

Scala Day 1



Created 2004

"Java"

By Martin Odersky

Scala → "Scalable Programming Language"

Std

JVM

CN

Java 8
functional Programming

Java
OOP

Scala

Groovy, JRuby

-functional Programming

"

Spark

framework

* Scala 176%

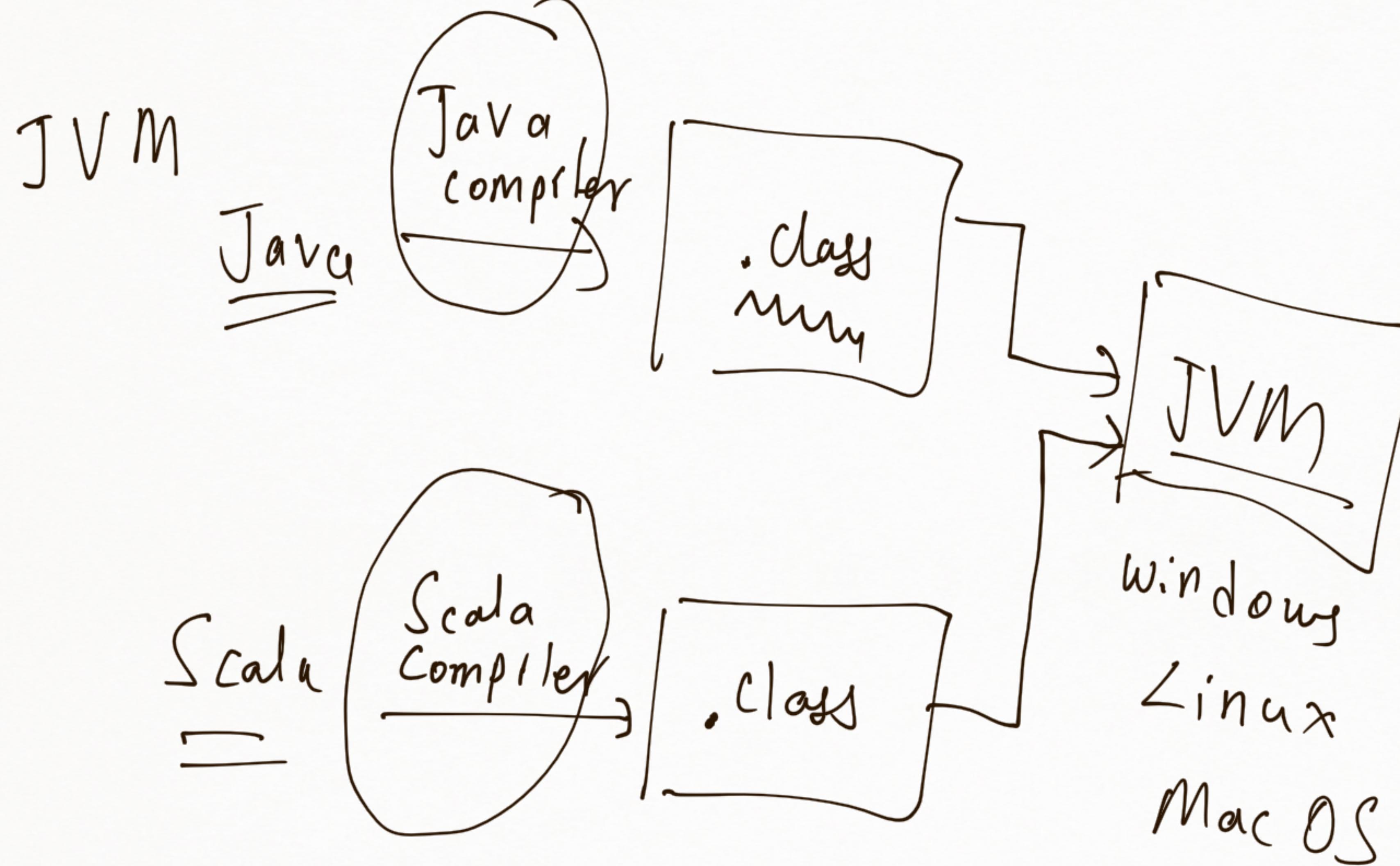
* Java 8 25%

* Python (ML)

PYSPARK

R

10%
90%



Scala

OOP

+ Functional Programming

Java 8

OOP

+ Functional Programming

Execution / Development Environment

* Java 8

* Eclipse
~~Maven~~

* Scala

Or IntelliJ
~~PyCharm~~

Or (VS Code)
~~Maven~~
Python, PyCharm

Java → Maven

Scala → SBT (build.sbt)
Scala Build Tool

Java → Excel → Read
Scala → Excel → Write → "Apache POI"
package

Java

1] OOP Language

Primitive data type : ~~Int, Char, Boolean,~~
~~Byte, Short, Long, Float, Double~~

Operators: +, -, *, /

2] public static ~~int~~ sum(~~int a, int b~~)
{
 int c = a + b;
 return c;
}

~~arguments~~
~~parameters~~

→ Imperative

def sum(a: Int, b: Int) = {a + b}

Scala

1] Pure OOP Language

~~Int~~

→ Methods +, -, *, /

3) Java
int c = a + b;
integer
primitive data type

3) Scala int int → Value
Val c = a + b → Val a = 20
Var c = a + b → Var a = 20
~~a = 15 Error~~
~~a = 12~~
Language.
Static typed
Val c: Int = a + b → c = a + 10
Var c: Int = a + b
Objects

Java

```
;  
    end of statement  
int a = 5;  
int b = 2;
```

Scala

i is optional

Statement 1

Val a = 5 ;

Statement 2

Val b = 2 ;

Val a = 5 ; Val b = 2 ;

* * *

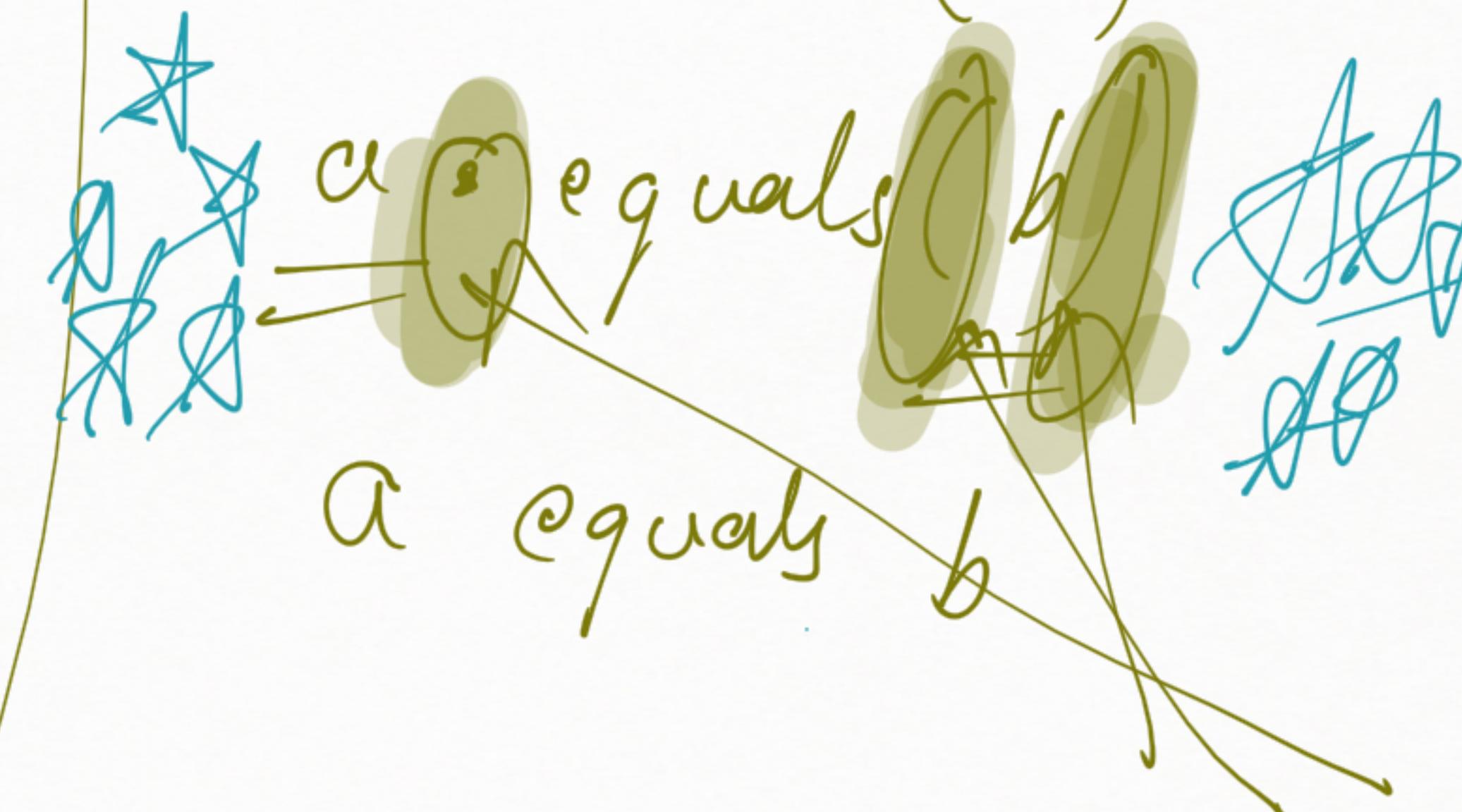
Java

sum (a, b)

Operators + - x /, ,

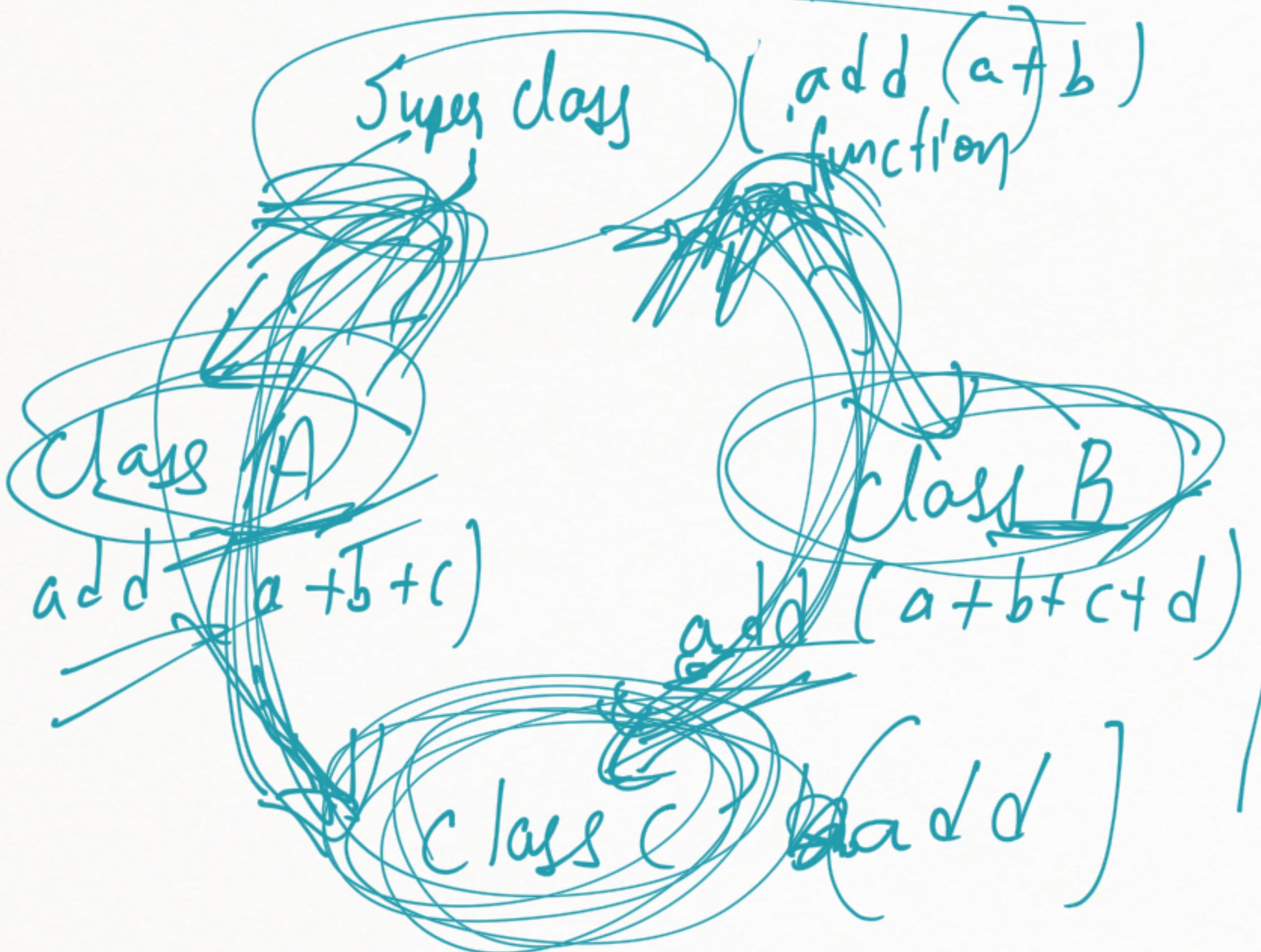
Scala .

a . sum (b)



~~(int)~~ c = a + b
~~c = a.sum(b)~~
~~sum~~
Subtract
Multiply
Divide

"Java"
"Diamond" \Rightarrow Inheritance
problem.



Scala
Trait (avoids problem)
Diamond

softwareengineering.stackexchange.com/questions/237115/how-do-trait-in-scala-avoid-the-diamond-error

StackExchange Search on Software Engineering ... Log in Sign up

Home Questions Tags Users Unanswered

21

Scala avoids the diamond problem by something called "trait linearization". Basically, it looks up the method implementation in the traits you extend from right to left. Simple example:

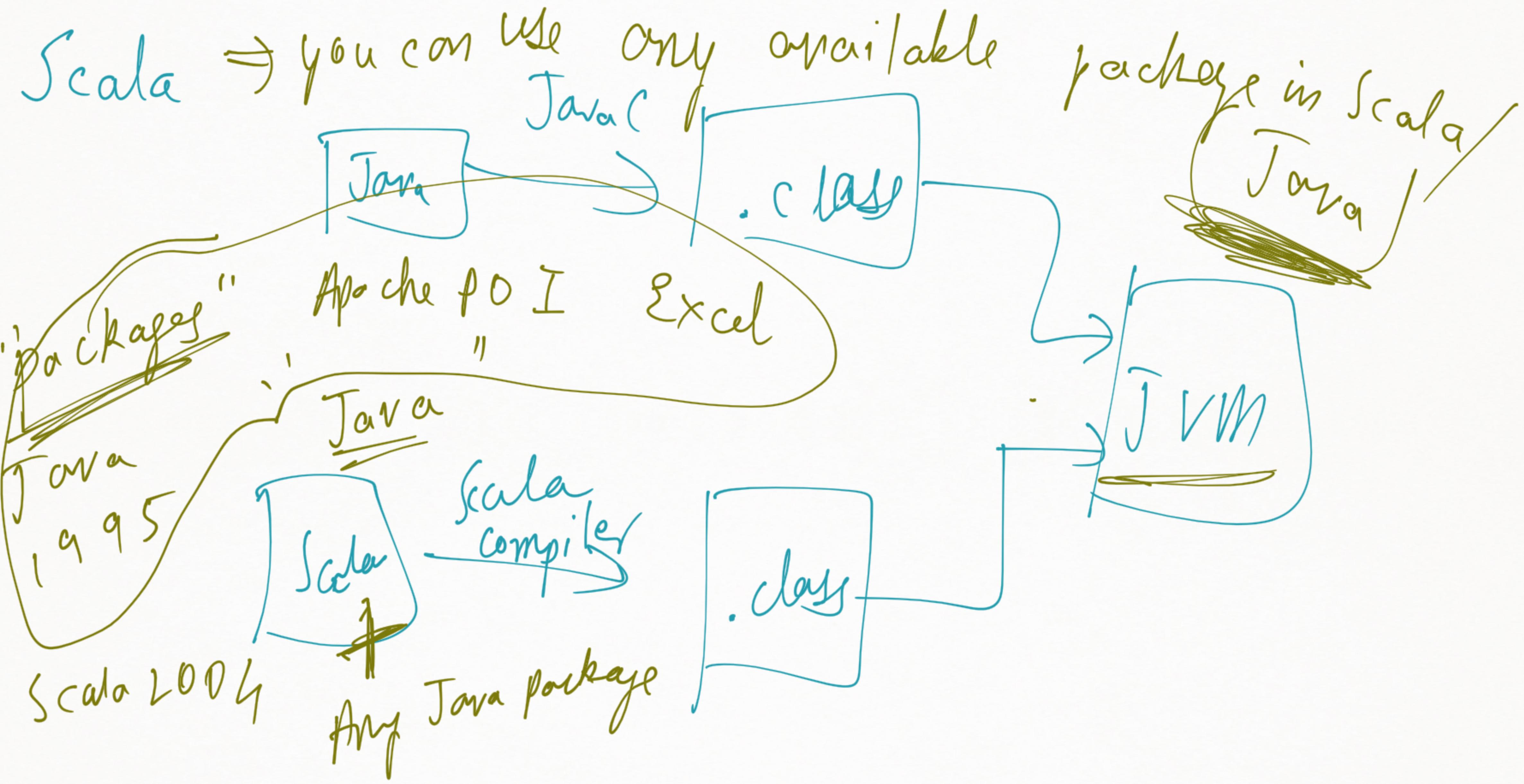
```
trait Base { def op: String } trait Foo extends Base { override def op = "foo" } trait Bar extends Base { override def op = "bar" } class A extends Foo with Bar class B extends Bar with Foo (new A).op // res0: String = bar (new B).op // res1: String = foo
```

Having said that, the list of traits it looks up might contain more than the ones you explicitly gave, since they might extend other traits. A detailed explanation is given here: [Traits as stackable modifications](#) and a more complete example of the linearization here: [Why not multiple inheritance?](#)

I believe in other programming languages this behavior is sometimes referred to as "Method resolution order" or "MRO".

share improve this answer answered May 14 '14 at 9:25 lutzh 446 3 7 add a comment

What was the point of horse armour?
What is the origin of the word "geroff"?
Low tire pressure causing fall in corner?
Trail marking material that will last a few years
Moving between Schengen Countries
Which tool to use when monitoring machines (linux+windows) with one way communication?
How does affinement work?
Usage about pronoun who
How can I take a screenshot of the app switcher in macOS?
Question feed



Refer to Work document to
install java + Scala.

