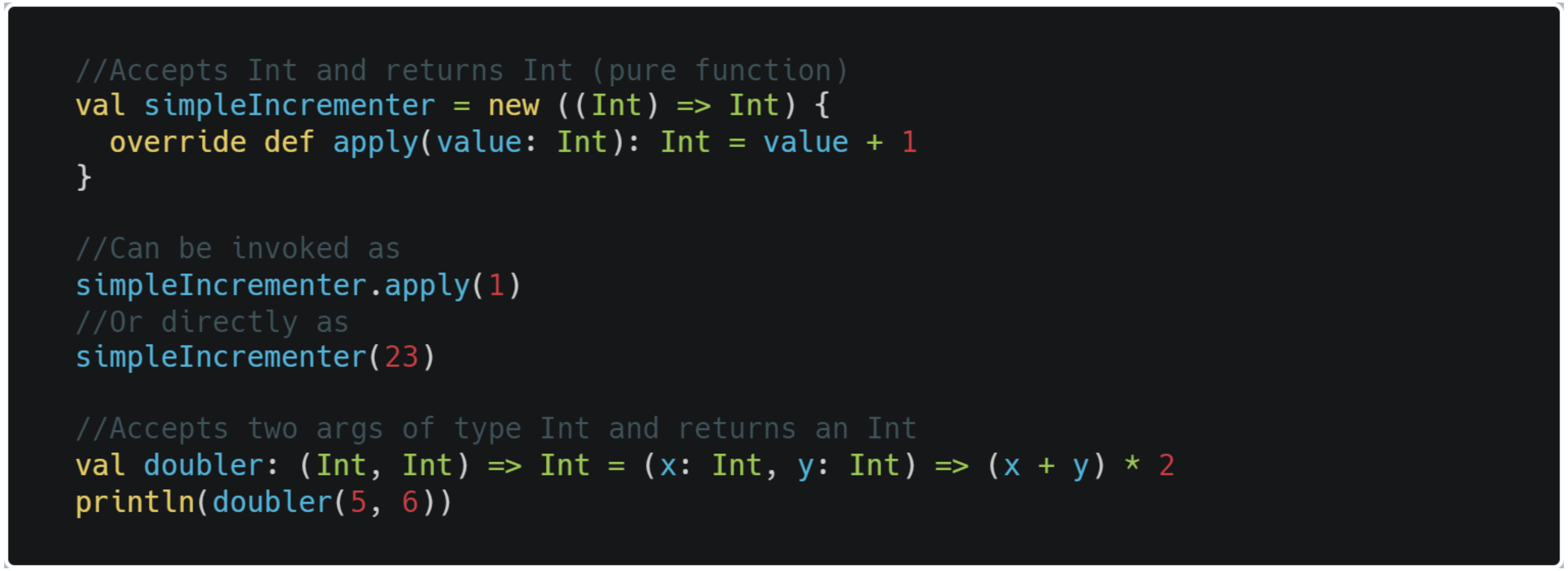


```
class CalculatorSuite extends AnyFunSuite {
  val calculator = new Calculator

  test("Multiplying any number by zero should return zero") {
    assert(calculator.multiply(1, 0) == 0)
  }

  test("Dividing by zero should throw exception") {
    assertThrows[ArithmeticException](calculator.divide(1, 0))
  }

}
```

Data Transfer Objects

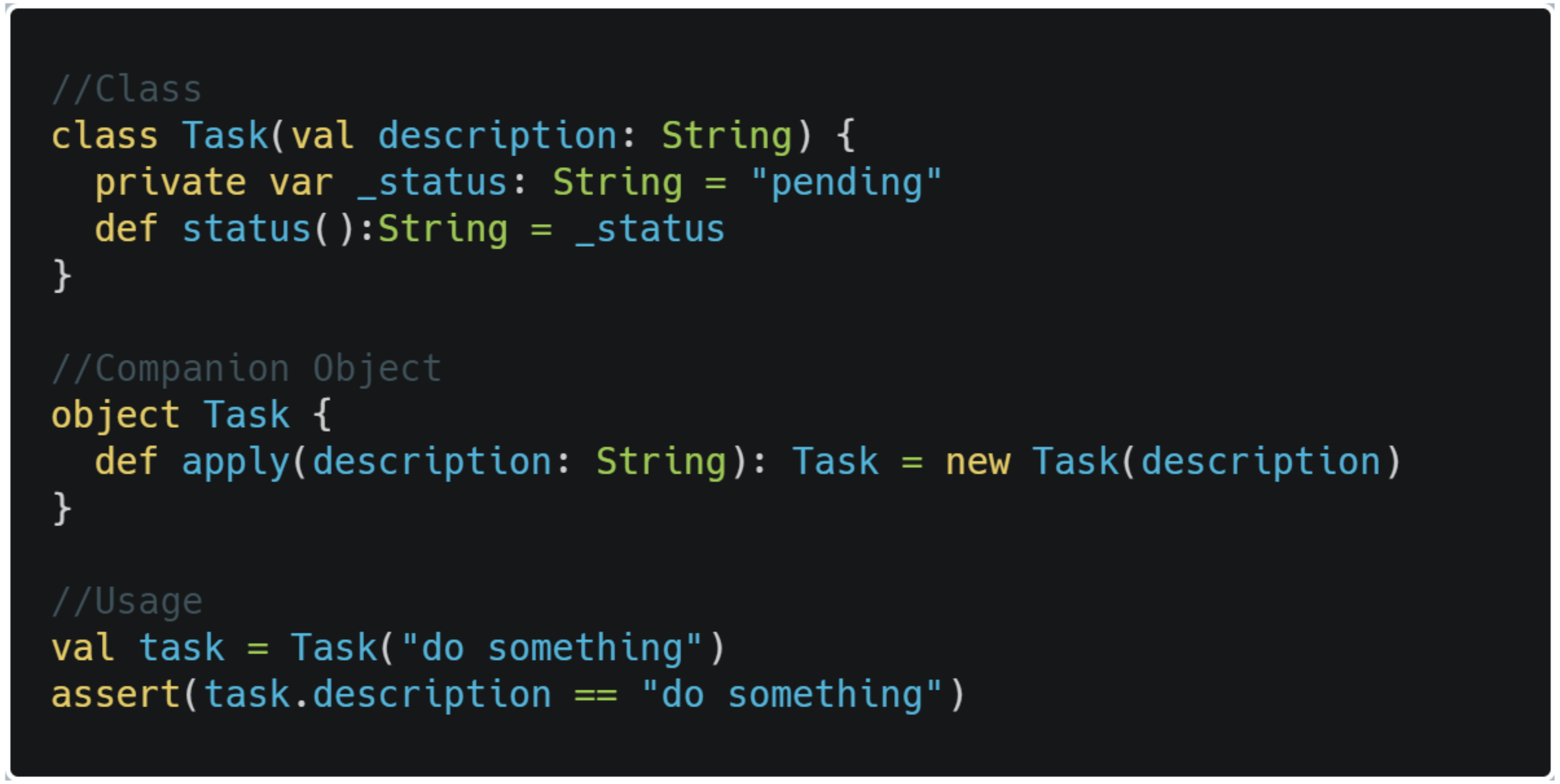
```
//Data Transfer Object (DTO)
case class Item(name: String, price_for_quantity: Int, quantity_unit: String)

//Deserialize from JSON to Item
Gson.fromJson(response.body, classOf[Item])

//POST request with generic type [Item]
protected def post[T](path: String, body: T): WSResponse = {
    val response = Await.result(baseRequest(path).post(Gson.toJson(body)), MaxWait)
    response
  }
```

Functional programming



```scala
//Accepts Int and returns Int (pure function)
val simpleIncrementer = new ((Int) => Int) {
  override def apply(value: Int): Int = value + 1
}

//Can be invoked as
simpleIncrementer.apply(1)
//Or directly as
simpleIncrementer(23)

//Accepts two args of type Int and returns an Int
val doubler: (Int, Int) => Int = (x: Int, y: Int) => (x + y) * 2
println(doubler(5, 6))
```

Companion objects



```scala
//Class
class Task(val description: String) {
  private var _status: String = "pending"
  def status():String = _status
}

//Companion Object
object Task {
  def apply(description: String): Task = new Task(description)
}

//Usage
val task = Task("do something")
assert(task.description == "do something")
```

API Testing

Testing API in Scala world

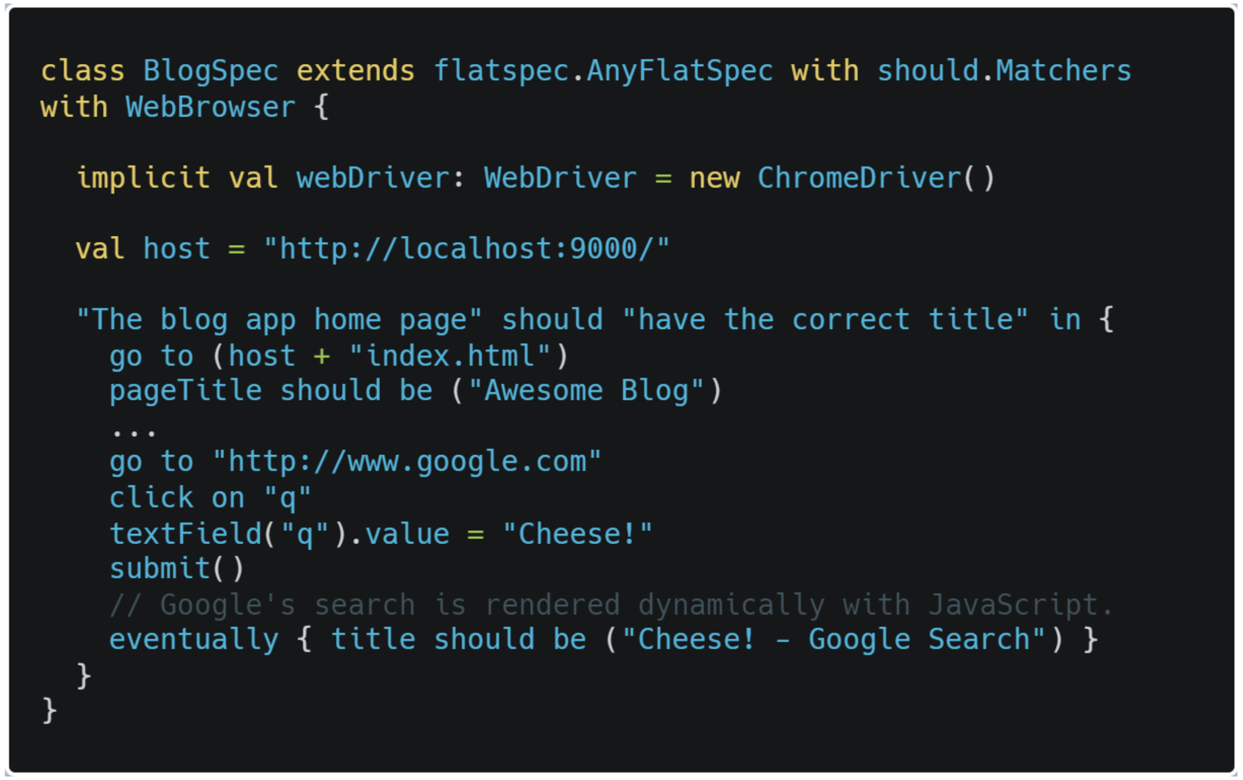
API tests readability [Play lib]

```
//Delete client
val response = clientAPI deleteClient id
response.status mustBe 200
response.body mustBe "{\"success\":{\"message\":\"Клиента е изтрит\"}}"
```

UI Testing

Testing web apps in Scala world

UI tests readability [Selenium]

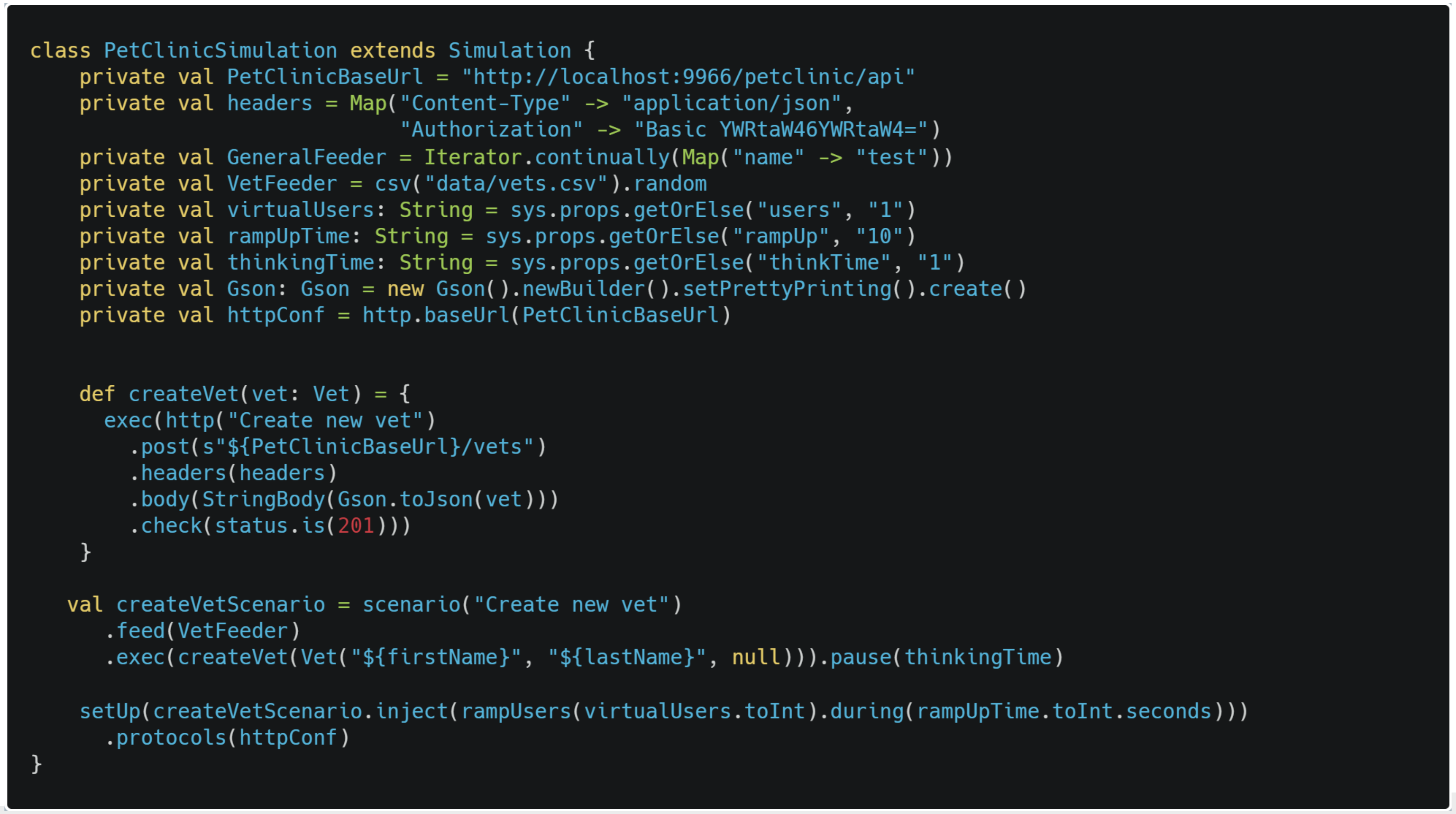


```
class BlogSpec extends flatspec.AnyFlatSpec with should.Matchers
with WebBrowser {

  implicit val webDriver: WebDriver = new ChromeDriver()

  val host = "http://localhost:9000/"

  "The blog app home page" should "have the correct title" in {
    go to (host + "index.html")
    pageTitle should be ("Awesome Blog")
    ...
    go to "http://www.google.com"
    click on "q"
    textField("q").value = "Cheese!"
    submit()
    // Google's search is rendered dynamically with JavaScript.
    eventually { title should be ("Cheese! - Google Search") }
  }
}
```

Load Testing

Loading web apps in Scala world

Load tests readability [Gatling]



```scala
class PetClinicSimulation extends Simulation {
    private val PetClinicBaseUrl = "http://localhost:9966/petclinic/api"
    private val headers = Map("Content-Type" -> "application/json",
                              "Authorization" -> "Basic YWRtaW46YWRtaW4=")
    private val GeneralFeeder = Iterator.continually(Map("name" -> "test"))
    private val VetFeeder = csv("data/vets.csv").random
    private val virtualUsers: String = sys.props.getOrElse("users", "1")
    private val rampUpTime: String = sys.props.getOrElse("rampUp", "10")
    private val thinkingTime: String = sys.props.getOrElse("thinkTime", "1")
    private val Gson: Gson = new Gson().newBuilder().setPrettyPrinting().create()
    private val httpConf = http.baseUrl(PetClinicBaseUrl)


    def createVet(vet: Vet) = {
      exec(http("Create new vet")
        .post(s"${PetClinicBaseUrl}/vets")
        .headers(headers)
        .body(StringBody(Gson.toJson(vet)))
        .check(status.is(201)))
    }

   val createVetScenario = scenario("Create new vet")
      .feed(VetFeeder)
      .exec(createVet(Vet("${firstName}", "${lastName}", null))).pause(thinkingTime)

    setUp(createVetScenario.inject(rampUsers(virtualUsers.toInt).during(rampUpTime.toInt.seconds)))
      .protocols(httpConf)
}
```

Popular Scala libraries

- Gatling
- Spark
- Akka
- Slick
- Spray
- … and all Java libs ☺

Talk is cheap ☺

Thank you!

Contacts:

in https://www.linkedin.com/in/akaramfilov

https://github.com/karamfilovs

СЛЕДВАЩО СЪБИТИЕ

Лектор

Дата

Език

Следете актуалните обяви за **Software QA** DEV.BG