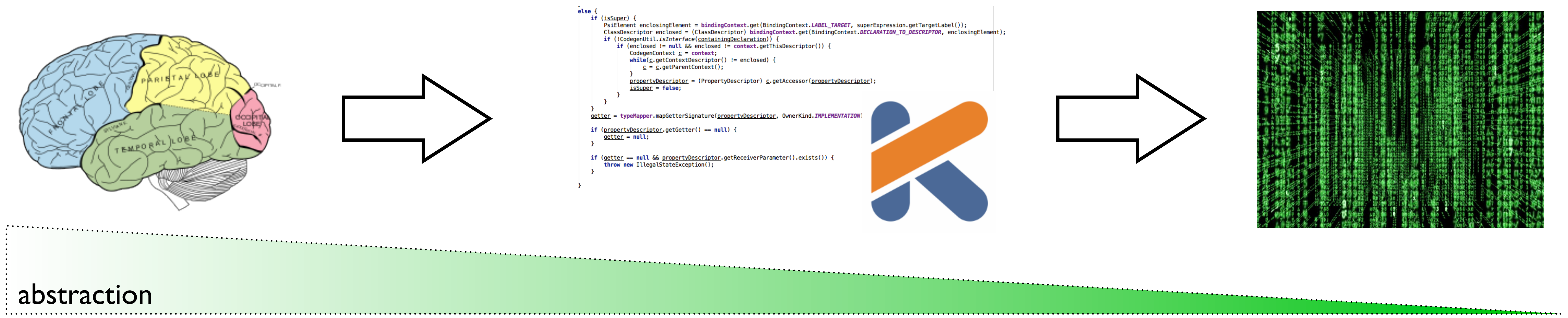
Kotlin: How Things Work

Practical Aspects of JVM Language Implementation

Andrey Breslav

Why you should care

- Language is an abstraction
  - ➡ thus it leaks



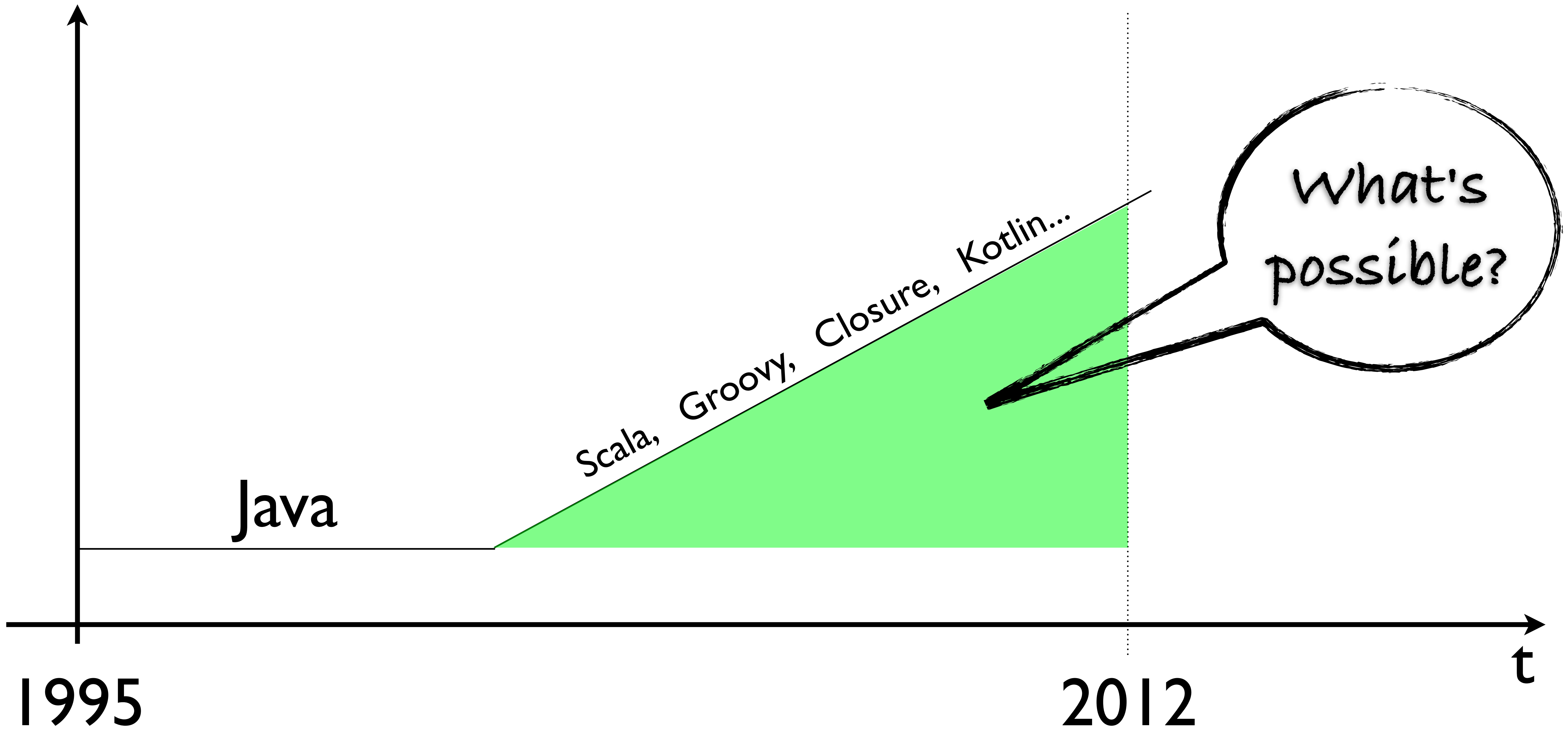
```
else {
    if (isSuper) {
        PsiElement enclosingElement = bindingContext.get(BindingContext.LABEL_TARGET, superExpression.getTargetLabel());
        ClassDescriptor enclosed = (ClassDescriptor) bindingContext.get(BindingContext.DECLARATION_TO_DESCRIPTOR, enclosingElement);
        if (!CodegenUtil.isInterface(containingDeclaration)) {
            if (enclosed != null && enclosed != context.getThisDescriptor()) {
                CodegenContext c = context;
                while(c.getContextDescriptor() != enclosed) {
                    c = c.getParentContext();
                }
                propertyDescriptor = (PropertyDescriptor) c.getAccessor(propertyDescriptor);
                isSuper = false;
            }
        }
    }
    getter = typeMapper.mapGetterSignature(propertyDescriptor, OwnerKind.IMPLEMENTATION)

    if (propertyDescriptor.getGetter() == null) {
        getter = null;
    }

    if (getter == null && propertyDescriptor.getReceiverParameter().exists()) {
        throw new IllegalStateException();
    }

}
```

- When something weird happens
  - ➡ you need to "see through the Matrix"

TETRIS IS WAY TOO EASY

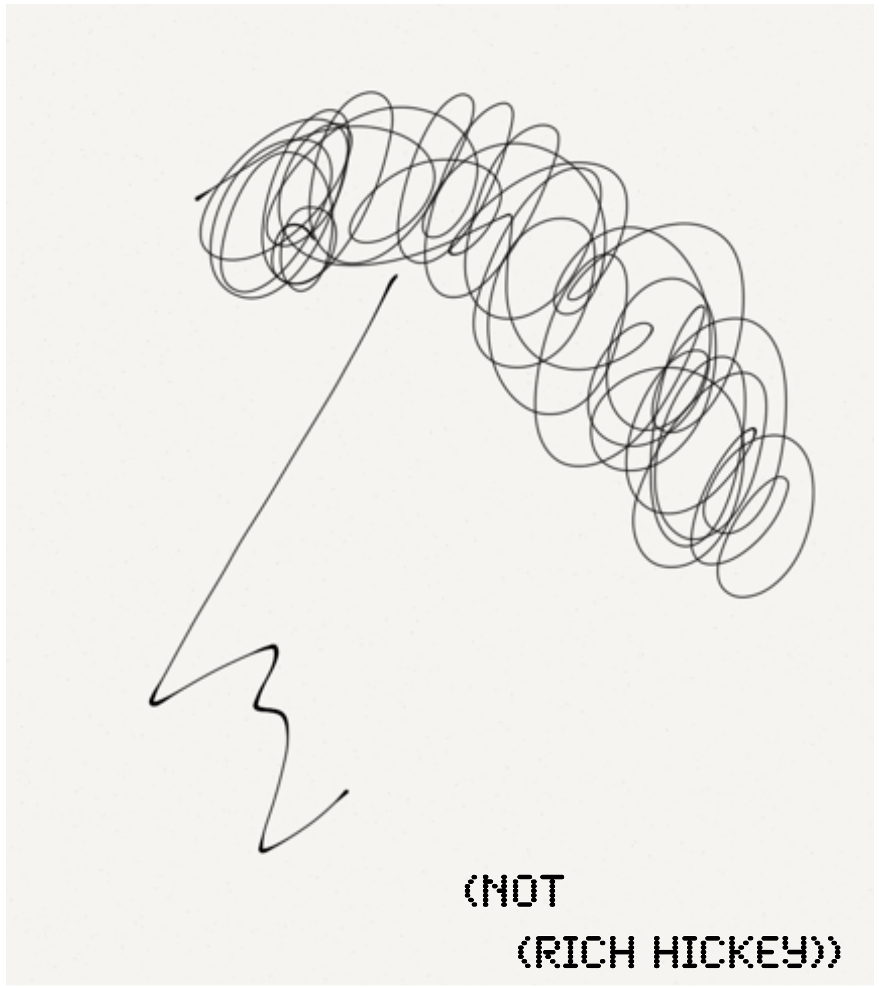
WHEN PLAYED THIS WAY

JVM & Its Languages



About Me

- Project lead of Kotlin
  - ➡ at JetBrains since 2010
- EG member of JSR-335
  - ➡ Project **Lambda**
  - ➡ at Java Community Process



Kotlin

Modern Language for **Industry**

- Smart compiler
  - ➡ Less boilerplate
- Flexible abstractions
  - ➡ Powerful libraries

- Static typing
- Readability
- Tool support
- **Interoperability**

Outline

- Quick intro to the Matrix
- Constructors
- Default Arguments
- Extensions
- Collections

- More (maybe)...

$ javap -c hello.Matrix

```
Compiled from "Matrix.java"
public class hello.Matrix extends java.lang.Object{
public hello.Matrix();
  Code:
   0: aload_0
   1: invokespecial    #1; //Method java/lang/Object."<init>":()V
   4: return

public static void main(java.lang.String[]);
  Code:
   0: getstatic  #2; //Field java/lang/System.out:Ljava/io/PrintStream;
   3: ldc   #3; //String Hello, Matrix!
   5: invokevirtual    #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
   8: return

}
```

How many constructors?

```java
public class Hello {
    String name;

    void sayHello() {
        System.out.println("Hi, I am " + name);
    }
}
```

How many constructors?

```
$ javap Hello
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
    java.lang.String name;
    public Hello();
    void sayHello();
}
```

Is it empty?

How many constructors?

```
$ javap Hello
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
    java.lang.String name;
    public Hello();
    void sayHello();
}
```

It's there!

Is it empty?

Default constructor

```
$ javap -c Hello
```

Default constructor

```
$ javap -c Hello
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
java.lang.String name;

public Hello();
  Code:
   0:aload_0
   1:invokespecial java/lang/Object."<init>":()V
   4:return
```

Not empty!

Default constructor

```
$ javap -c Hello
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
java.lang.String name;

public Hello();
  Code:
   0:aload_0
   1:invokespecial java/lang/Object."<init>":()V
   4:return
```

super();

Not empty!

Kotlin Constructor Demo

- Primary constructors
- Properties
- Default arguments

https://github.com/abreslav/javaone_2012/tree/master/constructors

Default Arguments in Scala

- Live Demo

https://github.com/abreslav/javaone_2012/tree/master/scala-default-args

Extensions

- Live Demo

https://github.com/abreslav/javaone_2012/tree/master/extensions

Collections

How Data-Compatible is Your Language?

Collections & Variance

Java:

```
static <T extends Comparable<? super T>>
        T min(List<? extends T> ts) {
        // ...
}
```

Kotlin:

```
fun <T: Comparable<T>> min(ts: List<T>): T {
    // ...
}
```

Declaration-Site Variance

```
trait Comparable<in T> {
    fun compare(a: T, b: T): Int
}

trait List<out T> {
    fun get(index: Int): T
}

trait MutableList<T>: List<T> {
    fun set(index: Int, value: T)
}
```

Translation

```
trait List<out T> {
    fun get(index: Int): T
}
```

$\rightarrow \varnothing$

```
trait MutableList<T>: List<T> {
    fun set(index: Int, value: T)
}
```

$\rightarrow$ java.util.List

```
fun <T> copy(from: List<T>, to: MutableList<T>)
```

$\downarrow$

```
<T> void copy(j.u.List<T> from, j.u.List<T> to)
```

JetBRAINS

Translation: Inheritance

```
class MyList: List<String> {
    override fun get(index: Int): String {
        return "..."
    }
}
```

↓

```
class MyList extends List<String> {
    String get(int i) {
        return "..."
    }

    void set(int i, String s) {
        throw new UnsupportedOperationException();
    }
}
```

Translation: Variance

```
fun <T> copy(from: List<T>, to: MutableList<out T>)
```

⬇

```
<T> void copy(
    j.u.List<? extends T> from, j.u.List<? super T> to
)
```

Calling Java From Kotlin

```
<T> j.u.List<T> copy(j.u.List<T> from)
```

⬇

```
fun <T> copy(from: List<T>): MutableList<T>
```

Summary

- Subtle implementation details
  - ➡ Subtle implications
- Looks cool $\not\Rightarrow$ Works Cool
- Languages are about Tradeoffs

Kotlin Resources

- Docs: http://kotlin.jetbrains.org
- Demo: http://kotlin-demo.jetbrains.com
- Code: http://github.com/jetbrains/kotlin
- Twitter:
  - @project_kotlin
  - @abreslav