The dangerous side of Kotlin

bol.com

Bart Enkelaar

Site Reliability Engineer

@BartEnkelaar

benkelaar@bol.com


bol.com

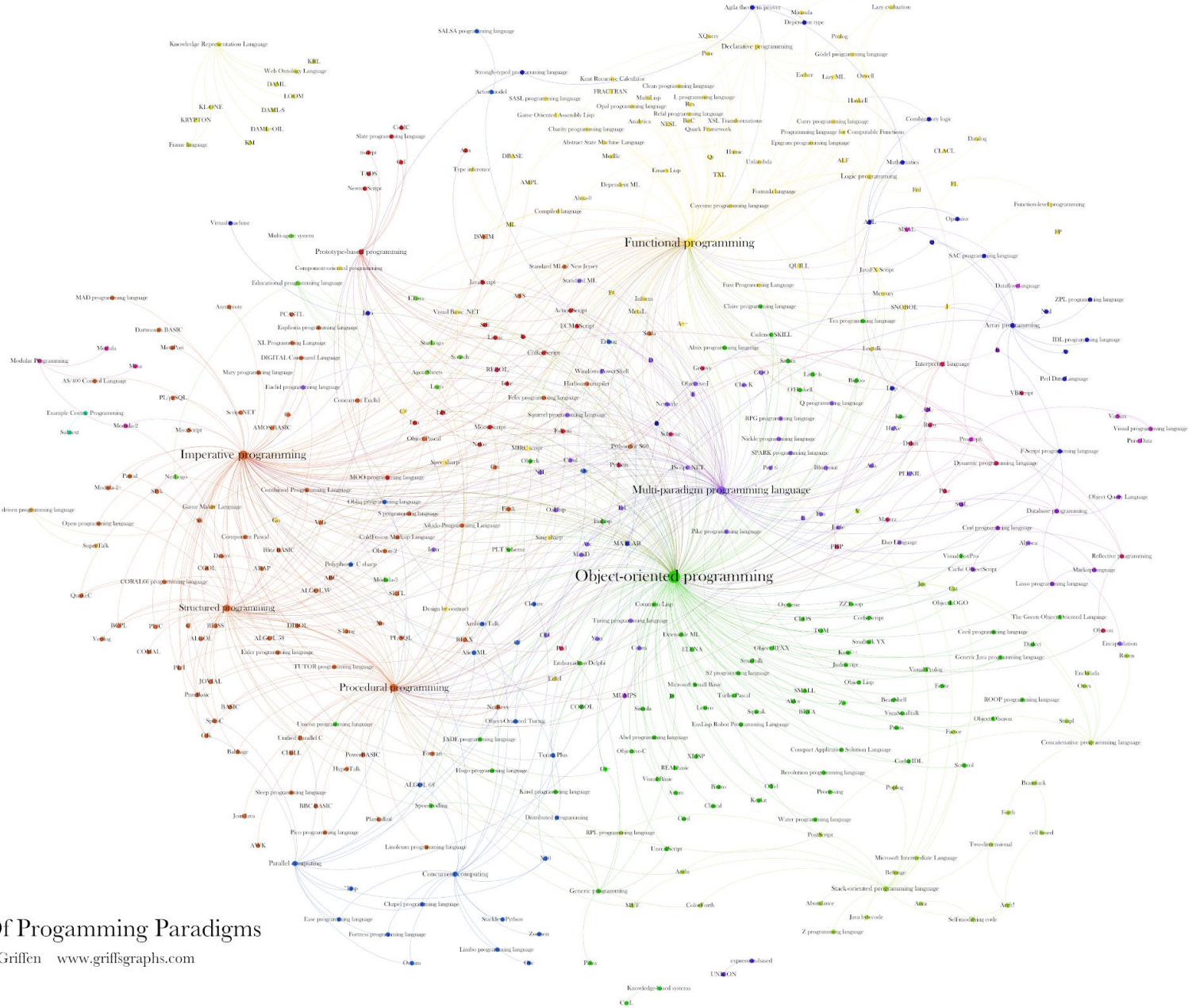

feestbeesten


Hallo
bol.com

Kotlin

DANGEROUS Kotlin

DANGEROUS Kotlin



The Graph Of Progamming Paradigms

By Brendan Griffen www.griffsgraphs.com

- Fix our problem
- Easy to write
- Easy to read

I call it my billion-dollar mistake. It was the invention of the null reference in 1965.

```kotlin
data class Coffee(val amount: MilliLiters) {
    fun merge(that: Coffee) = this + that

    private operator fun plus(extra: Coffee) = Coffee( amount: amount + 
}

typealias MilliLiters = Double
typealias CoffeeMerge = (Coffee, Coffee) -> Coffee

private val merge: CoffeeMerge = Coffee::merge

fun Double.toOunces() = this / 29.574

inline fun <reified T> T?.getClassString() = T::class.java.toString()
```


I ♥ Whiskey

Extension functions

```kotlin
private fun String.asUrl() = URL( spec: this)
```

```kotlin
private fun String?.asUrl() = this?.let { URL(it) }
```

```kotlin
private fun String?.toUrl() =
    try { this?.let { URL(it) } }
    catch (e: MalformedURLException) { null }
```

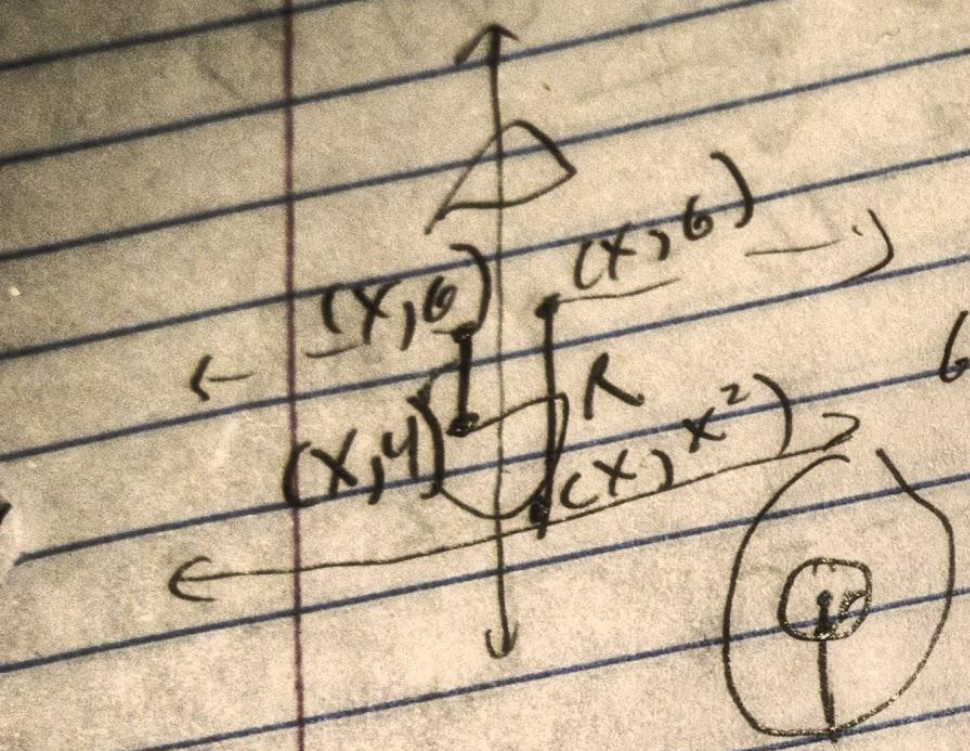
```kotlin
data class PromotionReport(
        val eans: Set<String>?,
        val supplierId: String?,
        val attribute: List<PromotedAttribute>?,
        val promotedAttributesCount: Int?
): BeagleEntity

fun PromotionReport.getEans(): Set<String> = this.eans.orEmpty()

fun PromotionReport.getTraceId(): String? {
    return this.attribute?.firstOrNull { it.id == AttributeId.TRACE_ID.valu
}

fun PromotionReport.getTraceIdOrThrow() = this.getTraceId() ?: throw Illega
```

```kotlin
val value = BigDecimal( val: "3.5")

val moreValue = value * 4

operator fun BigDecimal.times(int: Int) = this * BigDecimal(in
```

```kotlin
inline class Money(private val value: BigDecimal) {
    constructor(value: Number) : this(BigDecimal(value.toString()))

    fun toCurrency(type: CurrencyType) = Currency( money: this, type)

    operator fun times(number: Number) = this * Money(number)
    operator fun times(money: Money) = this * money.value
    operator fun times(bigDecimal: BigDecimal) = Money( value: value * bigDecimal)
}
```

Operator overloading

```
inline class Money(private val value: BigDecimal) {
    constructor(value: Number) : this(BigDecimal(value.toString()))

    fun toCurrency(type: CurrencyType) = Currency( money: this, type)

    operator fun times(number: Number) = this * Money(number)
    operator fun times(money: Money) = this * money.value
    operator fun times(bigDecimal: BigDecimal) = Money( value: value * bigDecimal)
}
```



$$\int \pi (R^2 - r^2)\, dx$$

$6 - x^2 = R$

$$\pi \int (6-x^2)^2 - r^2\, dx$$

$$\pi \int (6-x^2)^2 - (2)^2\, dx$$

$$\pi \int_{-2}^{2} (6-x^2)^2 - 4\, dx$$

$$\pi \int_{-2}^{2} 36 - 12x^2 +$$

-4

$2 = r$

limits

$x = 4$

4

$x = x^2$

StockTransactions
PriceTransactions

StockValueTransactions

$$x\Delta S * y\Delta P = xy\Delta V$$

```
val stockValueTransactions = stockTransactions * priceTransactions
```

Lambdas with receiver

```kotlin
private fun configureTopicAndSubscription(topic: String, subscriptions: List<String>) {
    pubSubAdmin.ignoringNotFound { deleteTopic(topic) }

    pubSubAdmin.createTopic(topic)
    subscriptions.forEach { it: String
        pubSubAdmin.ignoringNotFound { deleteSubscription(it) }
        pubSubAdmin.createSubscription(it, topic)
    }
}

private fun PubSubAdmin.ignoringNotFound(adminOperation: PubSubAdmin.() -> Unit) = try {
    adminOperation()
} catch (e: NotFoundException) {
    // ignore
}
```

Thanks

Attributions

- “Dangerous pirate” by Felix Lichtenfeld via Pixabay
- Math picture by Robert Couse-Baker from Pxhere

- All other images Creative Commons license


bol.com
Bart Enkelaar
Site Reliability Engineer
@BartEnkelaar
benkelaar@bol.com
https://github.com/benkelaar/dangerous-kotlin

Bonus

```kotlin
fun <T, U, V> receivingStuff(instance: T, functionWithReceiver: T.(U) -> V) = { input: U -> instance.functionWithReceiver(input) }

// fun <T, U, V> receivingStuff(instance: T, functionWithReceiver: T.(U) -> V) = instance.functionWithReceiver
// or
// fun <T, U, V> receivingStuff(instance: T, functionWithReceiver: T.(U) -> V) = instance::functionWithReceiver
```

```kotlin
fun <T, U, V> curry(function: (T, U) -> V) = { t: T -> { u: U -> function(t, u) } }

fun <T, U, V, W> curry(function: (T, U, V) -> W) = { t: T -> { u: U -> { v: V -> function(t, u, v) } } }

fun <T, U, V> uncurry(function: (T) -> (U) -> V) = { t: T, u: U -> function(t)(u) }

fun <T, U, V, W> uncurry(function: (T) -> (U) -> (V) -> W) = { t: T, u: U, v: V -> function(t)(u)(v) }
```

```kotlin
fun <T, U, V> curriedReceivingStuff(instance: T, functionWithReceiver: T.(U) -> V) = curry(functionWithReceiver)(instance)
```