Scala EE: миф или реальность?

или

Стоит ли мне использовать Scala?

или

Об истории одного проекта на Scala

Марат Ахин

Application Developer Days – 3

12-05-2012

http://ice-phoenix.info/info

Application Developer Days

Scala EE

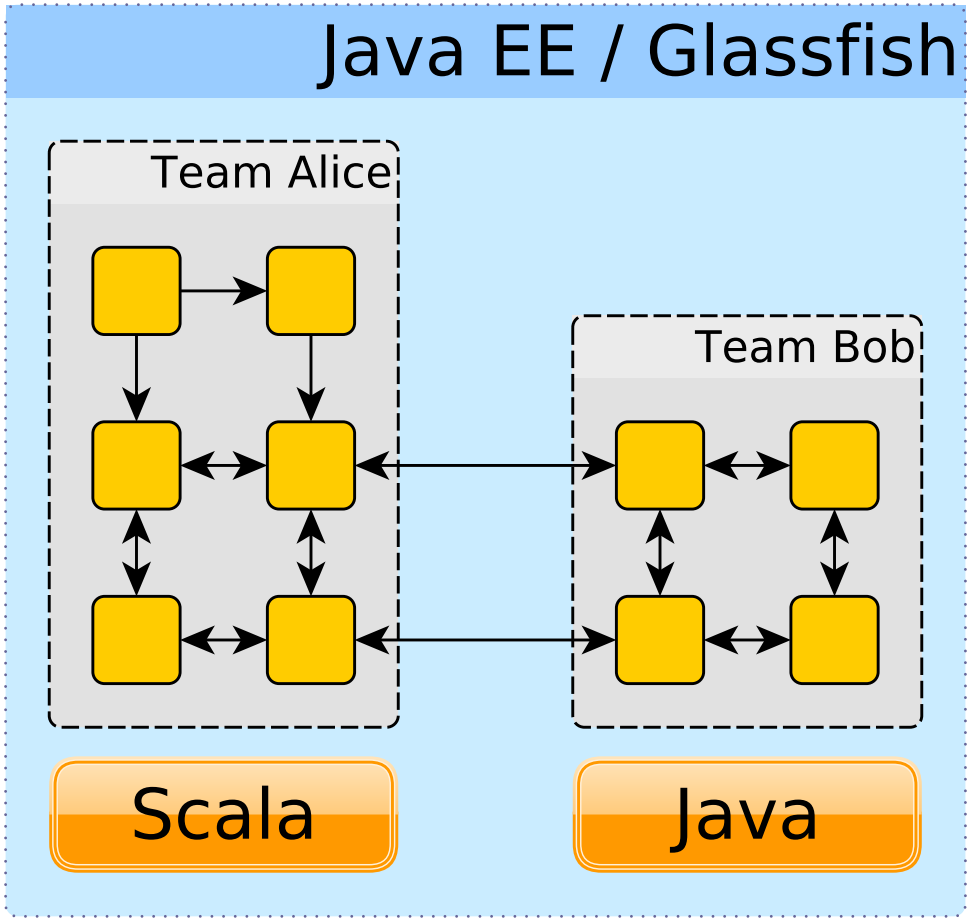
Scala

- Объектно-ориентированный и функциональный язык программирования
- Обладает очень выразительной системой типов
- Выполняется на Java Virtual Machine (JVM)

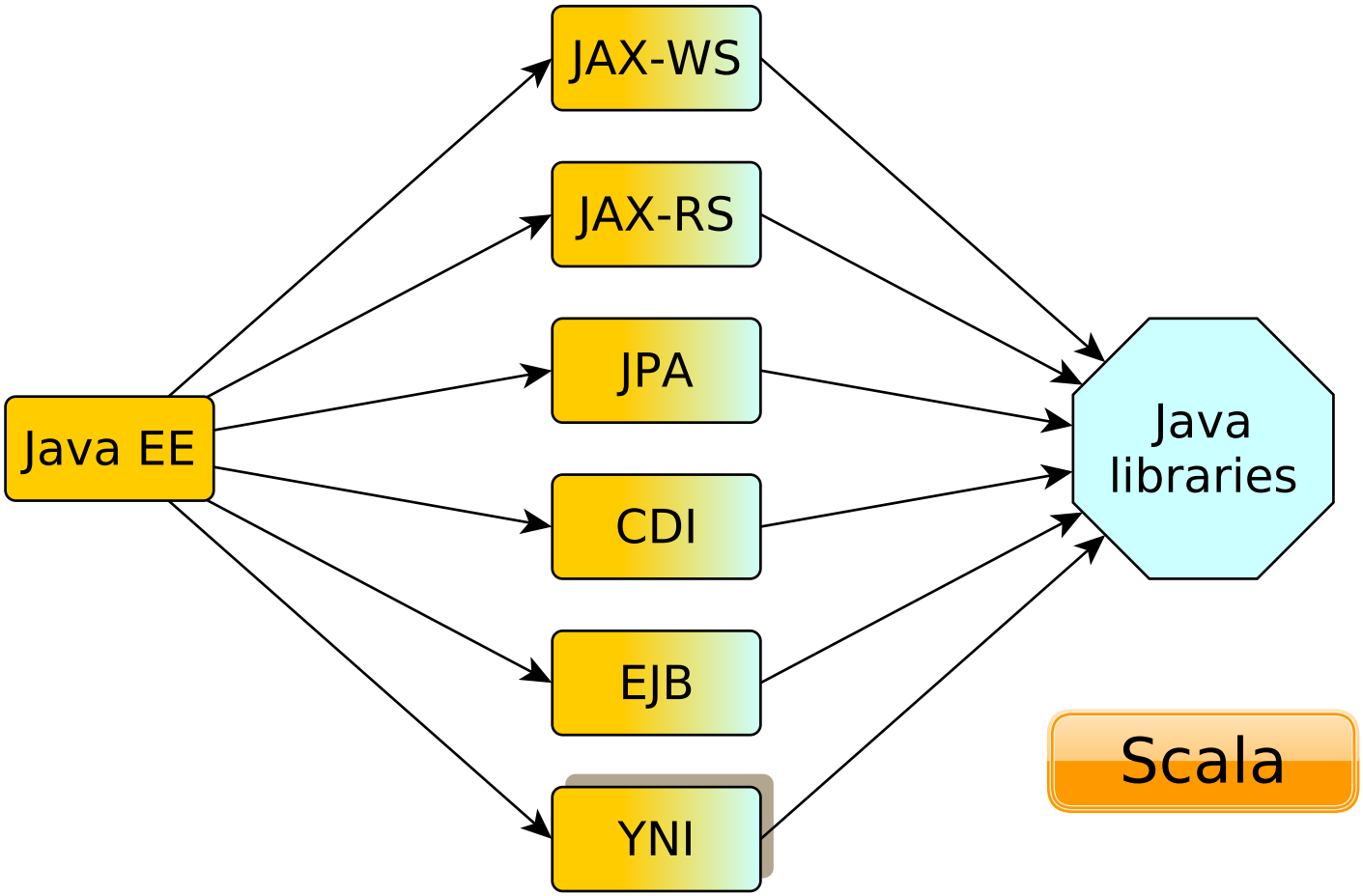
Java EE

- Набор технологий для разработки серверных приложений
- Расширяет Java SE
- Позволяет сфокусироваться на бизнес-логике

Пилотный проект



Java EE



Что такое хорошо, что такое плохо

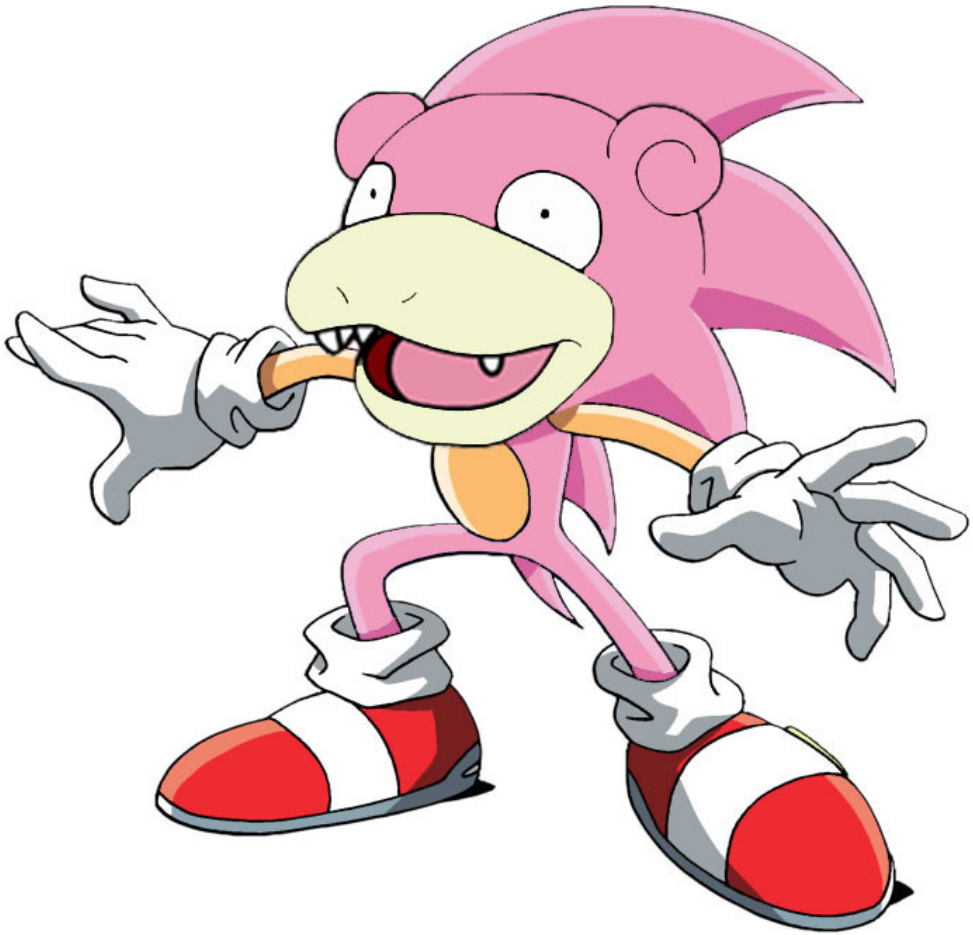
Интероперабельность между Scala и Java

- Способность вызывать Java из Scala и наоборот
- Прозрачная работа с объектами Scala в Java и наоборот

Использование Scala вместо Java

- Большая выразительность Scala
- Разница в производительности между Scala и Java

Проблемы с производительностью



Проблемы с производительностью


```
1 def main(args: Array[String]) {
2   var sum = 0.0;
3   val array = (1 to 16000000).map(e => random)
4   val start = System.currentTimeMillis()
5   for (e <- array) {
6     sum = sum + e
7   }
8   val end = System.currentTimeMillis()
9 }
```

Аналогичный код на Java выполняется за 30 мс[1]

За сколько выполняется код на Scala?

[1] на сферической тестовой машине в вакууме

Проблемы с производительностью

Все дело в том, что...

```
1 def main(args: Array[String]) {
2   // ...
3   array.foreach(
4     new AbstractFunction1() {
5       def apply(e: Double) = { this.sum += e; }
6     }
7   );
8   // ...
9 }
```

Для вложенных циклов ситуация обстоит еще хуже

Если код на Scala выглядит как быстрый код на Java, это не значит, что он является таковым

Проблемы с коллекциями

HashMap Read / Write Performance

- java.util.HashMap: 1x / 1x
- scala.collection.immutable.HashMap: 2.5x / 4x
- scala.collection.mutable.HashMap: 1.05x / 1.05x

TreeMap Read / Write Performance

- java.util.TreeMap: 1x / 1x
- scala.collection.immutable.TreeMap: 1.5x / 2x

Проблемы с коллекциями

Почему immutable.HashMap так важен?

- Scala по умолчанию использует immutable коллекции
- Многие операции неявно приводят к созданию immutable.HashMap

```
1 def initCache(rs: List[ActionStatus]) {
2   cache = rs
3           .map(e => (e.getName, e))
4           .toMap[String, ServerStatus]
5 }
```

Понятный код на Scala не всегда приводит к понятным результатам

Скорость компиляции

Скорость компиляции


```
java   : core-common       : 2 s
java   : core-database     : 2 s
java   : core-business     : 2 s
java   : core-integration  : 1 s
java   : ws-one            : 1 s
java   : ws-two            : 1 s
java   : ws-three          : 1 s
++++++++++++++++++++++++++++++++
java   : total             : 10 s
```

Total Java code size: $\approx$ 60 KSLOC

Скорость компиляции


```
scala : core-common       : 16 s
scala : core-database     : 13 s
scala : core-business     : 13 s
scala : core-integration  :  6 s
scala : ws-one            :  8 s
scala : ws-two            : 10 s
scala : ws-three          :  6 s
++++++++++++++++++++++++++++++++
scala : total             : 72 s

Total Scala code size: ≈ 8 KSLOC
```

Скорость компиляции

Почему?

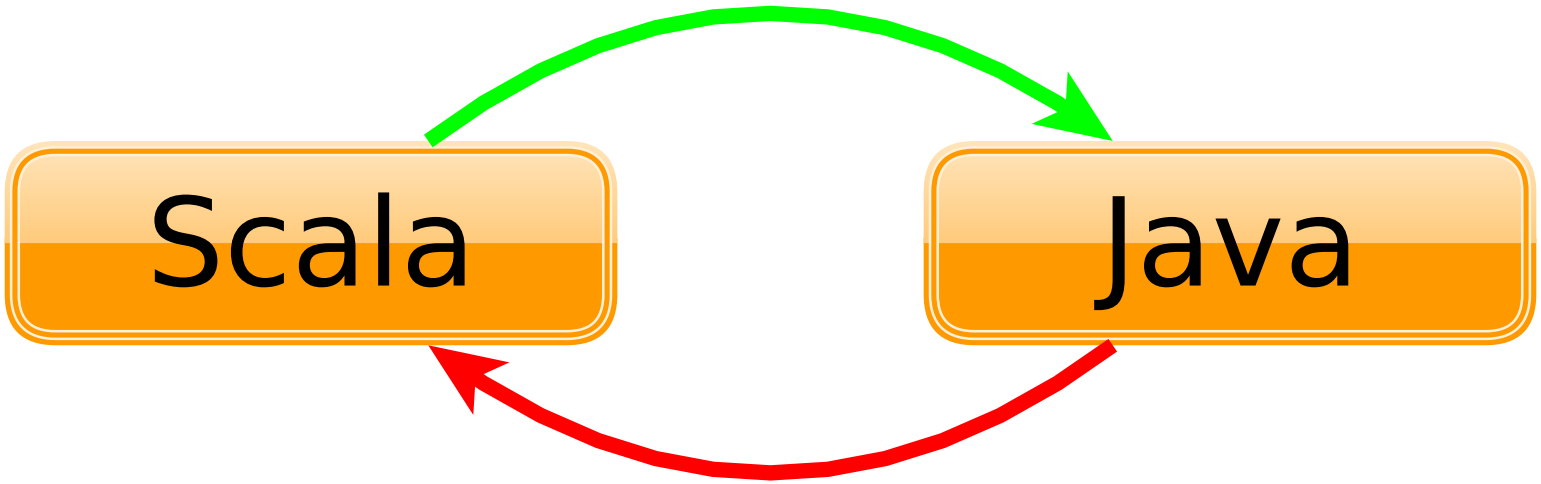
- Система типов Scala
  - Вывод типов
  - Неявные преобразования типов
  - Множество расширений по сравнению с Java

За богатые выразительные возможности приходится расплачиваться временем компиляции

Интероперабельность

Интероперабельность

Способность прозрачным образом работать с объектами и вызывать методы одного языка из другого



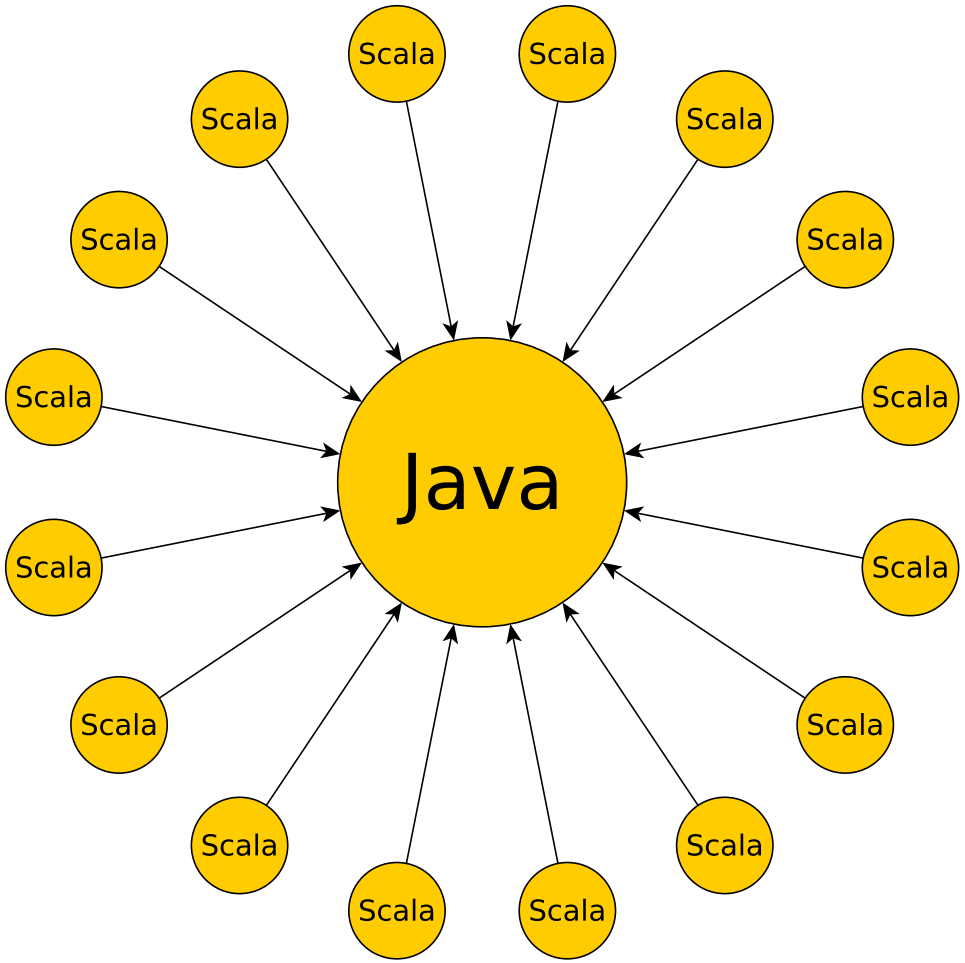
Интероперабельность

Scala -> Java

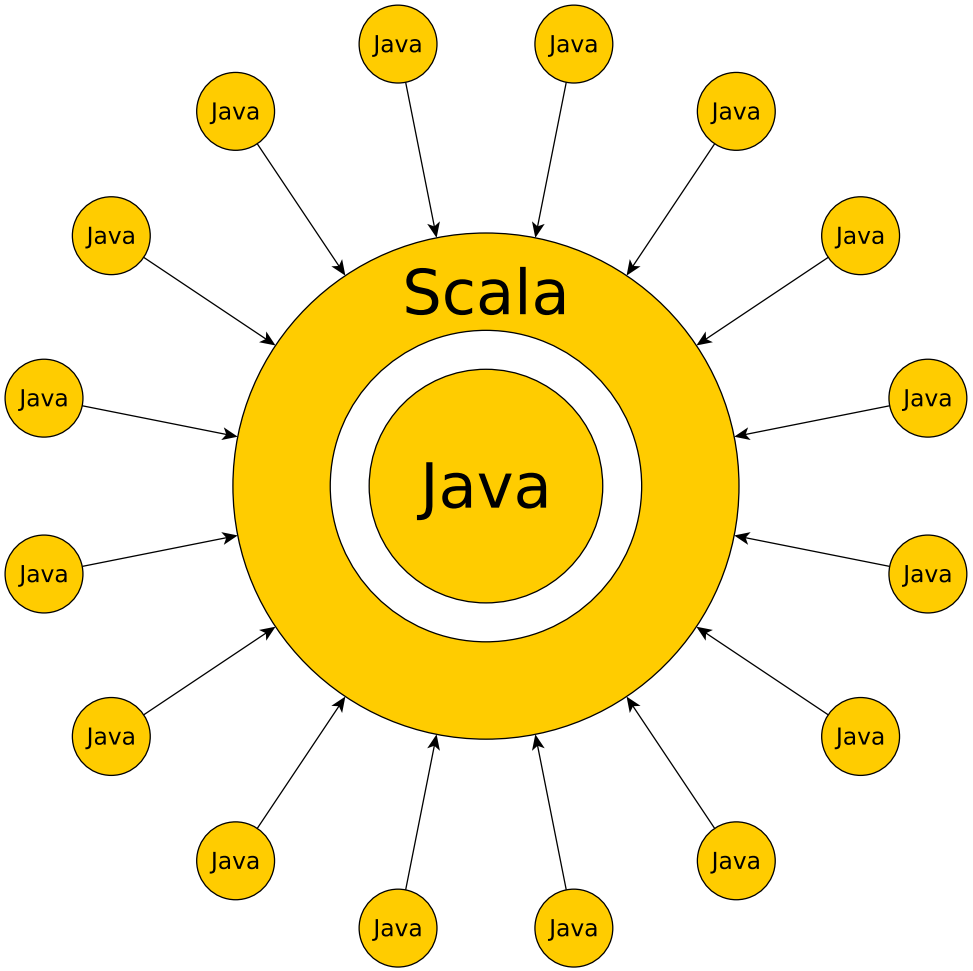
- Все объекты Java являются полноценными объектами Scala
- Все методы Java могут быть вызваны в Scala
- Все библиотеки Java доступны в Scala

```
1 var ServiceUrl =
2   new URL("http://web-service.net/ws/foo")
3
4 def HttpAuthToken =
5   DatatypeConverter.printBase64Binary(
6     (User + ":" + Password).getBytes)
```

Интероперабельность



Интероперабельность



Интероперабельность

Java -> Scala

- Все объекты Scala являются полноценными объектами Java
- Все методы Scala могут быть вызваны в Java
- Все библиотеки Scala доступны в Java

Но есть одна проблема...

Scala компилируется в очень специфический Java код

Специфический код


```
object ConfigManager extends Configuration {
  var DateFormatter =
    new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
  // ...
}
```

```
public void doSomething() {
  SimpleDateFormat sdf =
    ConfigManager$.MODULE$.DateFormatter();
  // ...
}
```

И это один из самых простых случаев...

Очень специфический код


```
object ConfigManager extends Configuration {
  // ...
  var LookupMap =
    Map.empty[String, Object] + ("ID01" -> ...)
  // ...
}
```

```
public void doNetherthing() {
  Map<~> map = ConfigManager$.MODULE$.LookupMap();
  map = map.$plus(...);
  // ...
}
```

Совсем уж специфический код


```
object ConfigManager extends Configuration {
  val Auth = new Configuration {
    val ErrorCodes = new Configuration {
      var FatalError = 0x01
      // ...
    }
  }
}
```

```
public void doNetherthing() {
  int fatalErrorCode =
    ConfigManager$$anon$1$$anon$9.FatalError();
  // ...
}
```

И снова о компиляции

Стандартная схема компиляции

```
javac *.java -> scalac *.scala
```

Все помнят проблемы со временем компиляции Scala?

Расширенная схема компиляции

```
scalac * -> javac *.java -> scalac *.scala
```

А теперь?

Scala код и библиотеки Java

Специфический Scala->Java код представляет собой проблемы для многих Java библиотек

```scala
class CommonData extends I18nable with Logging {
  // ...
}
```

```scala
// From grizzled.slf4j
trait Logging {
  protected def logger : Logger = ...
  // ...
}
```

JAXB + CommonData = Does not work

Scala код и библиотеки Java

- EJB + Wrapped Scala collection = Does not work
- JAX-WS + Scala traits = Does not work
- JPA + Scala collections = Does not work
- ...

Чтобы Scala могла работать вместе с Java, иногда приходится писать Java-подобный Scala код

Надежды нет?

Надежды нет?

Единственное преимущество в использовании Scala вместо Java

Надежды нет?

Scala позволяет писать более лаконичный и выразительный код

- Сопоставление с образцом
- Вывод типов
- Функции высших порядков
- Структурные типы

Pattern matching

Pattern matching

Switch на стероидах

```
1  def initNodeMapings() {
2    // ...
3    nodeKind match {
4      case BLOCK | IMPORT =>
5        (m + (nodeKind -> i), i + 2)
6      case MODIFIERS =>
7        (m + (nodeKind -> i), i + Modifiers.size)
8      case PRIMITIVE_TYPE =>
9        (m + (nodeKind -> i), i + TypeKinds.size)
10     case _ =>
11       (m + (nodeKind -> i), i + 1)
12   }
13   // ...
14 }
```

Pattern matching

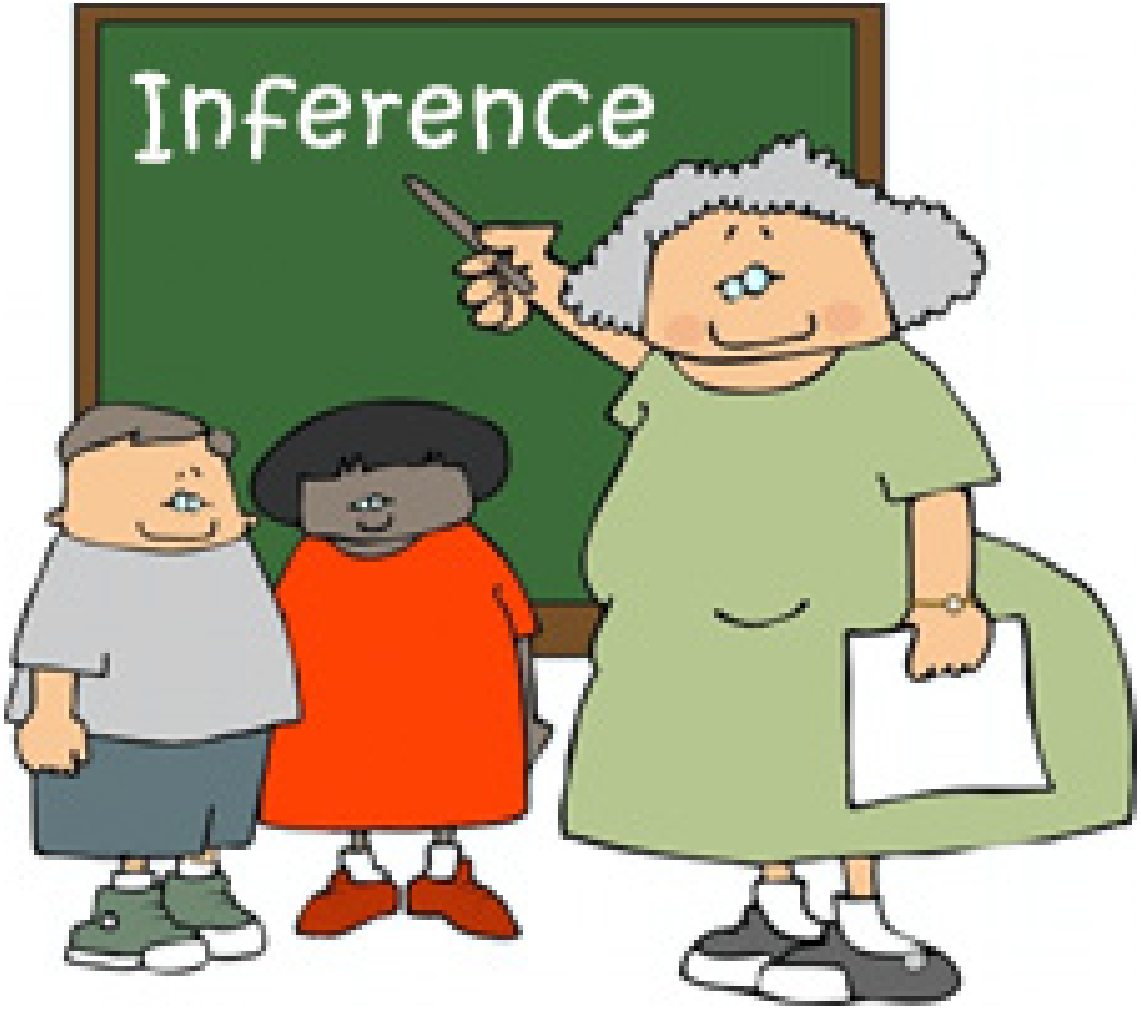

```
1  def process(node: Node) = {
2    node match {
3      case blockNode: BlockTree => {
4        // Process BlockTree node
5      }
6      case importNode: ImportTree => {
7        // Process ImportTree node
8      }
9      // ...
10     case _ => {
11       // Process all other nodes
12     }
13   }
14 }
```

Pattern matching


```
override def reduce(p1: SpliceMap,
                    p2: SpliceMap): SpliceMap = {
  (p1, p2) match {
    case (null, null) => return defaultSpliceMap
    case (_, null) => return p1
    case (null, _) => return p2
    case _ => p1.reduce(p2)
  }
}
```

- Case classes
- Pattern matching in XML
- ...
- unapply

Вывод типов



Вывод типов

В Scala используется Colored Local Type Inference

- Можно опускать указание типов в выражениях
- Можно опускать указание типа возвращаемого значения метода[2]

```
1 val action = new Action()
2 val map = Map.empty[Int, Action]
3 def innerDocument = factory.createDocument()
```

[2] для нерекурсивных методов

Вывод типов

Nota Bene!

- Необходимо указывать типы аргументов метода
- Необходимо указывать тип выражения тогда, когда вывод типов не справляется

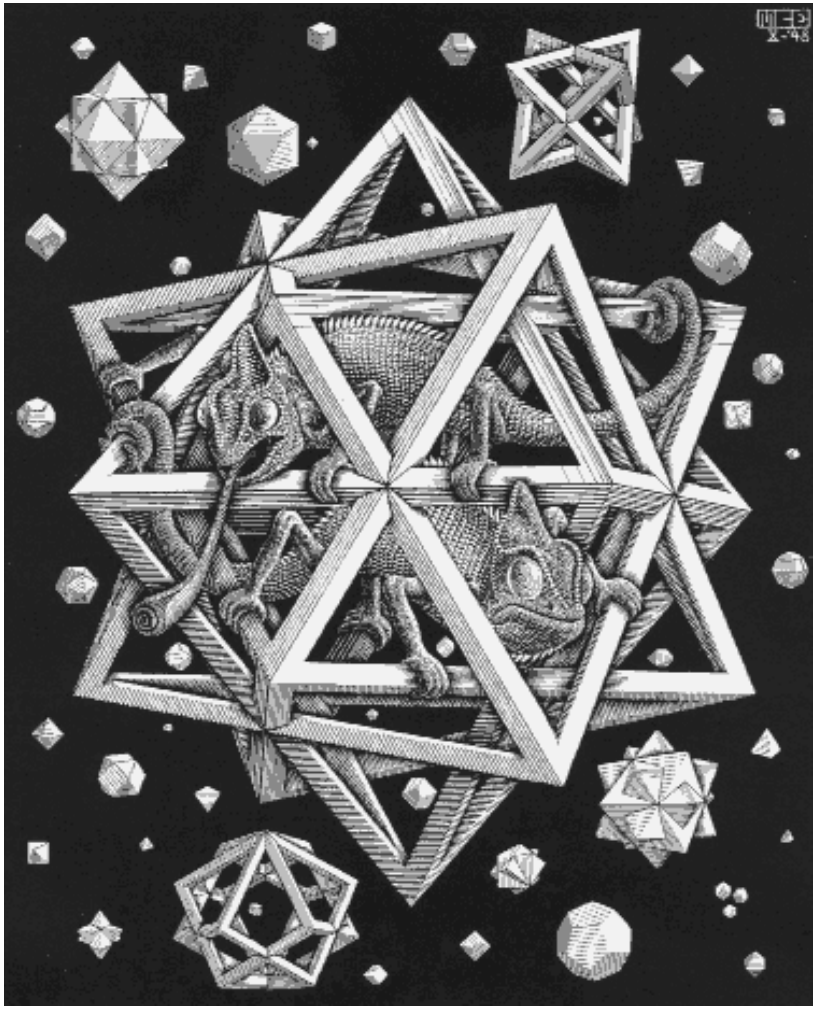
```
1 def getPositions(tree: Tree) = {
2   val start = srcPos.getStartPosition(cut, tree)
3   val end = srcPos.getEndPosition(cut, tree)
4   (new Position(start), new Position(end))
5 }
6
7 @EJB
8 var event: EventProcessorBeanLocal = _
```

Вывод типов


```
1 // ...
2 <T> Collection<T> mergeAll(Collection<T> entities)
3     throws CoreException;
4 // ...
```

```
1 // ...
2 def mergeAll[T](entities: Collection[T]) :
     JCollection[T]= {
3   val result = ...
4   // ...
5   result // scala.collection.immutable.Iterable[T]
6 }
7 // ...
```

Функции высших порядков



Функции высших порядков

Функции высших порядков

- могут принимать в качестве аргументов другие функции
- могут возвращать в качестве результата другие функции
- могут являться значением

```
1 def apply[V,R](f: V => R, v: V): R = f(v)
2
3 4 == apply( (i: Int) => i * i, 2 )
```

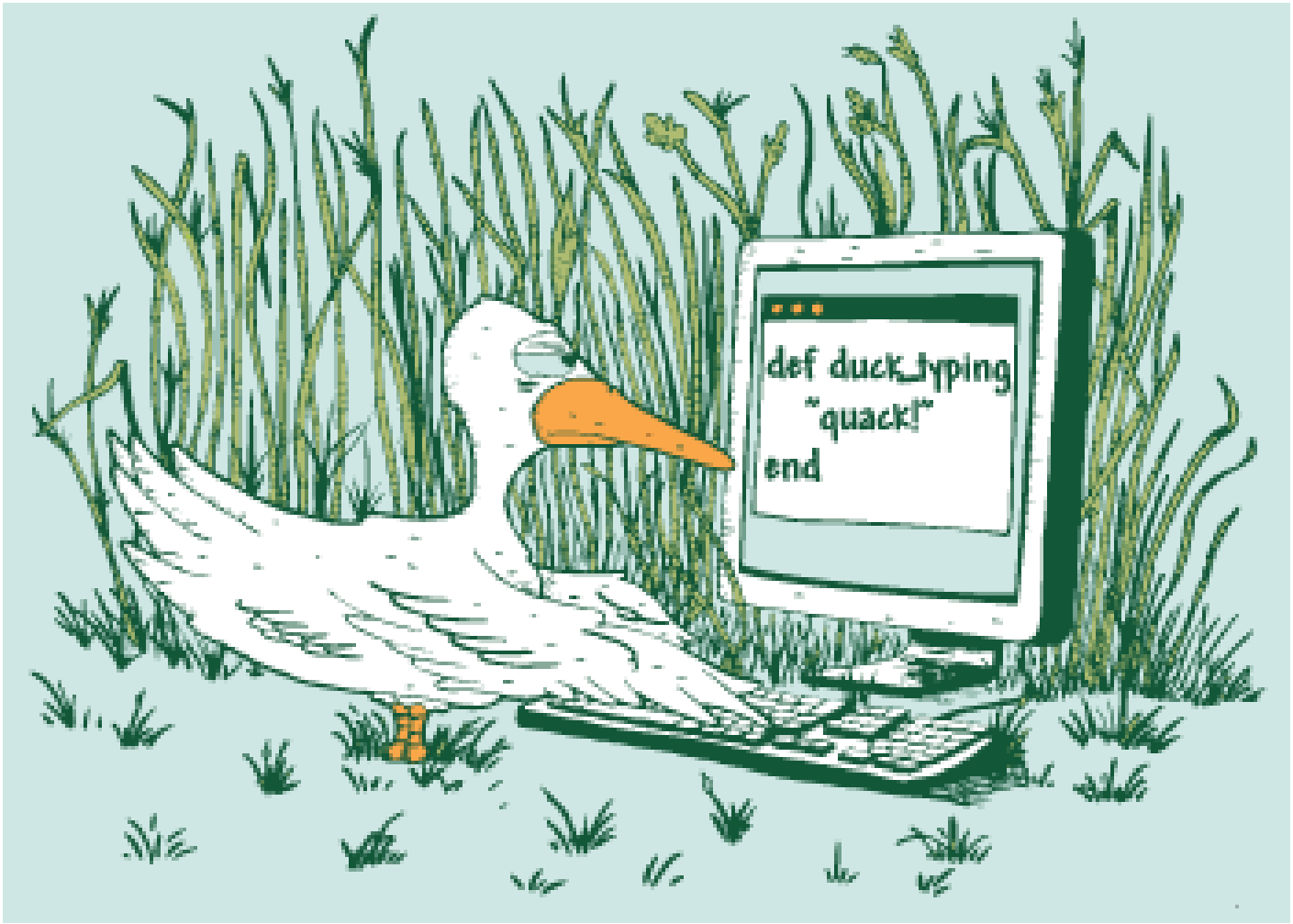
```
1 def apply[V,R](f: _ >: V => _ <: R, v: V): R
```

Функции высших порядков


```
1  def fromActions(
2    actions: List[Action],
3    converters: List[(Action) => CommonGroup]
4  ) = {
5    (new CommonData /: actions)(
6      (data, action) =>
7        data add converters.map(_(action))
8    )
9  }
10
11 val convs = List(summary _, details _)
```

Функции высших порядков позволяют экономить и не плодить лишние сущности

Структурная типизация



Структурная типизация

Отношения между типами задаются неявно, через их структуру

```
1  class Square(side: Double) {
2      def area = side * side
3      def perimeter = side * 4
4  }
5
6  class Circle(radius: Double) {
7      def area = PI * radius * radius
8      def perimeter = 2 * PI * radius
9  }
10
11 type Figure = {
12     def area: Double
13     def perimeter: Double
14 }
```

Структурная типизация

Задача: сгенерировать UUID для сущностей из БД

```
1 def generate(o: AnyRef, id: JInteger) = {
2     new java.util.UUID(
3         o.getClass.getCanonicalName.hashCode,
4         if (id == null) -1 else id.longValue
5     )
6 }
```

Вариант №1: дополнительный интерфейс

```
1 trait Idable {
2     def getId: JInteger
3 }
4
5 def generateFromId(o: Idable) = {
6     generate(o, o.getId)
7 }
```

Структурная типизация

Вариант №2: описать соответствующий структурный тип

```
1 def generateFromId[A <: AnyRef {def getId() :
    JInteger}](o: A) = {
2     generate(o, o.getId)
3 }
```

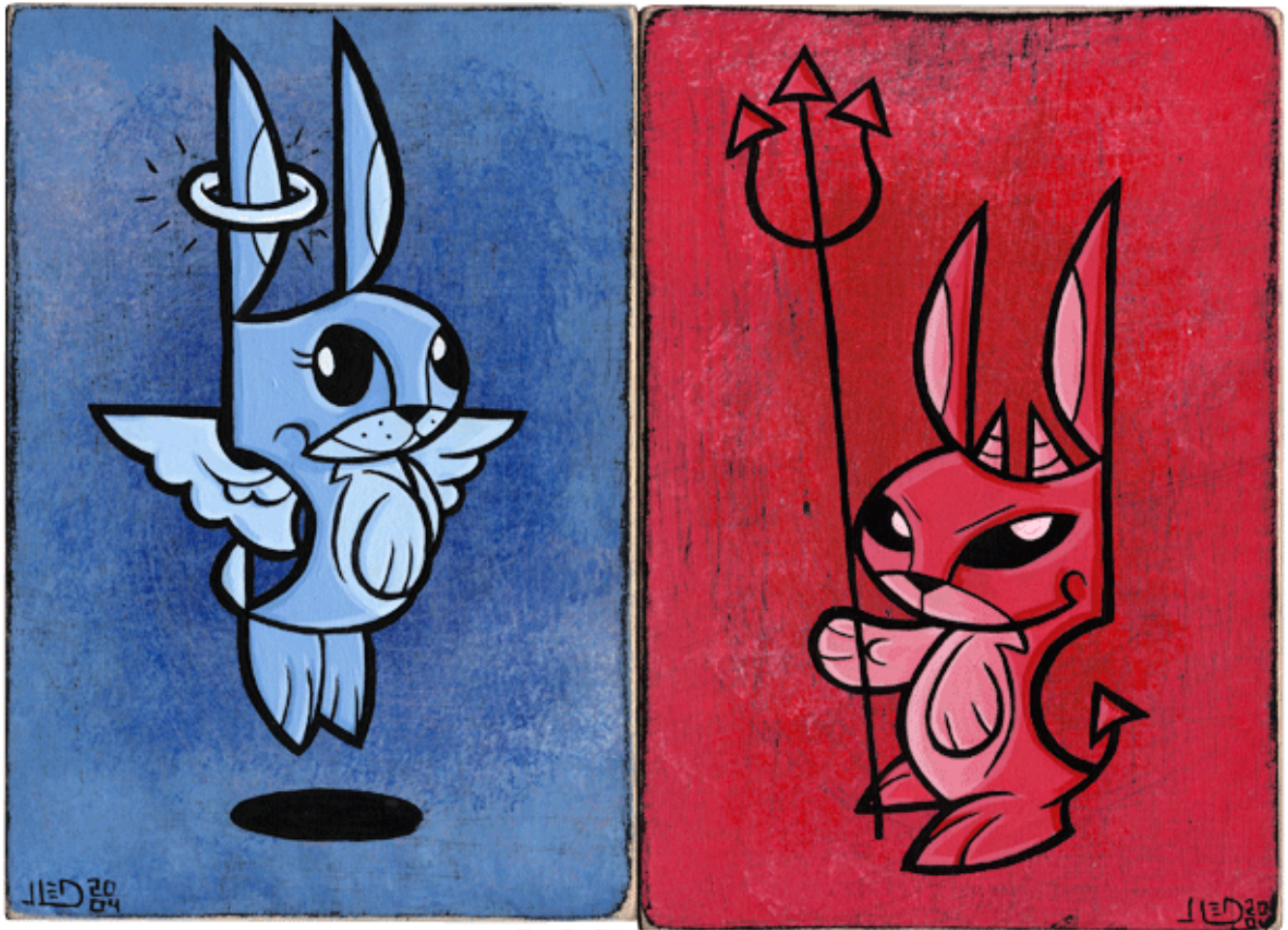
Преимущества

- Лаконичность
- Выразительность

Недостатки

- Дублирование кода
- Сложность реализации
- Ограничения в работе со структурными типами

И все же...



И все же...

Миф или реальность?

Реальность (причем чем дальше, тем реальнее)

Стоит ли мне использовать Scala?

Да, если Вы

- готовы потратить время на ее изучение
- не боитесь преодолевать трудности
- понимаете, что полностью избавиться от Java не получится

Какие будут вопросики?


```
1 def rate(who: Person) = {
2   (who.currentMood match {
3     case Bad => -1.0
4     case IDunno => random
5     case Good => 1.0
6   }) * normalizeCoeff
7 }
8
9 apply(rate _, self)
```