Hello World

First steps & intro

If Scala was

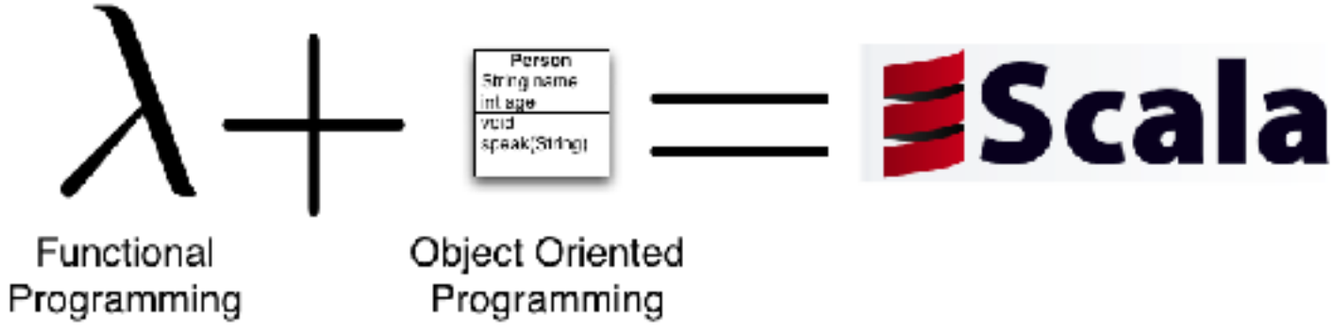
- what animal
- object oriented
- functional
- everything is an object

Hello, Scala!

Scala is a modern multi-paradigm programming lang ss
common programming patterns in a concise, eleg
It smoothly integrates features of object-oriented a .

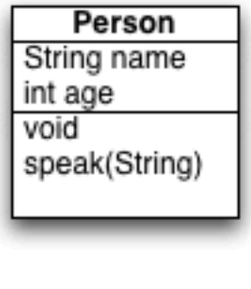
Why is Scala so cool?

- Object Oriented
- Functional Programming
- Statically typed
- JVM language



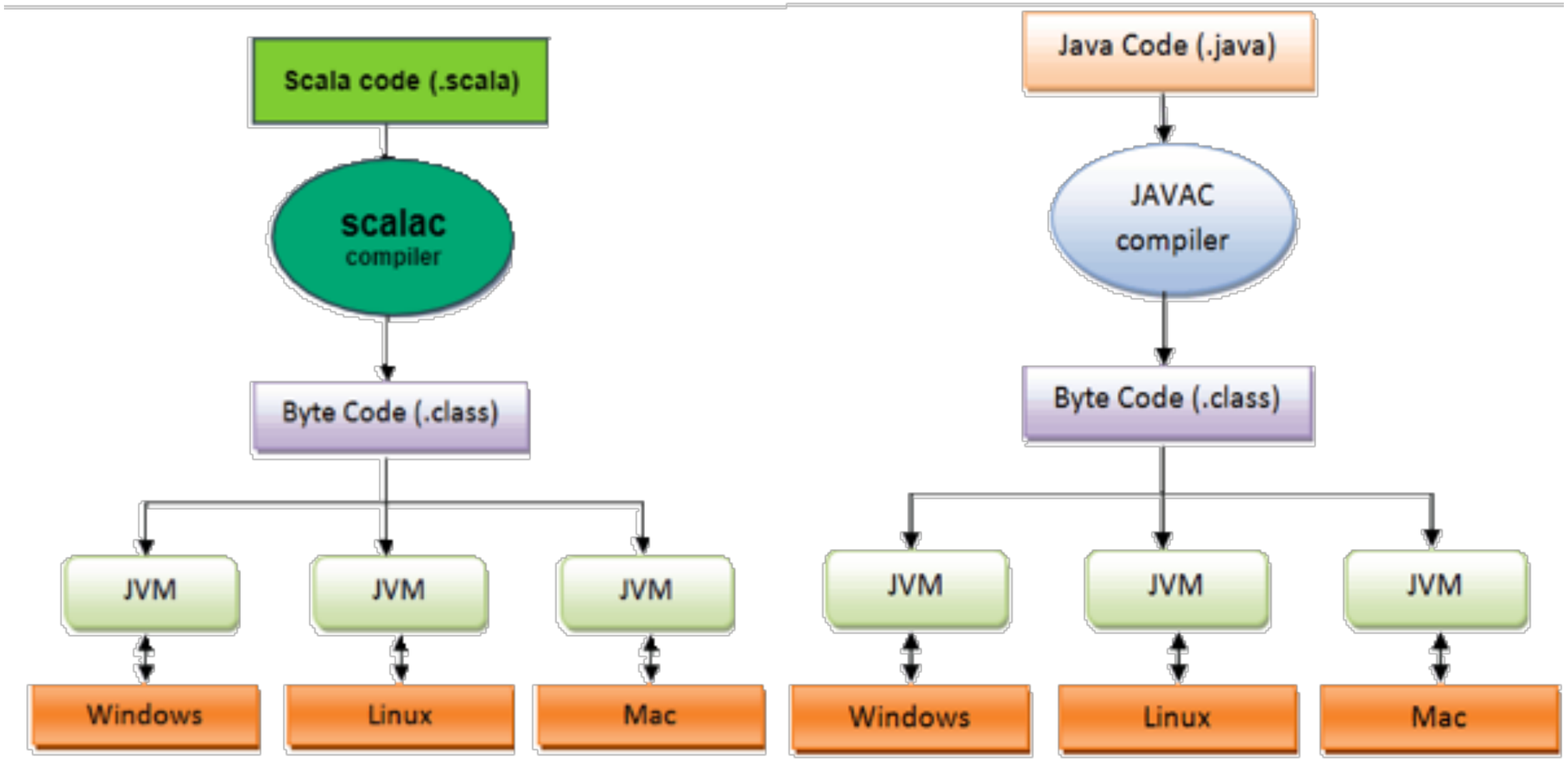
Best of both worlds

Functional Programming



Object Oriented Programming

Lives in the JVM eco system



Object Oriented

- Scala is a pure object-oriented language in the sense that every value is an object.
- Types and behavior of objects are described by classes and traits.
- Classes are extended by subclassing and a flexible mixin-based composition mechanism as a clean replacement for multiple inheritance.

Functional

- Scala is also a functional language in the sense that every function is a value.
- Scala provides a lightweight syntax for defining anonymous functions, it supports higher-order functions, it allows functions to be nested, and supports currying.
- Scala's case classes and its built-in support for pattern matching model algebraic types used in many functional programming languages.
- Singleton objects provide a convenient way to group functions that aren't members of a class.

Statically typed

- A type can be thought of as a *category* that a value (typically at run-time) can fall into. For example, the number 12 would have type `Int`, and its corresponding value would be 12.

- A language can be “statically typed” versus “dynamically checked”, or “strong” versus “weak”

- Scala is strongly statically typed, but it additionally stands out amongst other statically typed languages as having a particularly advanced advanced type system.

- Strong static typing has many benefits,

    - Correctness

    - Performance

    - Scalability

- A helpful way to think about Scala’s type system is to approach it as if types are boundaries that you erect for yourself to safeguard against wrong behavior.

- sophisticated type inference system that lets you omit almost all type information that’s usually considered as annoying.

```
val x = new HashMap[Int, String]()
val x: Map[Int, String] = new HashMap()
```

Scalable Language

- The name Scala stands for “scalable language”
- New types will feel like built in types
- Add your own control structures, that feel native

```
var capital = Map("US" -> "Washington", "France" -> "Paris")

capital += ("Japan" -> "Tokyo")

println(capital("France")) // Paris
```

Scala - a citizen of the JVM

- Seamless interoperability with Java - Scala programs compile to JVM bytecodes

- Heavily re-uses Java types

- Scala's Ints are represented as Java primitive integers of type int, Floats are represented as floats, Booleans as booleans, etc …

- Scala re-uses Java's types, and also "dresses them up" to make them nicer

- Scala lets you define implicit conversions, which are always applied when types would not normally match up

- Scala code can be invoked from Java code

Mandatory hello world

- Minimalistic runnable program - should be familiar to Java programmers!
- One method called main, which takes the command line arguments, an array of strings, as parameter
- Body of the main method consists of a single call to the predefined method println with the friendly greeting as argument
- Main method does not return a value (it is a procedure method) - not necessary to declare a return type.

```
object HelloWorld {

    def main(args: Array[String]) {
     println("Hello, world!” )
  }
}
```

Reading from input stream

- val - means value (unmutable)
- scala.io.stdin - library for handling the standard input

```
val name = scala.io.StdIn.readLine("What is your name?")
```

Compile

- Let us try to compile using the scala compiler - scalac
- This will generate a few class files in the current directory
- One of them will be called HelloWorld.class
- HelloWorld.class can be directly executed using the scala command
- Normally we use a build script, and keep source and classes in seperate folders!

```
Agatas-MacBook-Pro:helloworld_01 agatanoair$ scalac HelloWorld.scala
```

Run

- Once compiled, a Scala program can be run using the **scala** command

- Usage is very similar to the java command

- Accepts the same options as **java**

```
Agatas-MacBook-Pro:helloworld_01 agatanoair$ scala
dk.lundogbendsen.scala.labs.helloworld_01.HelloWorld
What is your name?Agata
Hello, world! Hi to, Agata
```

Call Java code

- Very easy to interact with Java code.
- Classes from the java.lang package are imported by default, others need to be imported explicitly.
- We want to obtain and format the current date according to the conventions used in Denmark - or Romania
- Java’s class libraries define powerful utility classes, such as Date and DateFormat.
- We can simply import the classes of the corresponding Java packages

```
import java.time.ZoneId
// …

val ro = ZoneId.of("Europe/Bucharest")
```

Demo Time

helloworld01/CallJavaFromScala

Use sbt

- Build, test and run
- Powerful tool, made for Scala
- Convention over configuration
- Read more at: https://www.scala-sbt.org/1.x/docs/Hello.html

```
Agatas-MacBook-Pro:scala_examples agatanoair$ sbtsbt:scala_examples>
run
Multiple main classes detected, select one to run:
 [1] class_hierarchies.helloWorld
 [2] helloworld_01.WorldTimes
 [3] polymorphism.use
Enter number: 2
[info] Running helloworld_01.WorldTimes
2018-03-14T00:58:04.381+01:00[Europe/Copenhagen]
2018-03-14T00:58:04.381+02:00[Europe/Bucharest]
[success] Total time: 2 s, completed Mar 14, 2018 12:58:04 AM
```

Lab time!

hello_world_01