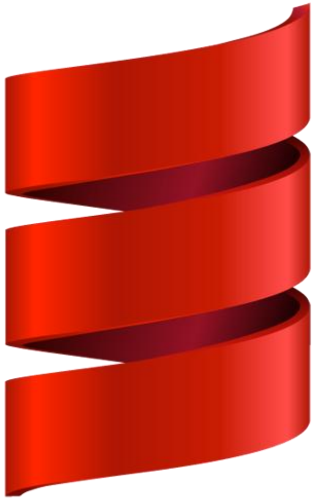
# Scala for Java Developers

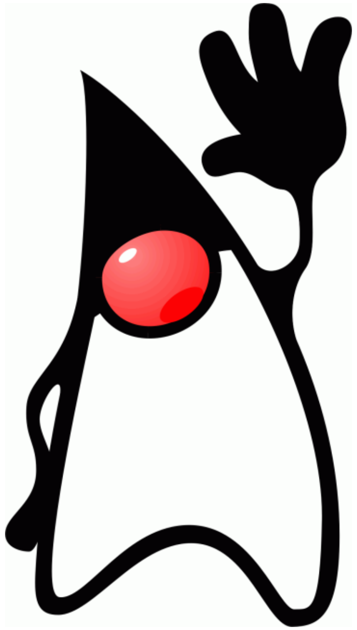
Toby Weston
http://baddotrobot.com
@jamanifin

pluralsight
hardcore dev and IT training

f(x)

if … then … else





# Modules

1

2

3

4

Module1

# Scala Tour

## The Scala Language

$f(x)$

00

$f(x)$

$f(x)+$

$f(x)+o$

f(x)+oo

inheritance

classes

objects

composition

polymorphism

…

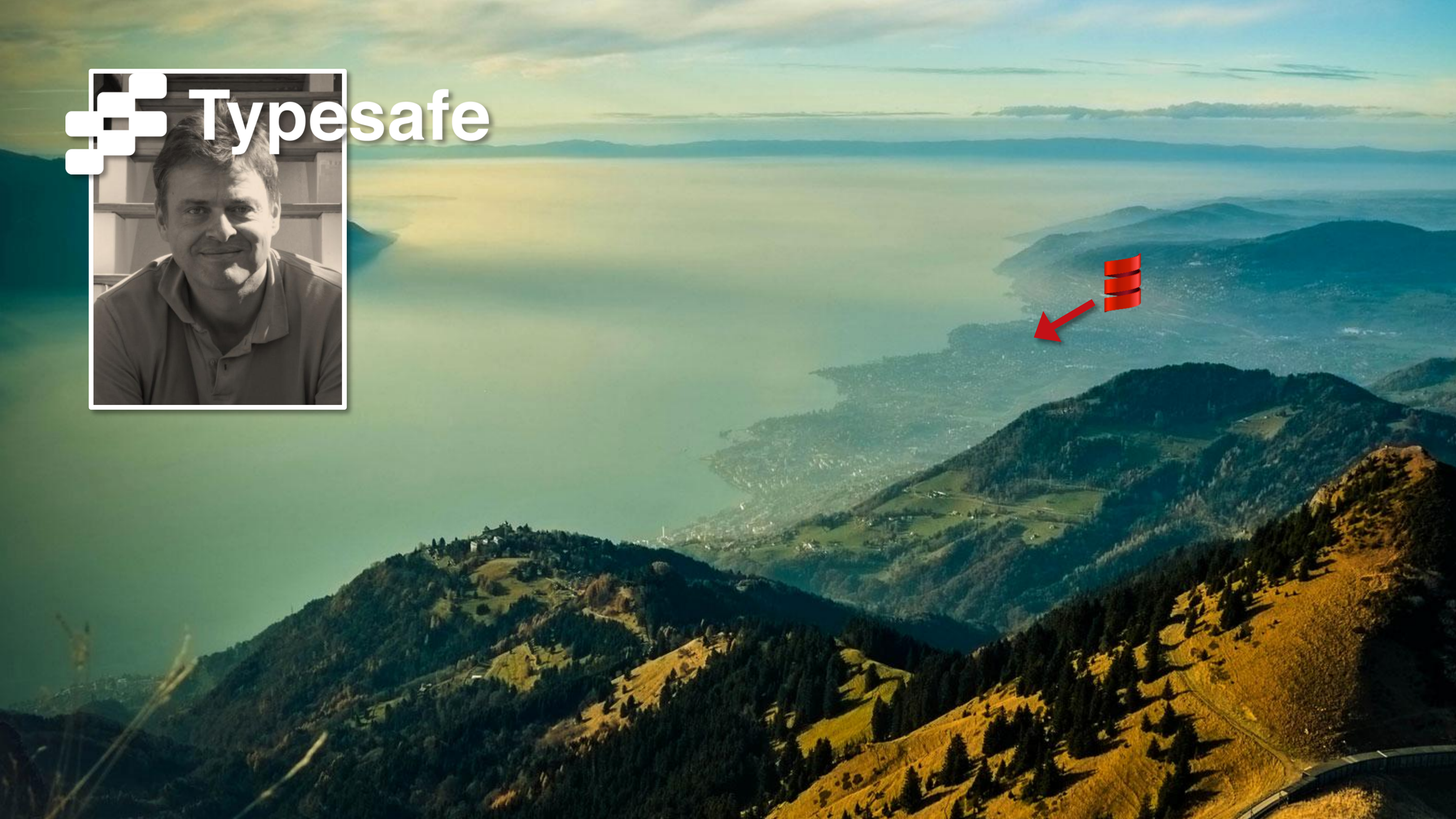
# Everything is an object

## Functional Language

☑ Pure functions

☑ Higher order functions

# Background



Typesafe

# JVM

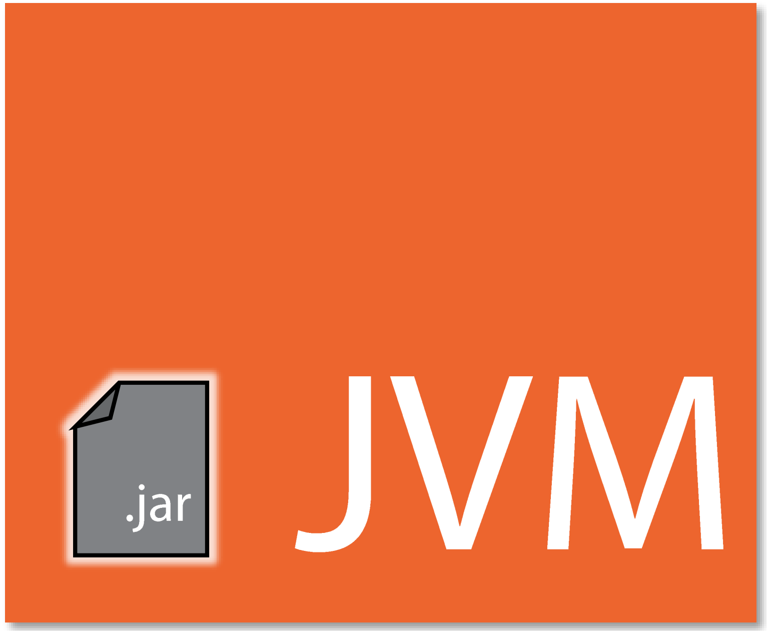
HIBERNATE

spring

mongoDB

mockito

elasticsearch.



.jar

JVM

specs2

play

Lift

scalaz

Scalatra

akka

# Adoption

ebay

YAHOO!

HSBC

Morgan Stanley

UBS

# HIGH SCORES

| | |
|---|---|
| Scala | 10,000,000 |
| Swift | 5,000,000 |
| Java | 3,250,000 |
| .NET | 2,350,000 |
| Objective-C | 2,010,000 |
| JavaScript | 1,995,000 |
| XSLT | 0 |

Module1

# Scala Tour

## Installing Scala

# 1

# Scala Interpreter

# 2

Scala Interpreter

Shell Scripts

# 3

Scala Interpreter

Shell Scripts

Compiler (`scalac`)

# 1

## Scala Interpreter

# DEMO

# 2

```
#!/bin/sh
exec scala "$0" "$@"
!#
object HelloWorld {
  def main(args: Array[String]) {
    println("Hello, " + args.toList)
  }
}
HelloWorld.main(args)
```

```
./hello.sh World!
```

2

```
#!/bin/sh
exec scala "$0" "$@"
!#
object HelloWorld {
  def main(args: Array[String]) {
    println("Hello, " + args.toList)
  }
}
HelloWorld.main(args)
```

```
$ scala hello.sh World!
```

# 3

```
scalac
scala
```

# 3

IntelliJ IDEA

Eclipse

Netbeans

# 3

Maven + `maven-scala-plugin`

IntelliJ IDEA + scala plugin

Eclipse + Scala-IDE plugin

- scala-lang.org
- scala-ide.org

SBT

SBT in your IDE

Next up…

Module 1

# Scala Tour

## Tour of the Syntax

- Variables & values
- Functions & methods

# DEMO

- Value, not a variable
- Immutability

- `val` is a constant
- `var`

```
def
```

```
def add(x: Int, y: Int): Int
def add(x: Int, y: Int): Int
def add(x: Int, y: Int): Int
```

Module 1

# Scala Tour

## Tour of the Syntax

- Operator overloading
- Infix notation
- Collections
- Java Interop

# Operator overloading

# Operators are methods

```
val age: Int = 35
```

```
age * .5
```

```
def *(x: Double): Double
```

# Numbers are objects

aaggee. **( ..55)

# 35 toString

```
35.+(10)   35 + 10
```

```
“aBC”.replace(“a”, “A”)
```

```
“aBC” replace(“a”, “A”)
```

train

train +

train + passenger

# Collections

# DEMO

```
int
```

Integer

# Everything is an object

```
Int
```

Byte

Short

Int

Long

Char

String

Float

Double

Boolean

# Scala vs. Java

```
val total = BigDecimal(10) + BigDecimal(20)
```

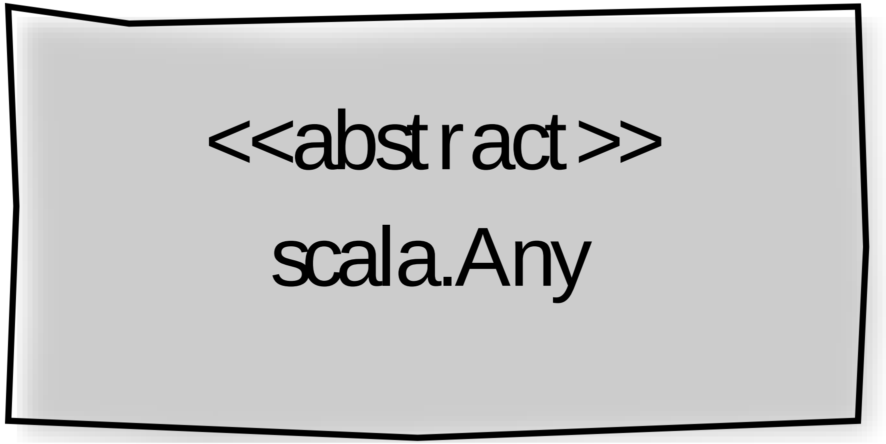
```
BigDecimal total = new BigDecimal(10).add(new BigDecimal(20);
```

Module 1

# Scala Tour

## Tour of the Syntax

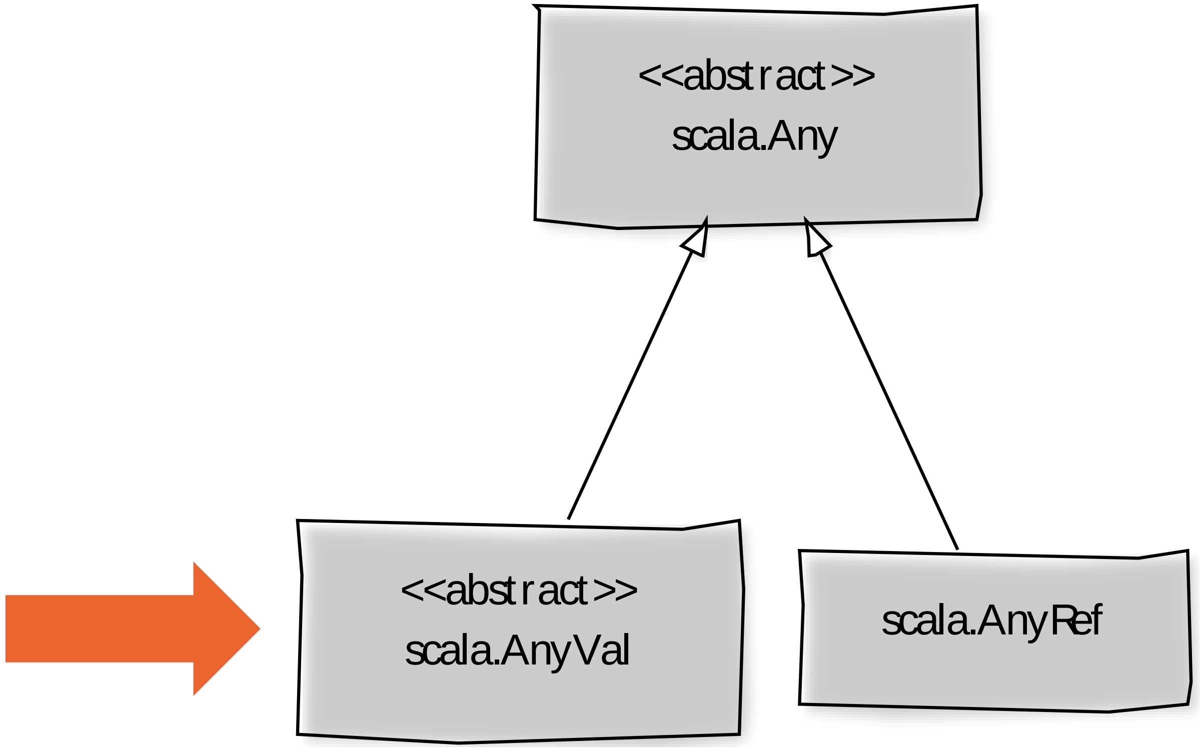
- Class hierarchy
- ScalaDoc



<<abstract>>

scala.Any

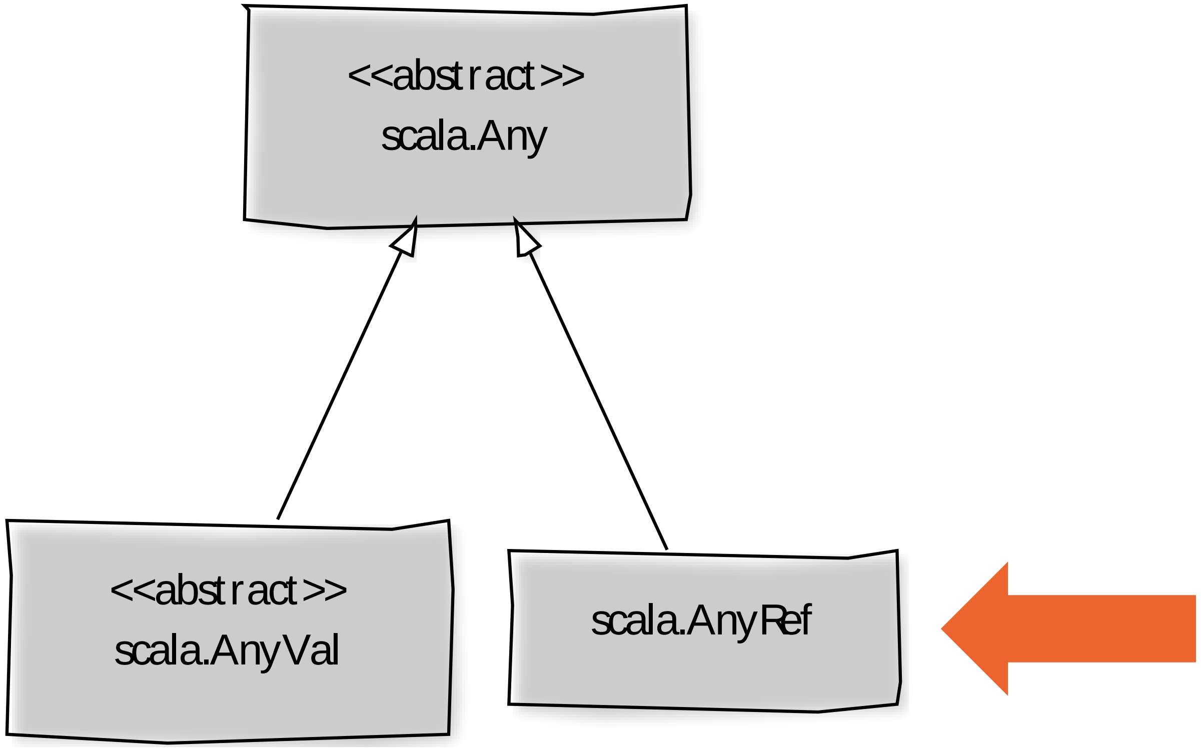
```
final def ==(that: Any): Boolean
final def !=(that: Any): Boolean
def equals(that: Any): Boolean
def ##: Int
def hashCode: Int
def toString: String
```


<<abstract>>
scala.Any

<<abstract>>
scala.AnyVal

scala.AnyRef

# *Value types*


<<abstract>>
scala.Any
<<abstract>>
scala.AnyVal
scala.AnyRef

# *Reference types*

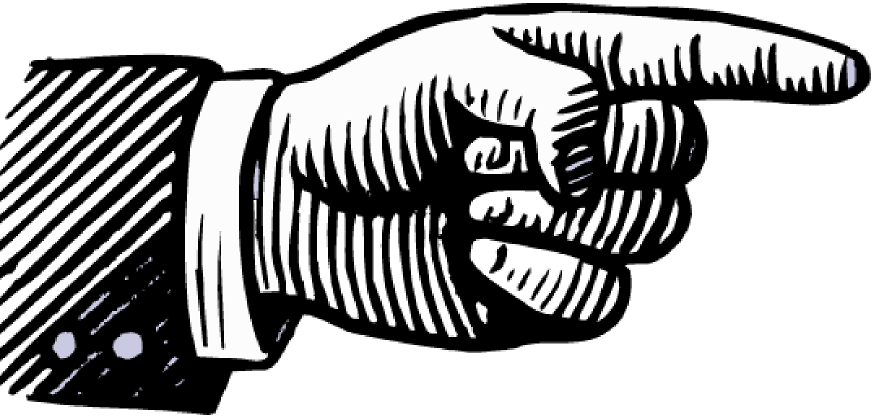
## *Value types*

*Value types* → `int, long, float, …`

*Value types* → `Int, Long, Float, …`

*Value types* → `42, 42L, 0.1, …`

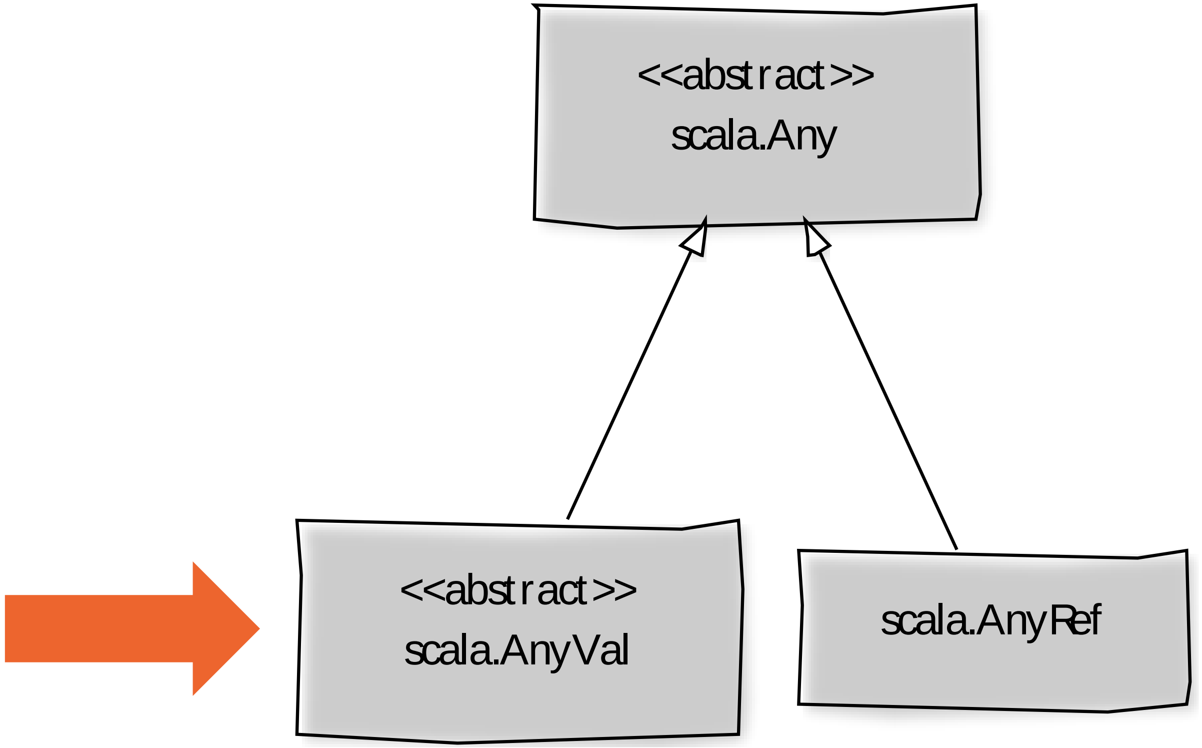
*Reference types*

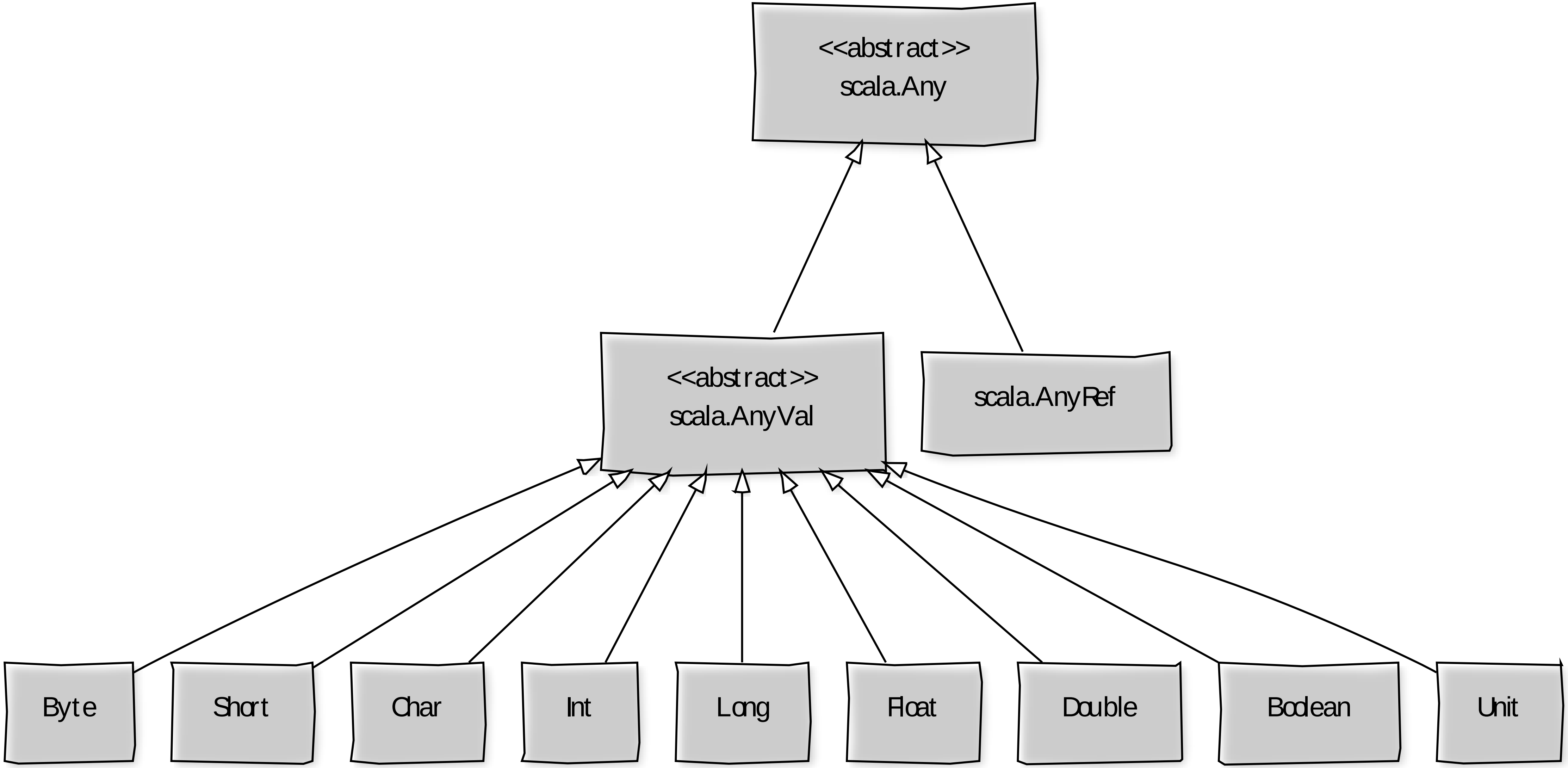


# *Pointer types*

```
*age = 64;
```

Dereferences the age pointer


<<abstract>>
scala.Any
<<abstract>>
scala.AnyVal
scala.AnyRef


<<abstract>>
scala.Any
<<abstract>>
scala.AnyVal
scala.AnyRef
Byte
Short
Char
Int
Long
Float
Double
Boolean
Unit

```
42 == 42 // scala

new Integer(42).equals(new Integer(42)); // java
```

`Int`

# Integer

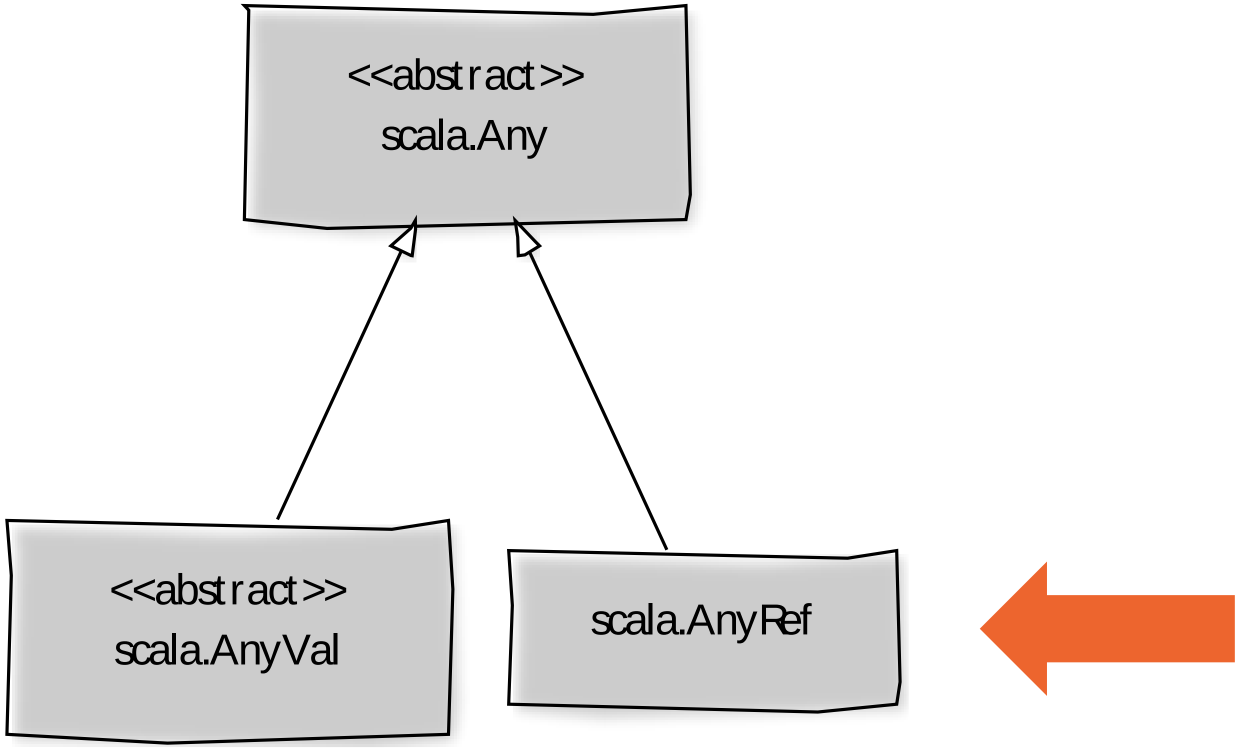
# Unit

```
Unit = Void = void
```

```
val unit: Unit = ()
val unit: Unit = ()
```

```java
public class DoNothing implements Callable<Void> {
    @Override
    public Void call() throws Exception {
        return null;
    }
}
```

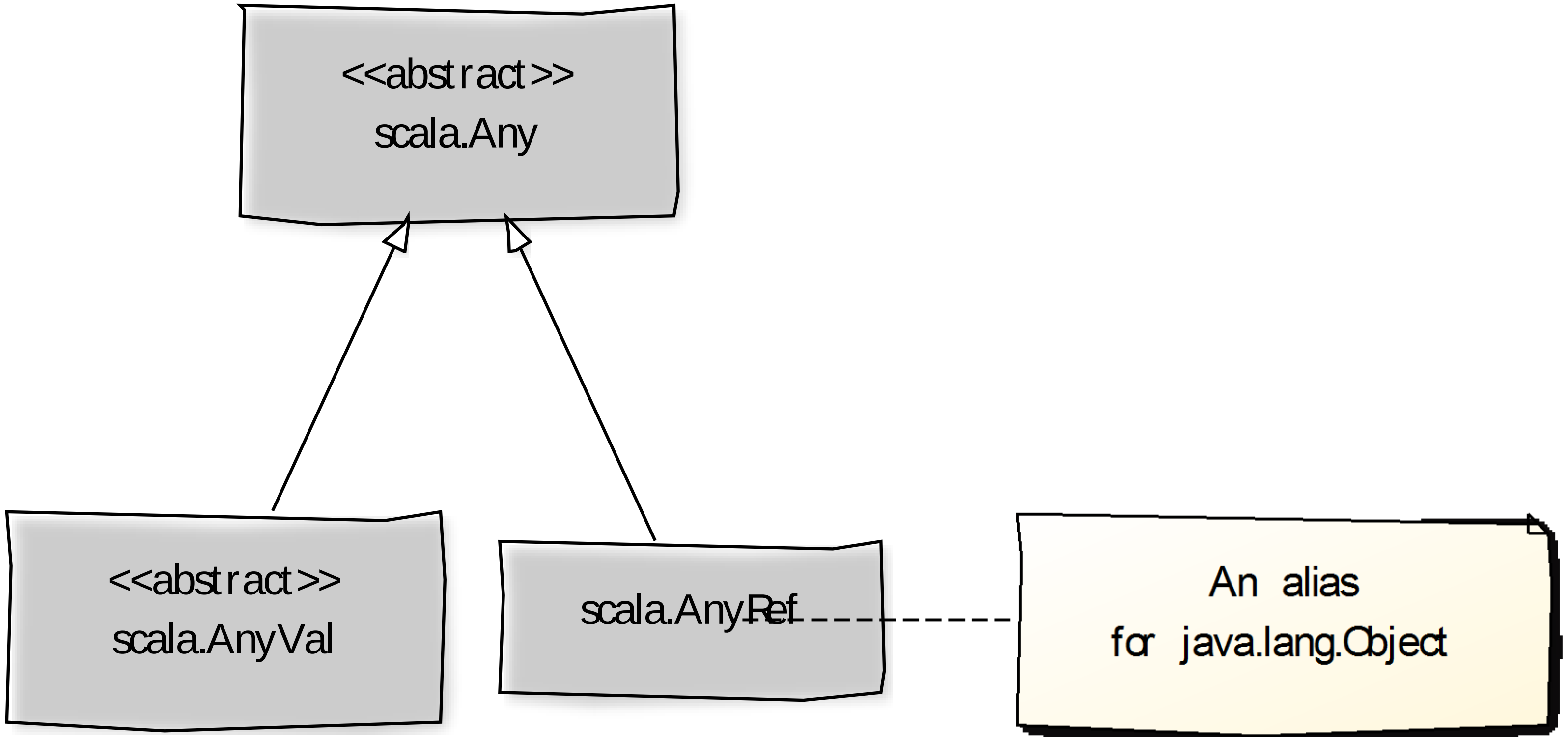
```scala
class DoNothing extends Callable[Unit] {
  def call: Unit = ()
}
```



<<abstract>>
scala.Any

<<abstract>>
scala.AnyVal

scala.AnyRef

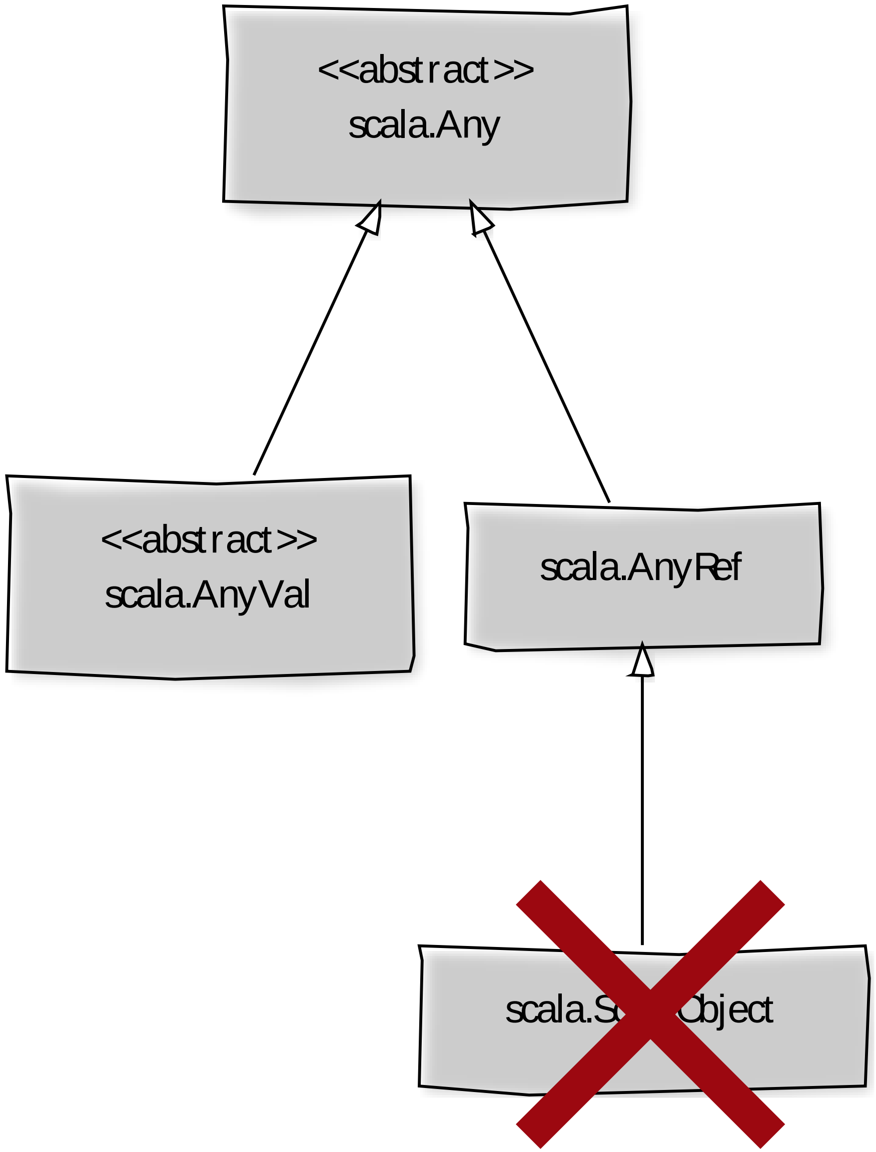


<<abstract>>
scala.Any

<<abstract>>
scala.AnyVal

scala.AnyRef

An alias
for java.lang.Object

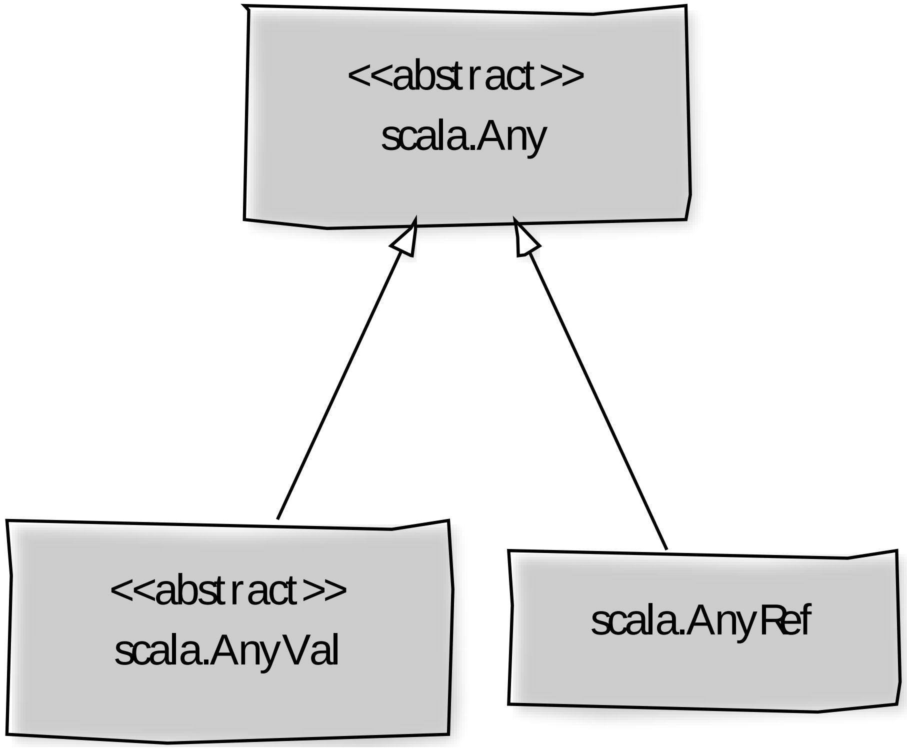


<<abstract>>
scala.Any

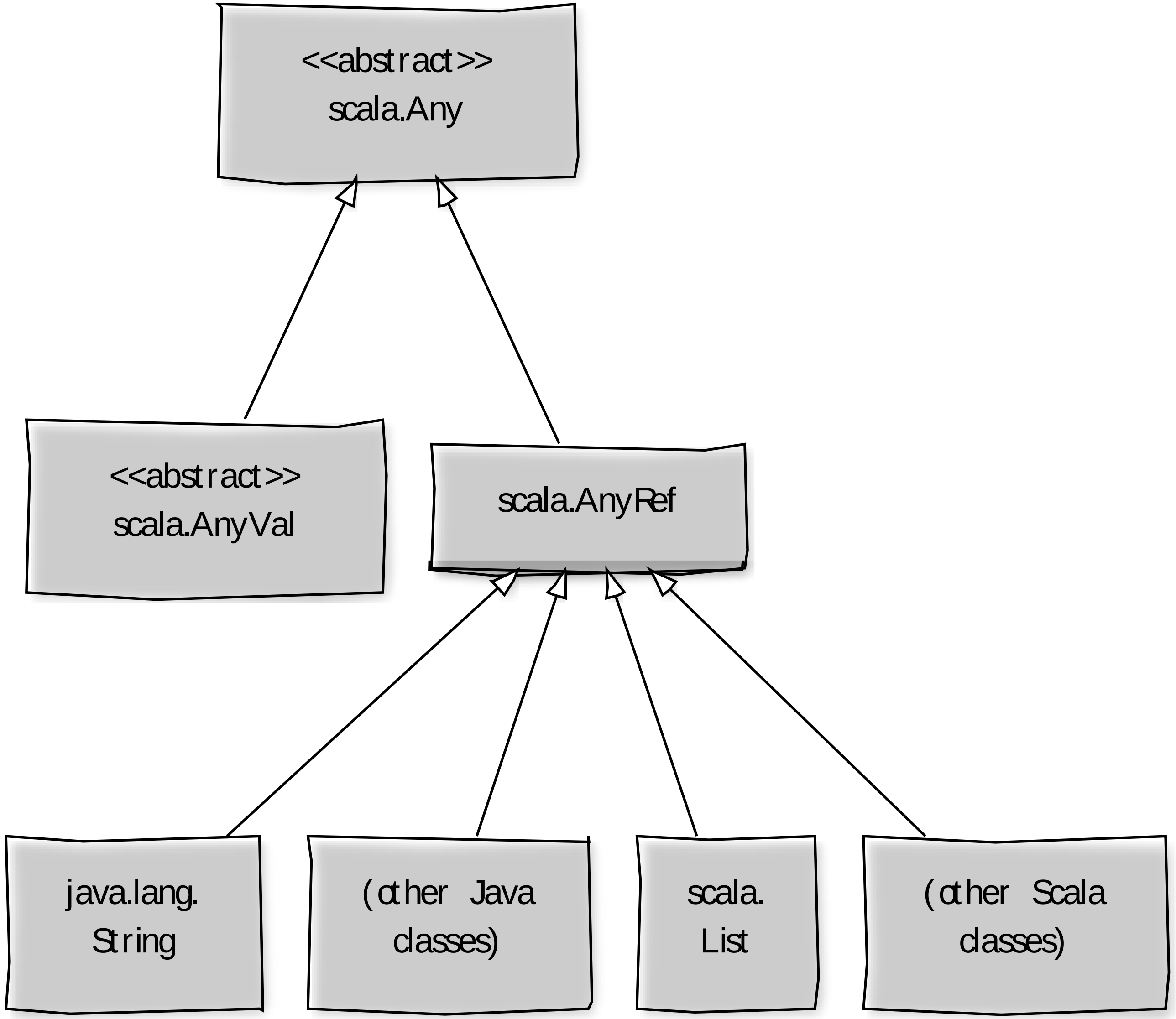
<<abstract>>
scala.AnyVal

scala.AnyRef

scala.ScalaObject



<<abstract>>
scala.Any
<<abstract>>
scala.AnyVal
scala.AnyRef

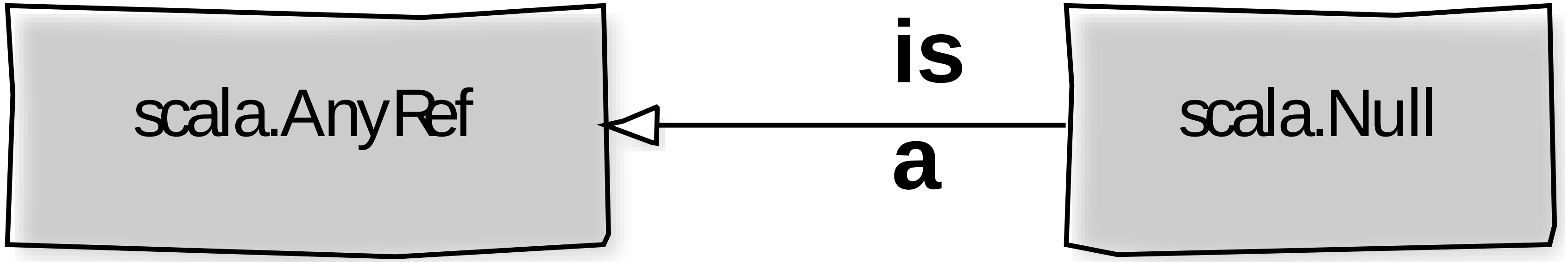

<<abstract>>
scala.Any
<<abstract>>
scala.AnyVal
scala.AnyRef
java.lang.
String
(other Java
classes)
scala.
List
(other Scala
classes)

```
new String(“A”) == new String(“A”)      // true in scala
new String(“B”).equals(new String(“B”)); // true in java
new String(“A”) eq new String(“A”) // false in scala

new String(“B”) == new String(“B”); // false in java
```

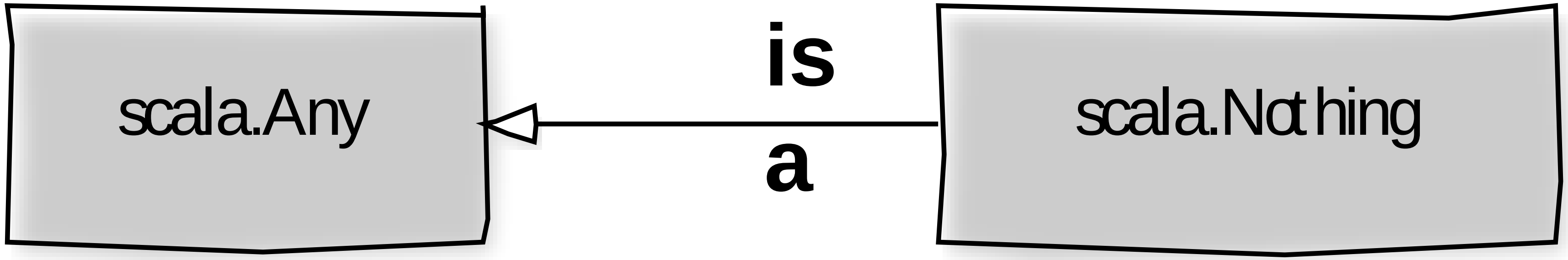
# Bottom types

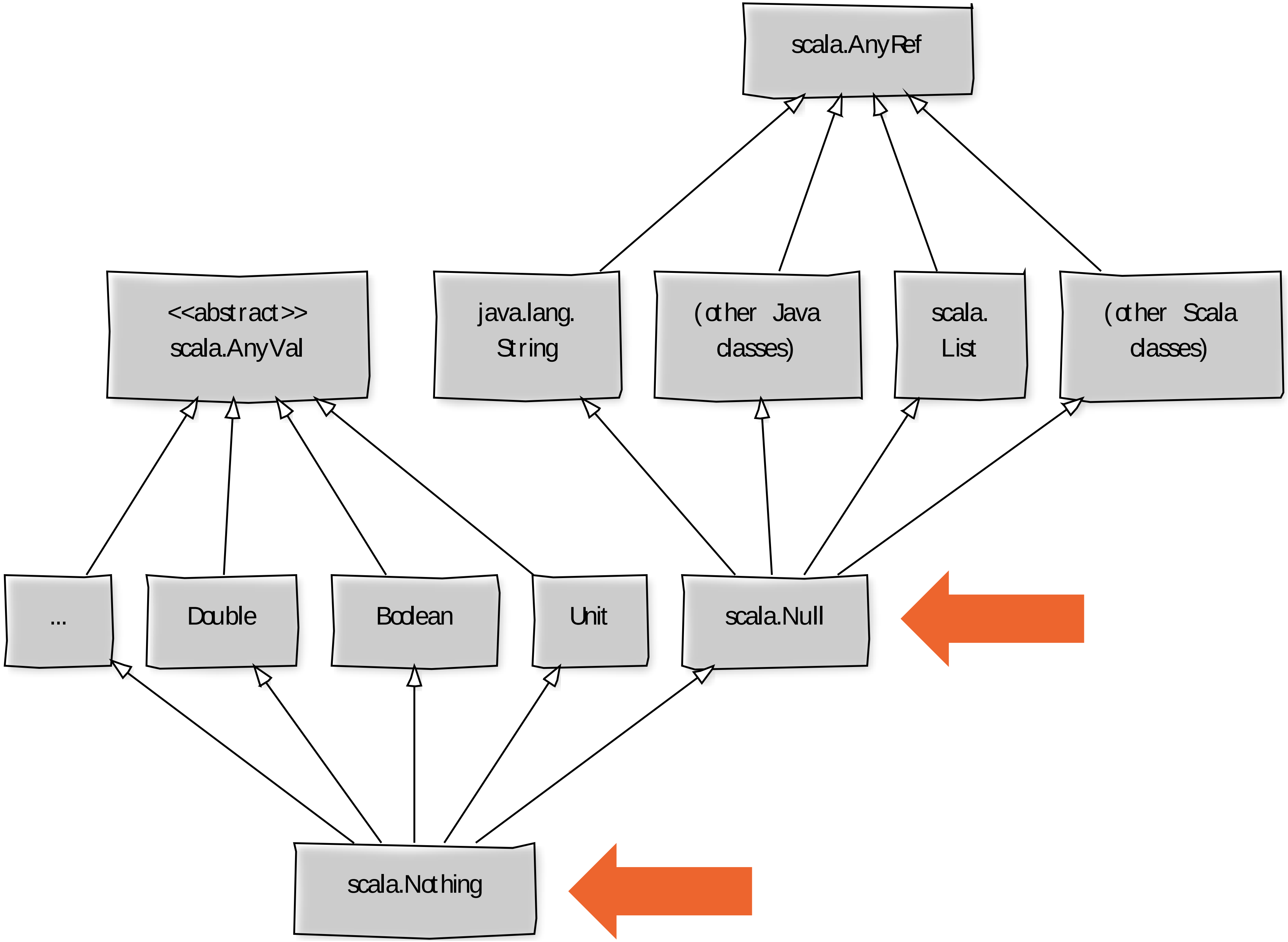
# Common sub-types

Null


scala.AnyRef
is
a
scala.Null

Nothing

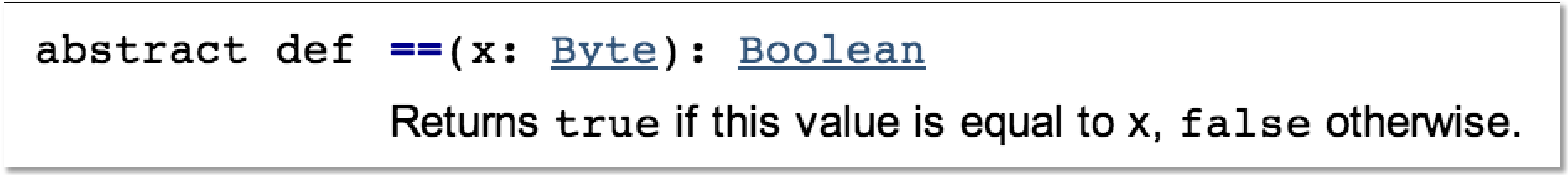

scala.Any
is
a
scala.Nothing


scala.AnyRef
<<abstract>>
scala.AnyVal
java.lang.
String
(other Java
classes)
scala.
List
(other Scala
classes)
...
Double
Boolean
Unit
scala.Null
scala.Nothing

# JavaDoc

# ScalaDoc

```scala
/** Returns `true` if this value is equal to x, `false` otherwise. */
def ==(x: Byte): Boolean
```



```
abstract def ==(x: Byte): Boolean
```

Returns `true` if this value is equal to x, `false` otherwise.

# ScalaDoc

# DEMO

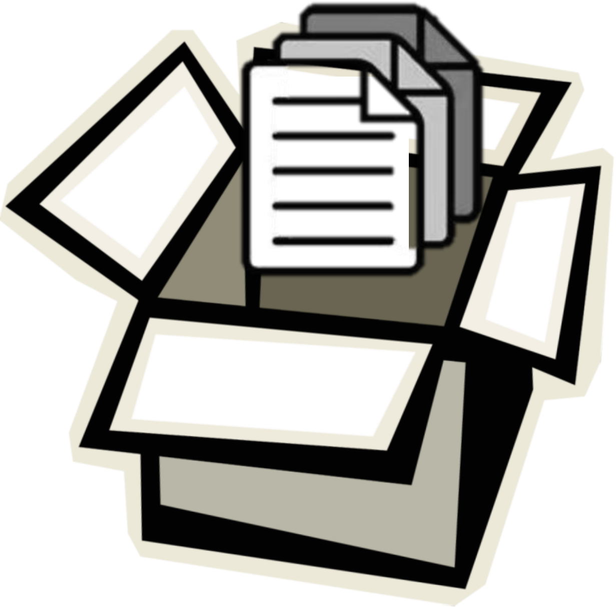
Module 1

# Scala Tour

## Language Features

- Source Files

- Source Files
- Packages

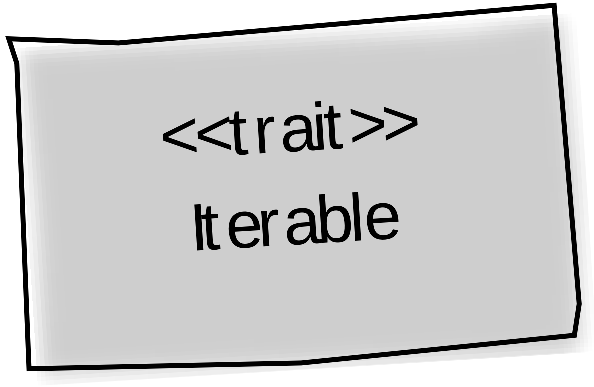


- Source Files
- Packages
- Package objects

- Source Files
- Packages
- Package objects
- Imports

- Source Files
- Packages
- Package objects
- Imports
- Type aliases

- Source Files
- Packages
- Package objects
- Imports
- Type aliases
- Traits



- Generics

```
class Stack[+A] {
  def push[B >: A](b: B): Stack[B] = ...
}
```

- Generics
- Variable Arguments

```
public add(String... names)  // java

def add(names: String*)      // scala
```

- Generics
- Variable Arguments
- Named Arguments

```
def swap(first: Int, second: Int)

swap(first = 3, second = 1)
swap(second = 1, first = 3)
```

- Generics
- Variable Arguments
- Named Arguments
- Default Values

```
def swap(first: Int, second: Int = 1)

swap(3)
swap(3, 2)
swap(first = 3)
swap(first = 3, second = 1)
```

- Generics
- Variable Arguments
- Named Arguments
- Default Values
- Lambdas

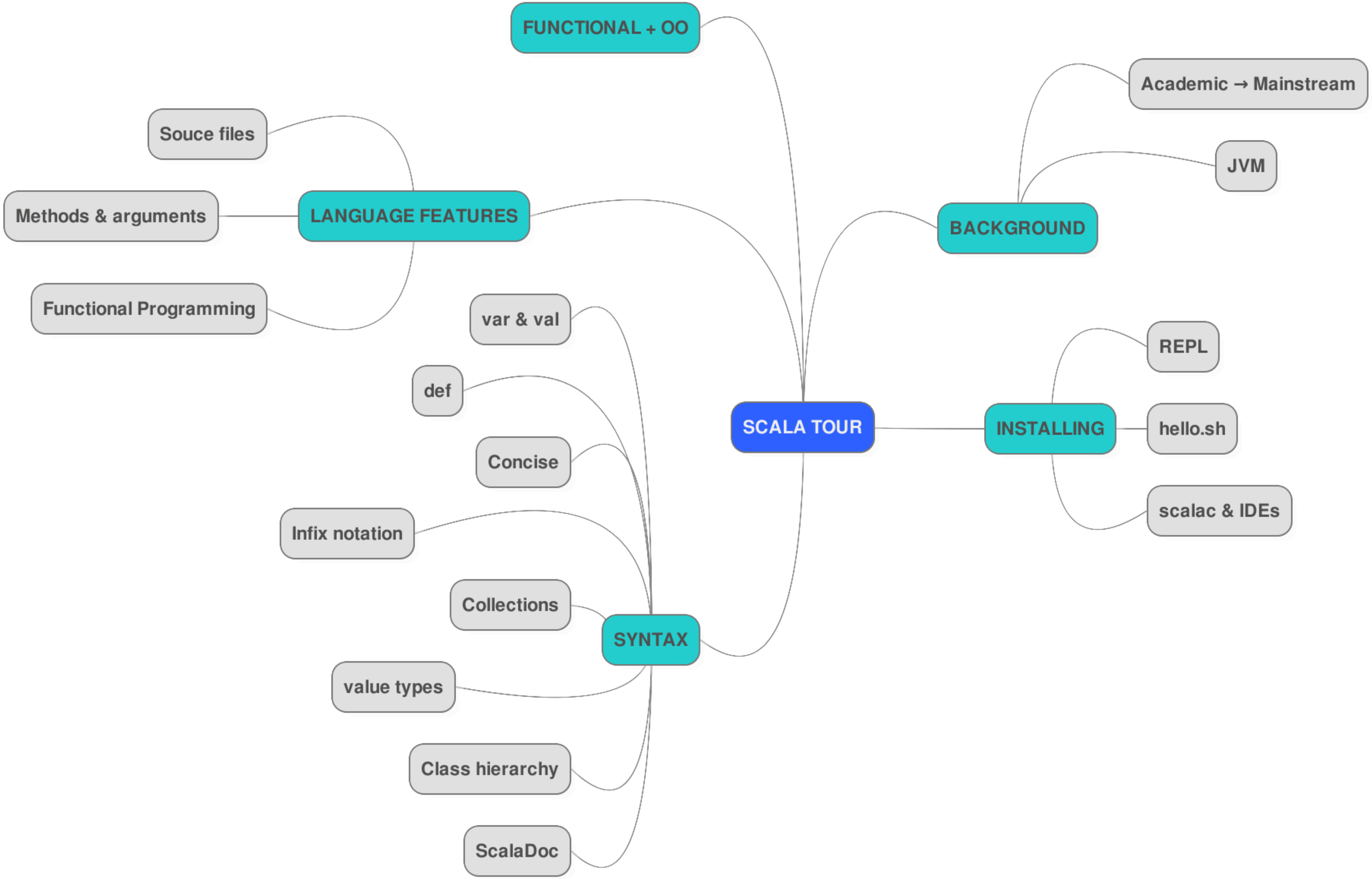
```
def test(f: () => Boolean): Boolean = ...

test(() => if (!tuesday) true else false)
```

- Pattern Matching
- For Comprehensions
- Currying
- Functional Literals (i.e. tuples)
- Tail Call Optimisation (`@tailrec`)

Module 1

# Scala Tour

## Module Summary



SCALA TOUR

FUNCTIONAL + OO

BACKGROUND

Academic → Mainstream

JVM

INSTALLING

REPL

hello.sh

scalac & IDEs

LANGUAGE FEATURES

Souce files

Methods & arguments

Functional Programming

SYNTAX

var & val

def

Concise

Infix notation

Collections

value types

Class hierarchy

ScalaDoc