Effective Java
with Groovy & Kotlin
How Languages Influence Adoption of Good Practices

CONF42

Naresha K

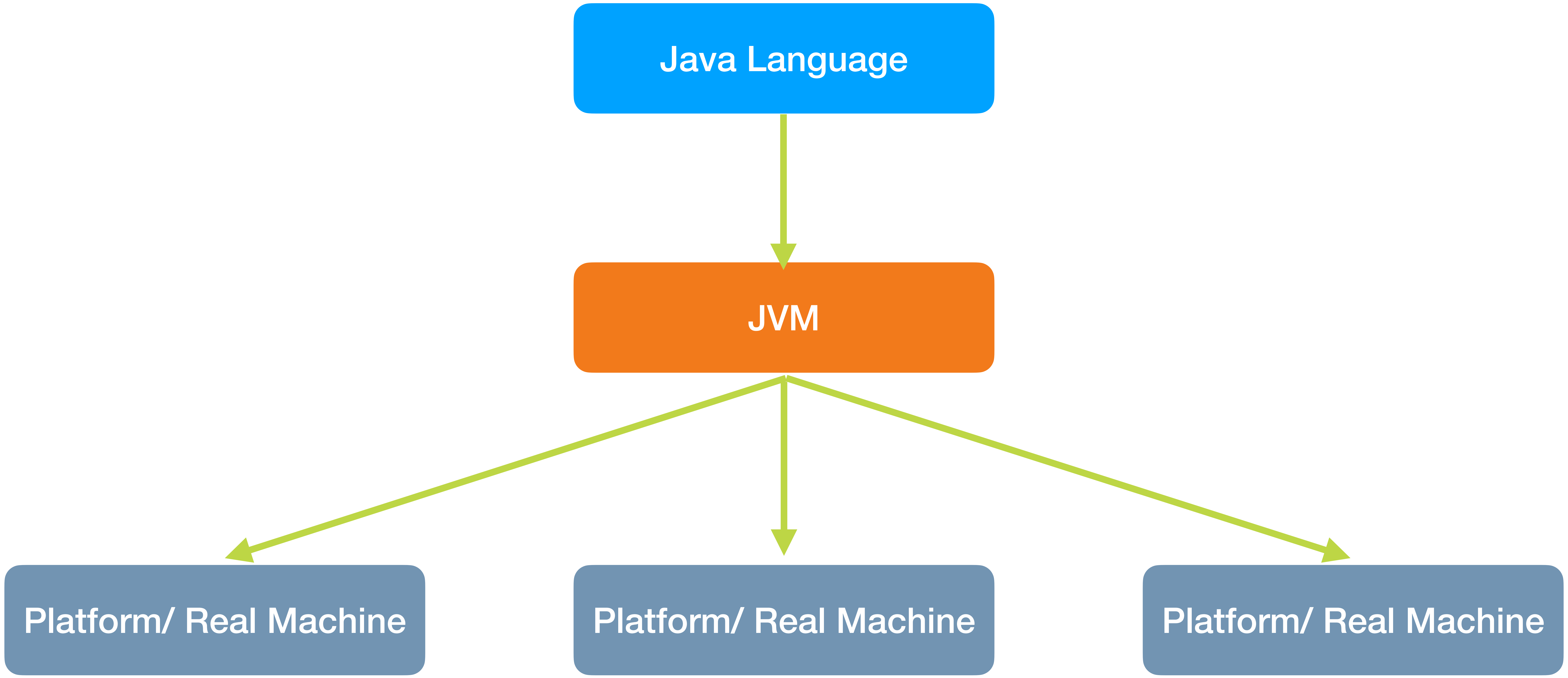
@naresha_k

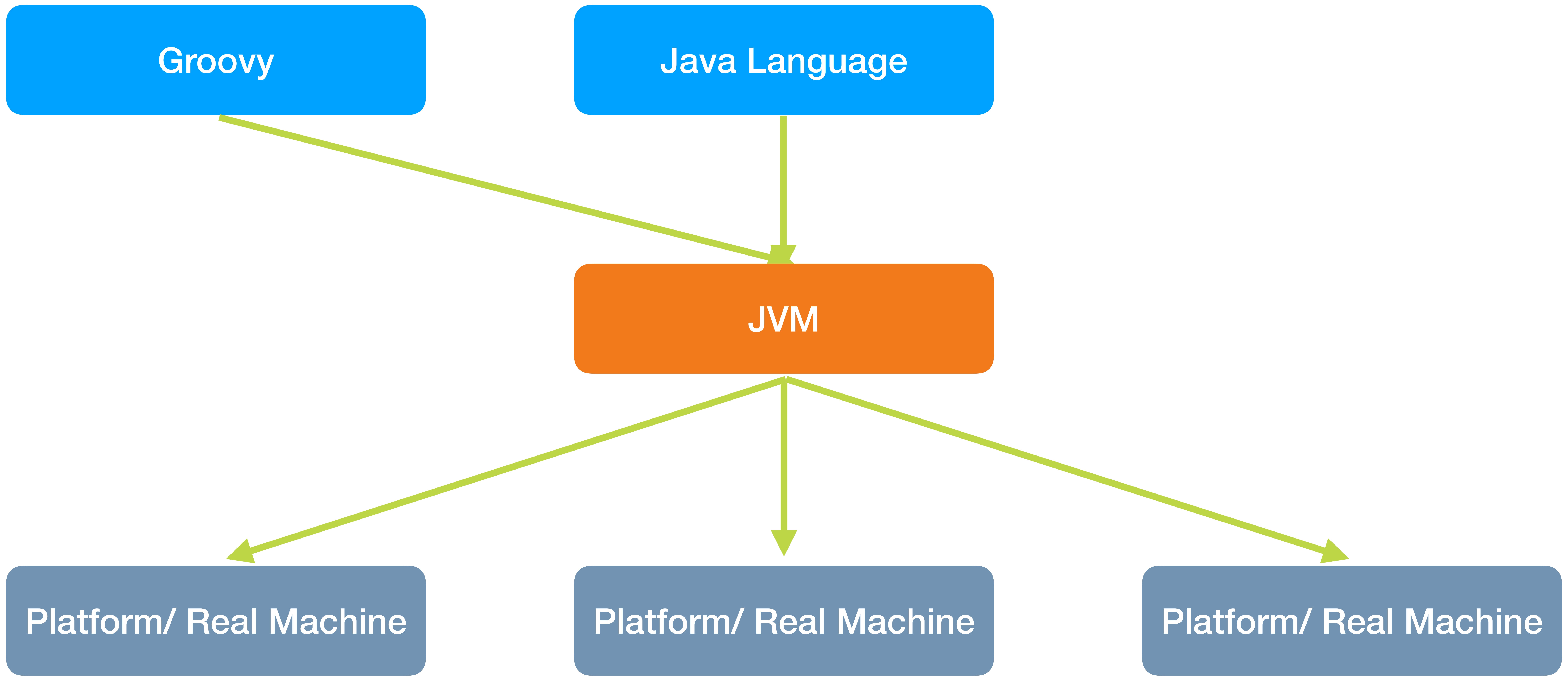
https://blog.nareshak.com/

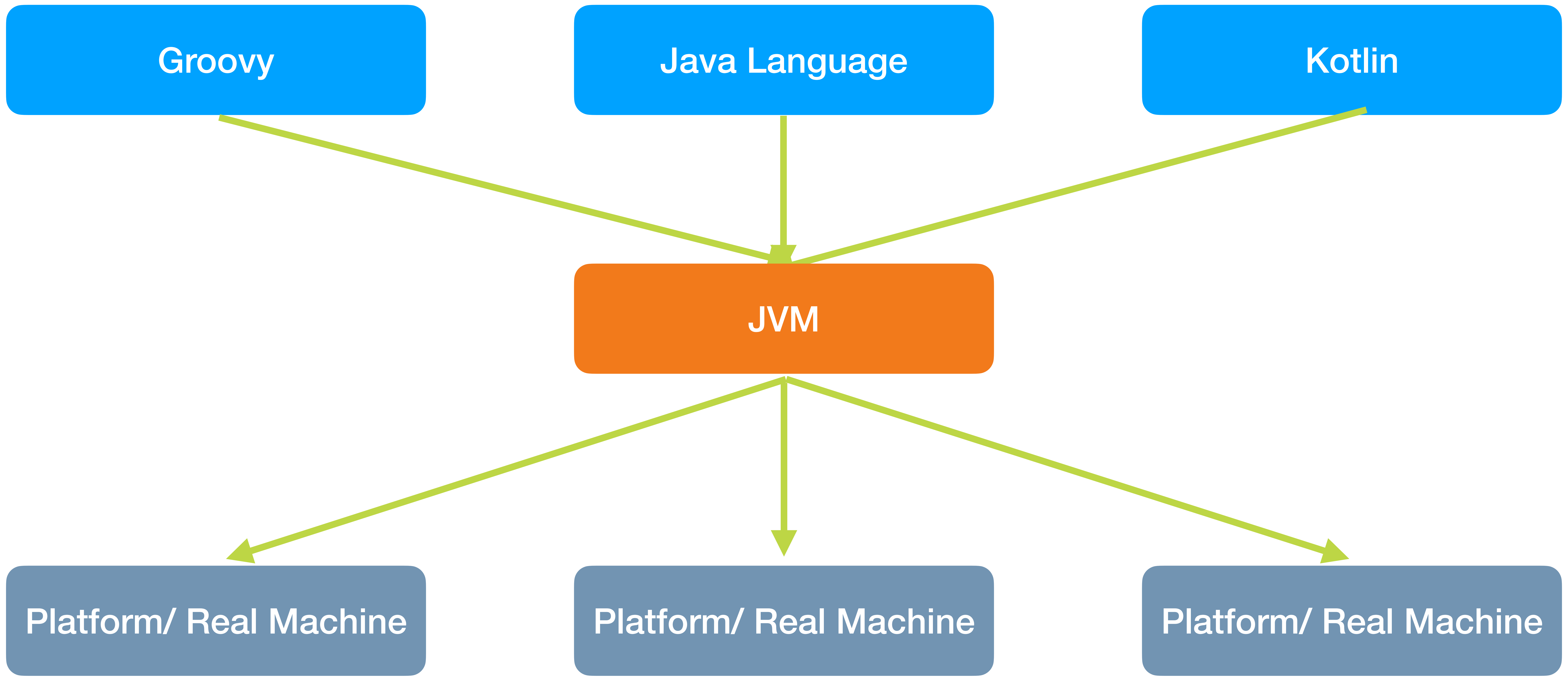
About me

Developer, Architect & Tech Excellence Coach

Founder & Organiser Bangalore Groovy User Group

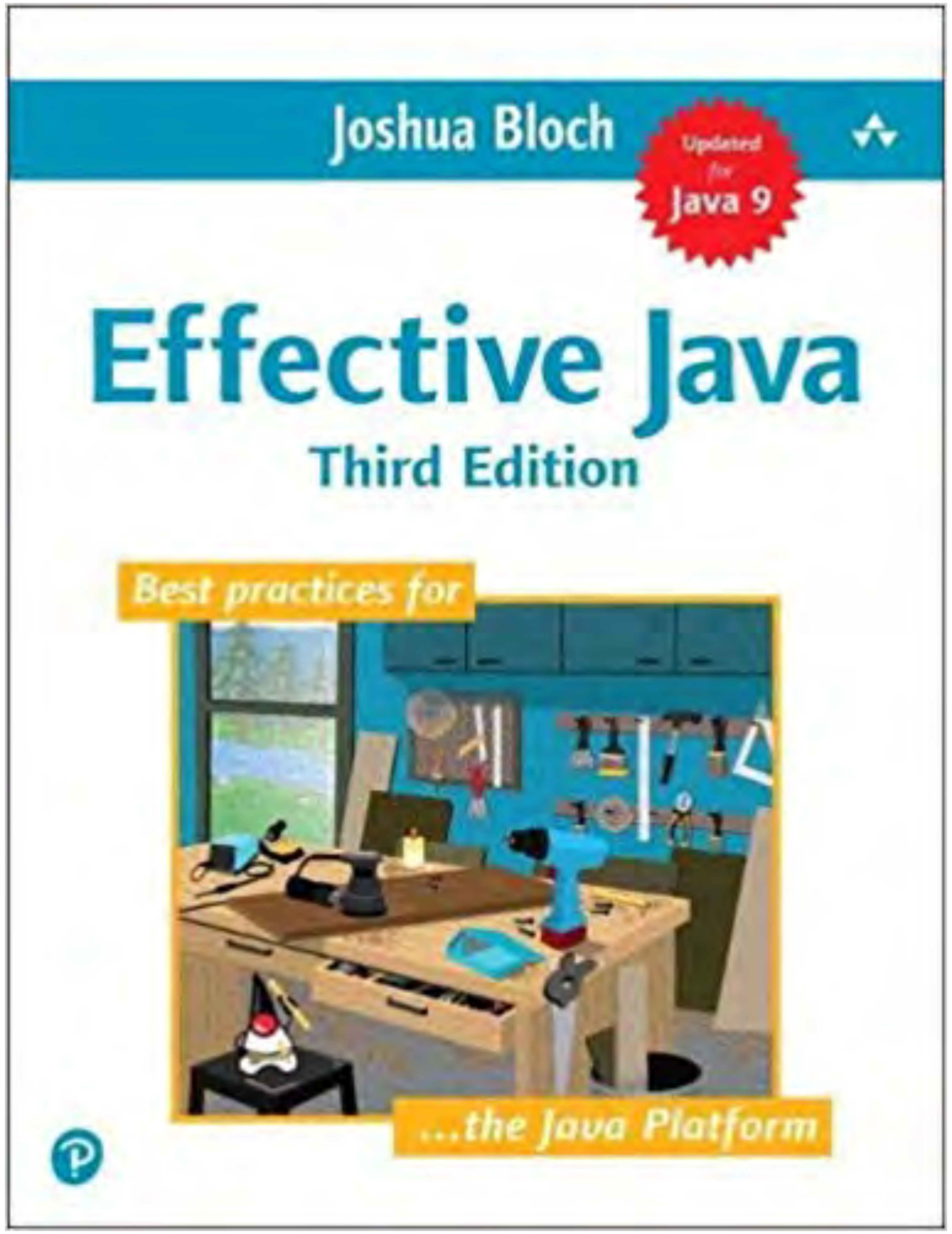

Java Language
JVM
Platform/ Real Machine
Platform/ Real Machine
Platform/ Real Machine


Groovy
Java Language
JVM
Platform/ Real Machine
Platform/ Real Machine
Platform/ Real Machine


Groovy
Java Language
Kotlin
JVM
Platform/ Real Machine
Platform/ Real Machine
Platform/ Real Machine

Most people talk about Java the language, and this may sound odd coming from me, but I could hardly care less. At the core of the Java ecosystem is the JVM.

- James Gosling,
Creator of the Java Programming Language(2011, TheServerSide)


Joshua Bloch
Updated for Java 9
Effective Java
Third Edition
Best practices for
...the Java Platform

http://groovy-lang.org/

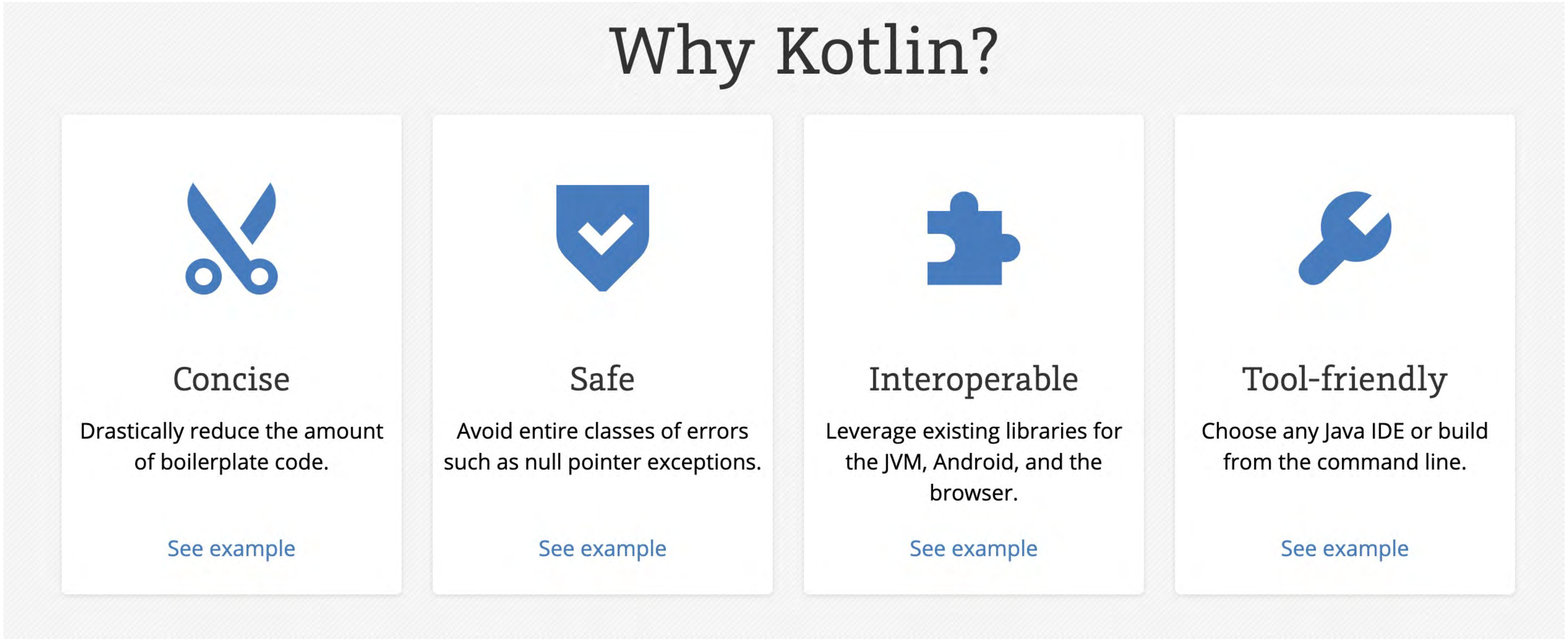
initial idea was to make a little dynamic language which compiles directly to Java classes and provides all the nice (alleged) productivity benefits

- James Strachan

http://radio-weblogs.com/0112098/2003/08/29.html

Kotlin

https://kotlinlang.org/



Why Kotlin?

Concise

Drastically reduce the amount of boilerplate code.

See example

Safe

Avoid entire classes of errors such as null pointer exceptions.

See example

Interoperable

Leverage existing libraries for the JVM, Android, and the browser.

See example

Tool-friendly

Choose any Java IDE or build from the command line.

See example

https://kotlinlang.org/

Know your guides

- Context/ the Problem
- What does “Effective Java” say?
- The traps
- Idiomatic Solution
- Lessons Learned

Java Groovy Kotlin

Tool Wisdom

START HERE

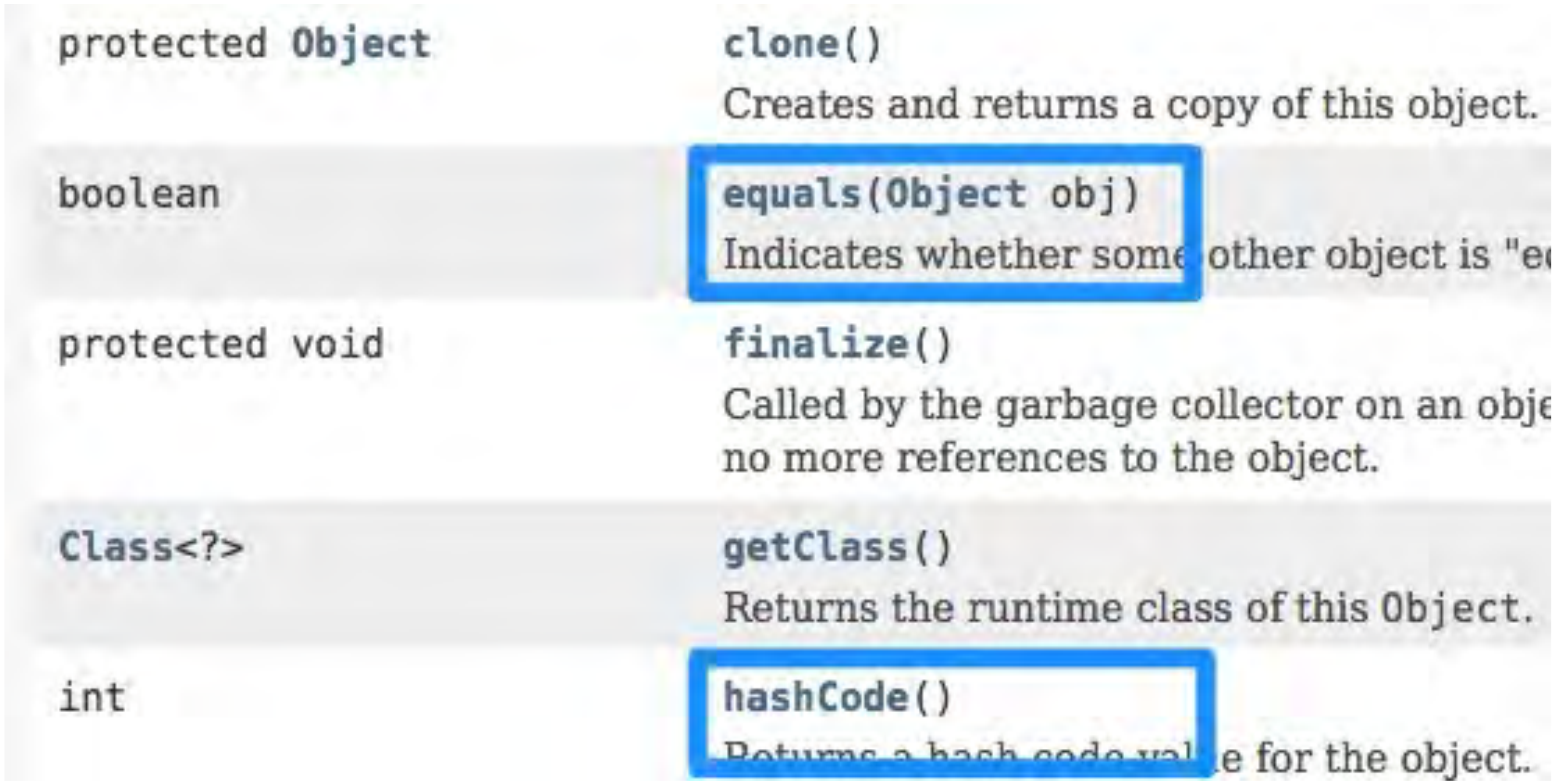
```groovy
class Product {
    String sku
    String description
    BigDecimal price
}
```

```
Product book1 = new Product(
  sku: 'P101',
  description: 'Effective Java’,
  price: 40.0)
Product book2 = new Product(
  sku: 'P101',
  description: 'Effective Java’,
  price: 40.0)
println book1 == book2
```

false

```groovy
Product book1 = new Product(
  sku: 'P101',
  description: 'Effective Java’,
  price: 40.0)
Product book2 = new Product(
  sku: 'P101',
  description: 'Effective Java’,
  price: 40.0)
def stock = [:]
stock[book1] = 100
println stock[book2]
```

null



protected Object	clone() Creates and returns a copy of this object.
boolean	equals(Object obj) Indicates whether some other object is "e
protected void	finalize() Called by the garbage collector on an obje no more references to the object.
Class<?>	getClass() Returns the runtime class of this Object.
int	hashCode() Returns a hash code value for the object.

#10: Obey the general contract when overriding equals

#11: Always override hashCode when you override equals

EqualsBuilder	Assists in implementing `Object.equals(Object)` methods.
HashCodeBuilder	Assists in implementing `Object.hashCode()` methods.

```
public boolean equals(Object obj) {
  if (obj == null) { return false; }
  if (obj == this) { return true; }
  if (obj.getClass() != getClass()) {
    return false;
  }
  MyClass rhs = (MyClass) obj;
  return new EqualsBuilder()
                .appendSuper(super.equals(obj))
                .append(field1, rhs.field1)
                .append(field2, rhs.field2)
                .append(field3, rhs.field3)
                .isEquals();
 }
```

```
class Product {
    String sku
    String description
    BigDecimal price
}
```

```
class Product {
    String sku
    String description
    BigDecimal price
    LocalDate dateOfManufacture
}
```

```
class Product {
    String sku
    String description
    BigDecimal price
    LocalDate dateOfManufacture
    LocalDate dateOfExpiry
}
```

```
@EqualsAndHashCode
class Product {
    String sku
    String description
    BigDecimal price
}
```

```
Product book1 = new Product(
  sku: 'P101',
  description: 'Effective Java’,
  price: 40.0)
Product book2 = new Product(
  sku: 'P101',
  description: 'Effective Java’,
  price: 40.0)
println book1 == book2
```

true

```groovy
Product book1 = new Product(
  sku: 'P101',
  description: 'Effective Java’,
  price: 40.0)
Product book2 = new Product(
  sku: 'P101',
  description: 'Effective Java’,
  price: 40.0)
def stock = [:]
stock[book1] = 100
println stock[book2]
```

100

```groovy
@EqualsAndHashCode(includes = 'id')
class Product {
    Long id
    String sku
    String description
    BigDecimal price
}
```

```
@EqualsAndHashCode(includes = 'id')
class Product {
    Long id
    String sku
    String description
    BigDecimal price
}
```

```
@EqualsAndHashCode(includes = ‘sku')
class Product {
    Long id
    String sku
    String description
    BigDecimal price
}
```

```kotlin
class Product(val sku: String, val description: String,
   val price: BigDecimal)

fun main() {
    val book1 = Product("P101", "Effective Java", BigDecimal("35.08"))
    val book2 = Product("P101", "Effective Java", BigDecimal("35.08"))
    println(book1 == book2)

    val stock = mapOf(book1 to 100)
    println(stock[book2])
}
```

false
null

Kotlin

```
data class Product(val sku: String, val description: String, val price:
BigDecimal)

fun main() {
    val book1 = Product("P101", "Effective Java", BigDecimal("35.08"))
    val book2 = Product("P101", "Effective Java", BigDecimal("35.08"))
    println(book1 == book2)

    val stock = mapOf(book1 to 100)
    println(stock[book2])
}
```

```
true
100
```

Kotlin

```kotlin
data class Product(val sku: String, val description: String,
  val price: BigDecimal, var id: Long?)

fun main() {
    val book1 = Product("P101", "Effective Java", BigDecimal("35.08"), null)
    val book2 = Product("P101", "Effective Java", BigDecimal("35.08"), 1L)
    println(book1 == book2)

    val stock = mapOf(book1 to 100)
    println(stock[book2])
}
```

```
false
null
```

Kotlin

```kotlin
data class Product(val sku: String, val description: String, val price:
BigDecimal) {
    var id: Long? = null
}

fun main() {
    val book1 = Product("P101", "Effective Java", BigDecimal("35.08"))
    val book2 = Product("P101", "Effective Java", BigDecimal("35.08"))
    book2.id = 1L
    println(book1 == book2)

    val stock = mapOf(book1 to 100)
    println(stock[book2])
}
```

```
true
100
```

AST Transformation

Language / Compiler

Single point of representation of any knowledge

```java
List<String> languages = List.of("Java", "Groovy", "Kotlin");
for (int i = 0; i < languages.size(); i++) {
    System.out.println(languages.get(i));
}
```

```java
for (String language : languages) {
    System.out.println(language);
}
```

#58: Prefer for-each loops to traditional for loops

Kotlin

```
numbers.forEach {
    println(it)
}
```

```
10
20
30
40
```

```
println(numbers.map {
    it * 2
})
```

```
[20, 40, 60, 80]
```

```
println(numbers.fold(0)
   { acc, number -> acc + number })
```

```
100
```

```
numbers.each { number ->
    println number
}
```

```
10
20
30
40
```

```
println numbers.collect { number ->
    number * 2
}
```

```
[20, 40, 60, 80]
```

```
println numbers.inject(0,
        { result, number -> result + number }
)
```

```
100
```

Closures/ Higher Order Functions

Favor Internal iterators to external iterators

Minimise the moving parts

Million Dollar Effort

Million Dollar Effort

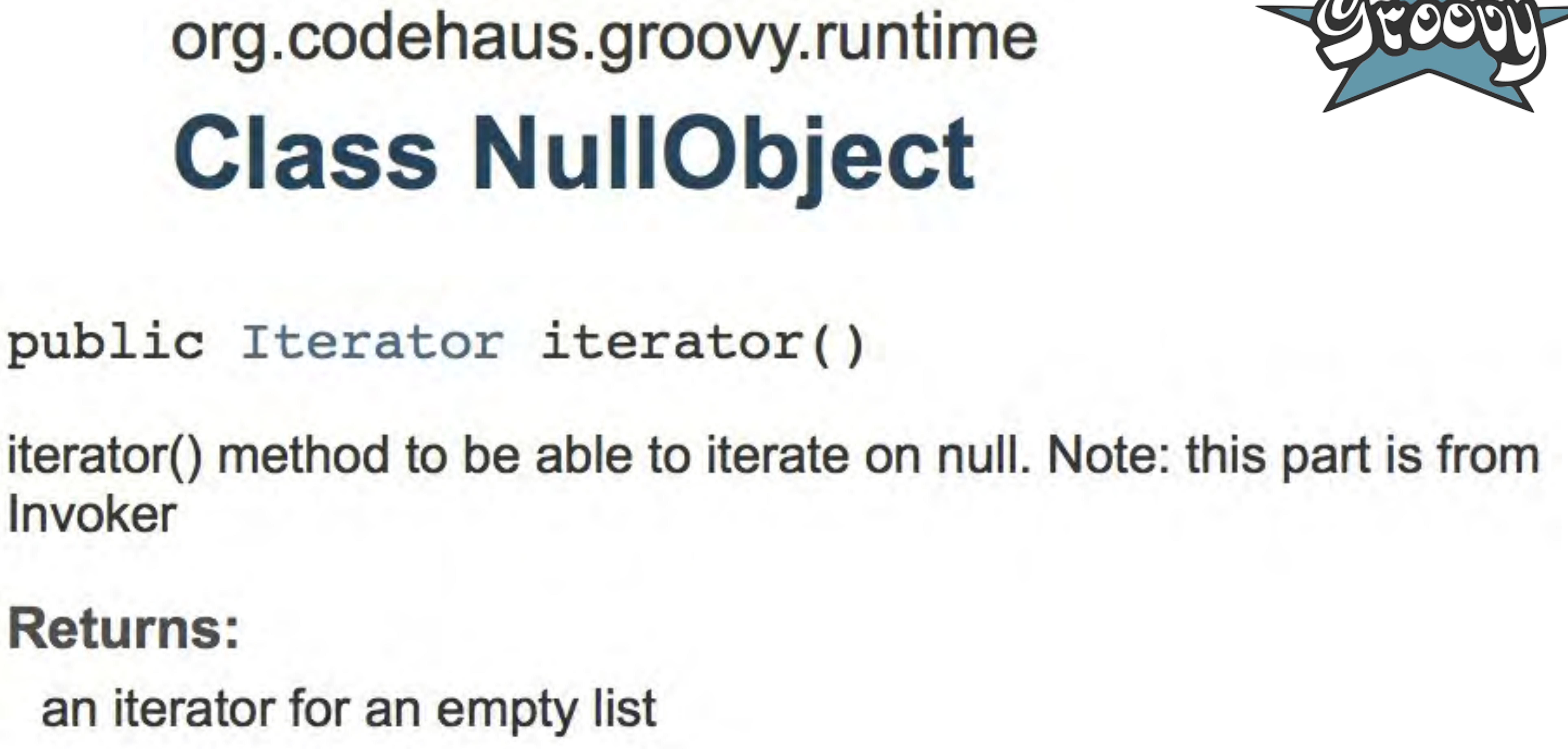
null check

```java
List<Speaker> getSpeakers(String conference) {
    return null;
}
List<Speaker> speakers = getSpeakers("CONF42 JAVA");
if (speakers != null) {
    //...
}
```

#54: Return empty arrays or collections, not nulls

```
println getSpeakers(‘CONF42 JAVA')
        .collect { it.firstName }
println getSpeakers(‘CONF42 JAVA')
        .findAll { it.firstName.length() > 5 }
```

```
println getSpeakers(‘CONF42 JAVA')
        .collect { it.firstName } // []
println getSpeakers(‘CONF42 JAVA')
        .findAll { it.firstName.length() > 5 } // []
```



org.codehaus.groovy.runtime

Class NullObject

```
public Iterator iterator()
```

iterator() method to be able to iterate on null. Note: this part is from Invoker

Returns:

an iterator for an empty list

```kotlin
fun getSpeakers(conference: String): List<Speaker> {
    return null
}
```

Kotlin

```kotlin
fun getSpeakers(conference: String): List<Speaker> {
    return null
}
```

Null can not be a value of a non-null type List<Speaker>

```kotlin
fun getSpeakers(conference: String): List<Speaker> {
    return null
}
```

Null can not be a value of a non-null type List<Speaker>

```kotlin
fun getSpeakers(conference: String): List<Speaker>? {
    return null
}
```

NullObject Pattern

Type system

No boilerplate code - $$$

Life is too short for null checks!

https://www.flickr.com/photos/38080114@N07/8594601982/

#17: Minimize Mutability

Rules to make a class immutable

1. Don’t provide any mutators
2. Ensure that the class can’t be extended
3. Make all fields final
4. Make all fields private
5. Ensure exclusive access to any mutable components

```
class ImmutableClass{
    private final def field1
    private final def field2
    //...
    private final def field10

    public ImmutableClass(f1, f2,… f10){
        //initialization
    }
}
```

```
import groovy.transform.Immutable

@Immutable
class Rectangle {
    int length
    int breadth
}
```

```
def r = new Rectangle(length: 10, breadth: 5)
println r // Rectangle(10, 5)
```

```
public final class Rectangle extends java.lang.Object
                implements groovy.lang.GroovyObject {

    private final int length
    private final int breadth

    public Rectangle(int length, int breadth) {
        //
    }

    public Rectangle(java.util.Map args) {
    }

    public Rectangle() {
        this([:])
    }
}
```

```
new Rectangle(length: 10, breadth: 5)
```

```kotlin
data class Rectangle(val length: Int, val breadth: Int)
```

Kotlin

```java
public final class Rectangle {
   private final int length;
   private final int breadth;

   public final int getLength() {
      return this.length;
   }

   public final int getBreadth() {
      return this.breadth;
   }

   public Rectangle(int length, int breadth) {
      this.length = length;
      this.breadth = breadth;
   }
  // more code
}
```

Kotlin

```
data class Rectangle(val length: Int, val breadth: Int)

fun main() {
    val rectangle = Rectangle(20, 10)
    val r2 = Rectangle(length = 20, breadth = 10)
}
```

```
public static final void main() {
   Rectangle rectangle = new Rectangle(20, 10);
   Rectangle r2 = new Rectangle(20, 10);
 }
```

AST Transformation

Syntactic Sugar

Readability Matters

#18: Favour composition over inheritance

```
def ph = ['919812312345', '4512341234', ‘19252199916']
 as PhoneNumbers
println ph.find { it == '19252199916'}
println ph.findAll { it.endsWith('4') }
```

```groovy
def ph = ['919812312345', '4512341234', '19252199916']
 as PhoneNumbers
println ph.find { it == '19252199916'}
println ph.findAll { it.endsWith('4') }
println ph.indianNumbers()
```

```
class PhoneNumbers extends ArrayList {

}
```

```
class PhoneNumbers {
    private @Delegate List phoneNumbers

    PhoneNumbers(numbers) {
        phoneNumbers = numbers
    }

    def indianNumbers() {
        phoneNumbers.findAll { it.startsWith('91') }
    }
}
```

```
class PhoneNumbers(val list: List<String>) :
        List<String> by list {

    fun indianNumbers(): List<String> {
        return list.filter { it.startsWith("91") }
    }
}
```

```
trait CanSing {
 def sing() {
  println "Singing"
 }
}
```

```
trait CanDance {
 def dance() {
  println "Dancing"
 }
}
```

```
class Person implements CanSing, CanDance {}

Person reema = new Person()
reema.sing()
reema.dance()
```

https://www.slideshare.net/nareshak/designing-with-groovy-traits-gr8conf-india

AST Transformation

Simplify

Take Aways

Some of the ‘Effective Java’ already built into the languages

Favour compiler generated code to IDE generated code

Don’t fall into the trap of copying the Java implementation

Programming languages can reduce friction to implement good practices

There could be multiple right solutions. Choose what fits your context

The way we code is influenced not just by the language we code in, but also by the languages we know.

Happy Coding. Thank You