Pourquoi Kotlin?

https://play.kotlinlang.org/byExample/overview

BEE
. s o f t w a r e

Vincent Tencé

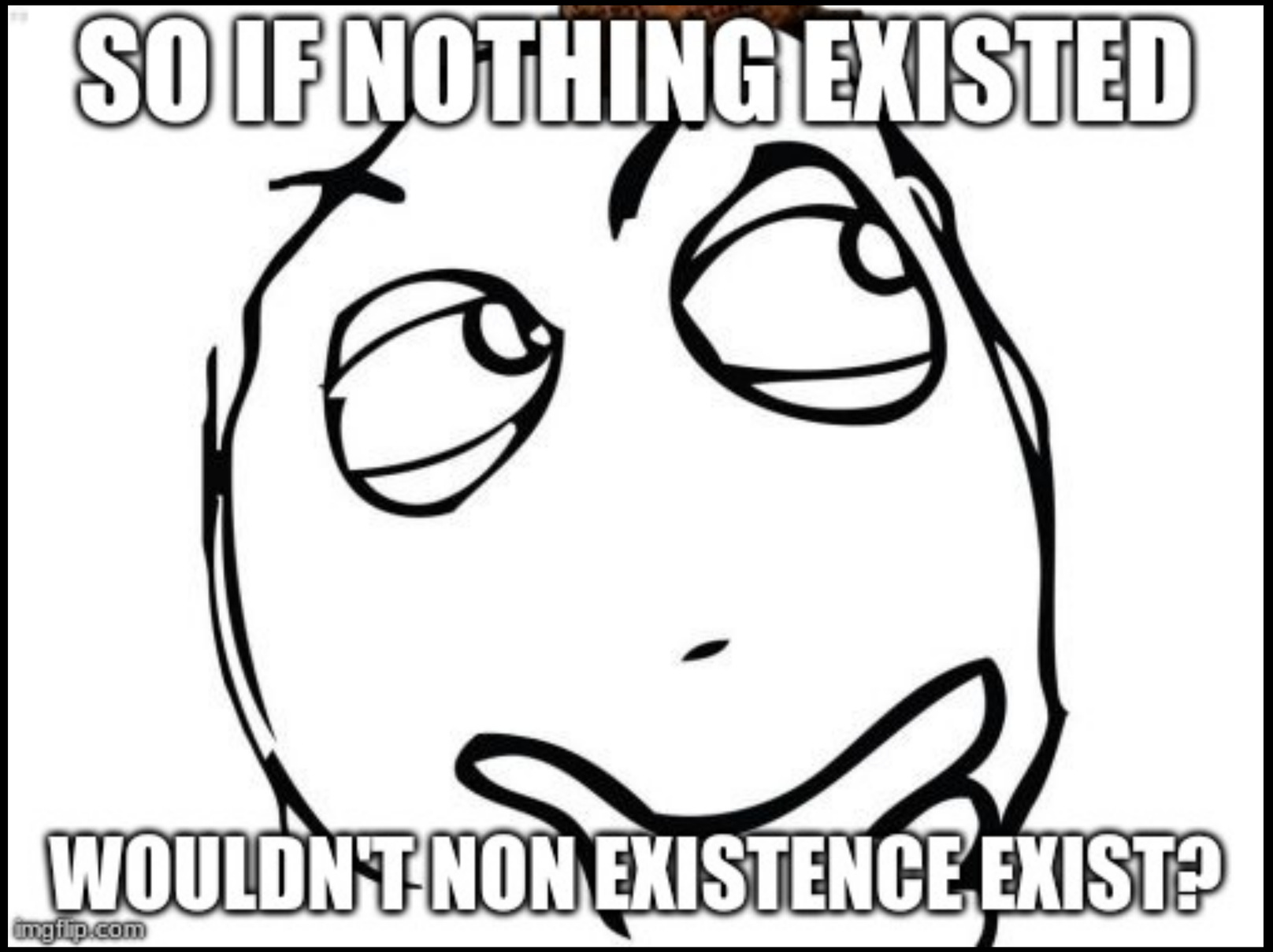
http://github.com/testinfected

- 20+ années d’expérience en développement
  - Java, Python, Ruby, Javascript, Scala, Kotlin, …
- 10+ années formateur Scrum Alliance et scrum.org
  - TDD, CSM, PSM, PSPO, PSD
- Co-Fondateur Développement Logiciel Bee

Aperçu

- Null Safety
- Operator Overloading
- Extension Methods
- Type System
- Collections
- Generics

Null Safety


SO IF NOTHING EXISTED
WOULDN'T NON EXISTENCE EXIST?
imgflip.com

Java

```
null, null, null, null, …
```

```
if (client.getNumber() == null) …
```

```
if (client.getNumber() != null) …
```

```
Optional<ClientNumber>
```

```
ClientNumber.none()
```

Kotlin

```
interface ClientFiles {
    fun findByNumber(number: ClientNumber): ClientFile?
}

val client = clients.findByNumber("11111")

client?.number

client?.number?.let { listOf(it) } ?: listOf()

client?.number?.let { listOf(it) }.orEmpty()
```

Data Classes


JAVA IS TOO VERBOSE
WHO CARES, INTELLIJ WRITES ALL THE CODE FOR YOU
imgflip.com

Java

```
public class Address {
    public final String streetNumber;
    public final String apartmentNumber;
    public final String street;
    public final City   city;
    public final String province;
    public final String country;
    public final String postalCode;

    public Address(String streetNumber, String apartmentNumber,
                   String street, City city, String province,
                   String country, String postalCode) {
        this.streetNumber = streetNumber;
        this.apartmentNumber = apartmentNumber;
        this.street = street;
        this.city = city;
        this.province = province;
        this.country = country;
        this.postalCode = postalCode;
    }

// … equals and hashCode
}
```

Kotlin

```
data class Address(
    val streetNumber: String,
    val street: String,
    val city: String,
    val postalCode: String? = null,
    val apartment: String? = null
)
```

Kotlin

```
class AddressLabelTest {
    @Test
    fun `without postal code or apartment number`() {
        val address = Address("34", "Sainte-Catherine W", "Montreal")

        assertThat(AddressLabel.onSingleLine(address), equalTo("34 Sainte-Catherine W, Montreal"))
    }

    @Test
    fun `with postal code but without apartment number`() {
        val address = Address("34", "Sainte-Catherine W", "Montreal", "H2K 1B4")

        assertThat(AddressLabel.onSingleLine(address), equalTo("34 Sainte-Catherine W, Montreal H2K 1B4"))
    }

    @Test
    fun `with both postal code and apartment number`() {
        val address = Address("34", "Sainte-Catherine W", "Montreal", "H2K 1B4", "#102")

        assertThat(AddressLabel.onSingleLine(address),
                   equalTo("#102 - 34 Sainte-Catherine W, Montreal H2K 1B4"))
    }
}


object AddressLabel {
    fun onSingleLine(address: Address) = with(address) {
        apartment?.plus(" - ").orEmpty() + "$streetNumber $street, $city" +
        postalCode?.let { " $it" }.orEmpty()
    }
}
```

Kotlin

```
data class EmailAddress(val email: String, val name: String? = null)

data class ContactCard(val firstName: String? = null, val lastName: String? = null,
                       val companyName: String? = null,
                       val email: String? = null, val phones: List<Phone> = listOf()) {

    val personalName
        get() = listOfNotNull(firstName, lastName).joinToString(" “)
                .trim().takeUnless { it.isEmpty() }

    val emailAddress get() = email?.let { EmailAddress(it, personalName ?: companyName) }

    fun phoneOfType(type: PhoneType) = phones.find { it.type == type }
}

data class ShippingLabel(val from: ContactCard,
                         val recipients: List<ContactCard> = listOf(),
                         val cc: List<ContactCard> = listOf()) {

    fun add(vararg recipients: ContactCard) = add(recipients.toList())

    fun add(extraRecipients: List<ContactCard>) =
           copy(recipients = recipients + extraRecipients)
}

class Envelope(val shippingLabel: ShippingLabel, val content: Content,
               val documents: List<Document> = listOf())
```

Operator Overloading


WHAT IF I TOLD YOU
YOU CAN ADD LISTS WITH +
imgflip.com

Java

```
TableSchema schema = new TableSchema(“document_index");

Table<IndexEntry> index = new Table<>(schema, new IndexEntryRecord(
        schema.STRING("folder"),
        schema.STRING("document")
));

public List<DocumentReference> list(Folder folder) {
    return Select.from(index)
                 .where("folder = ?”,
                        folder.identifier().toASCIIString())
                 .list(connection)
                 .stream()
                 .map(IndexEntry::document)
                 .collect(toList());
}
```

Java

```java
public class IndexEntryRecord implements Record<IndexEntry> {

    private final Column<String> folder;
    private final Column<String> document;

    public IndexEntryRecord(Column<String> folder,
                            Column<String> document) {
        this.folder = folder;
        this.document = document;
    }

    public IndexEntry hydrate(ResultSet rs) {
        return new IndexEntry(Folder.create(folder.get(rs)),
                              DocumentReference.of(document.get(rs)));
    }

    public void dehydrate(PreparedStatement st, IndexEntry entity) {
        folder.set(st, entity.folder().toASCIIString());
        document.set(st, entity.document().orElse(null));
    }
}
```

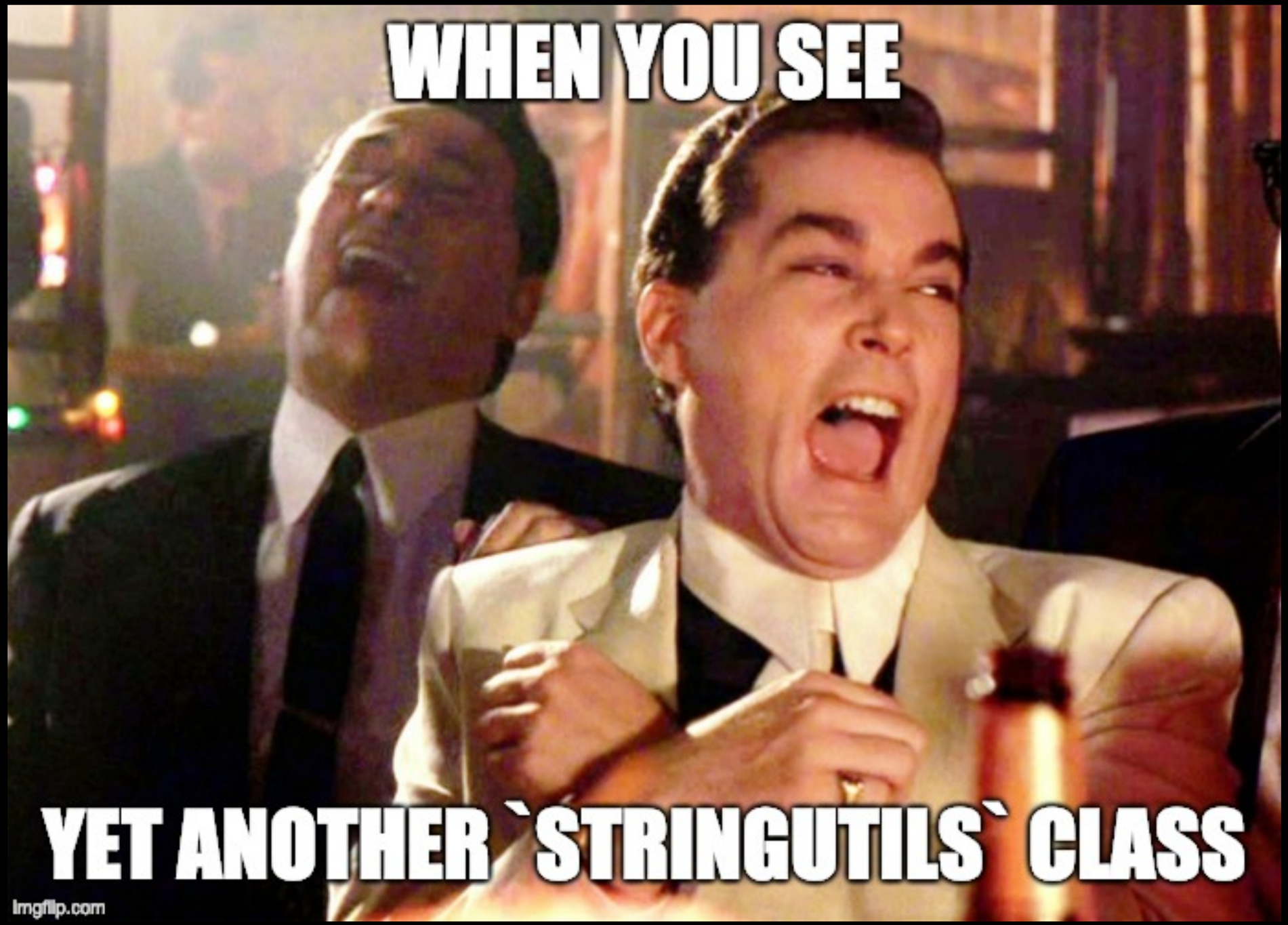
Kotlin

```
override fun hydrate(rs: ResultSet): StaffMember {
   return StaffMember(ContactCard(
           firstName = firstName[rs],
           lastName = lastName[rs],
           email = email[rs]))
}

override fun dehydrate(st: PreparedStatement,
                       contact: StaffMember) {
    st[firstName] = contact.contact.firstName
    st[lastName] = contact.contact.lastName
    st[email] = contact.contact.email
}

operator fun <T> ResultSet.get(col: Column<T>): T = col[this]

operator fun <T> PreparedStatement.set(col: Column<T>, value: T) {
    col[this] = value
}
```

Kotlin

```
val statuses = mappingOf(
    Status.Planned to "planifié",
    Status.Completed to "complété"
)

st[status] = statuses[entity.status]
entity.status = statuses[rs[status]]


fun <E: Enum<E>, V> mappingOf(vararg pairs: Pair<E, V>) =
EnumMapping(*pairs)

class EnumMapping<E: Enum<E>, V>(vararg pairs: Pair<E, V>) {
    val mappings = mapOf(*pairs)

    operator fun get(key: E) = mappings.get(key)
    operator fun get(value: V) : E? =
        mappings.filterValues { it == value }.keys.firstOrNull()
}
```

Extension Methods


WHEN YOU SEE
YET ANOTHER `STRINGUTILS` CLASS
imgflip.com

Java

```java
public class StringUtils {

    public static boolean stringIsNullOrEmpty(String s) {
        return s == null || s.trim().isEmpty();
    }

    public static String stringNullToEmpty(String s) {
        return s == null ? "" : s;
    }

    public static String stringEmptyToNull(String s) {
        return stringIsNullOrEmpty(s) ? null : s;
    }

    public static String stringOrDefault(String s, String defaultValue) {
        return stringIsNullOrEmpty(s) ? defaultValue : s;
    }

    public static Optional<String> stringNotEmptyNorNull(String s) {
        return Optional.ofNullable(stringEmptyToNull(s));
    }


    public static String sanitizeHtml(String s) {
        return s == null ? "" : Jsoup.parse(s).text();
    }
}
```

Kotlin

```
val sanitized =
“<p>Some potentially unsafe markup</p>”.sanitizeHtml()


fun String?.sanitizeHtml() = sanitize(this)


fun sanitize(unsafeHtml: String?) = unsafeHtml?.let {
        Jsoup.parse(it).text()
}
```

Types


STILL WAITING
FOR A PROPER
TYPE SYSTEM
imgflip.com

Java

`class`

`interface`

`@FunctionalInterface`

`Enum`

`int, long, array, etc.`

`Wrapper Types`

`void`

`static`

Kotlin

`class, enum class, data class, sealed class, inline class`

`object`

`interface`

`functions`

`typealias`

`Basic Types`

`Unit, Any, Any?, Nothing`

SAM Conversion

```
@FunctionalInterface
public interface Runnable {
   void run();
}
```

```
fun start() {
    scheduler.scheduleAtFixedRate(
        { trigger.executeEventsDueAt(clock.instant()) },
        0, delayInSecs, TimeUnit.SECONDS)
}
```

Function Types

```
interface View<T> {
    fun render(Response response, T model)
}

typealias View<T> = (model: T) -> Response


typealias ContentRenderer<T> = (data: T) -> Content


interface ContentConverter<out T : Content> {
    fun convert(content: Content): T
}
```

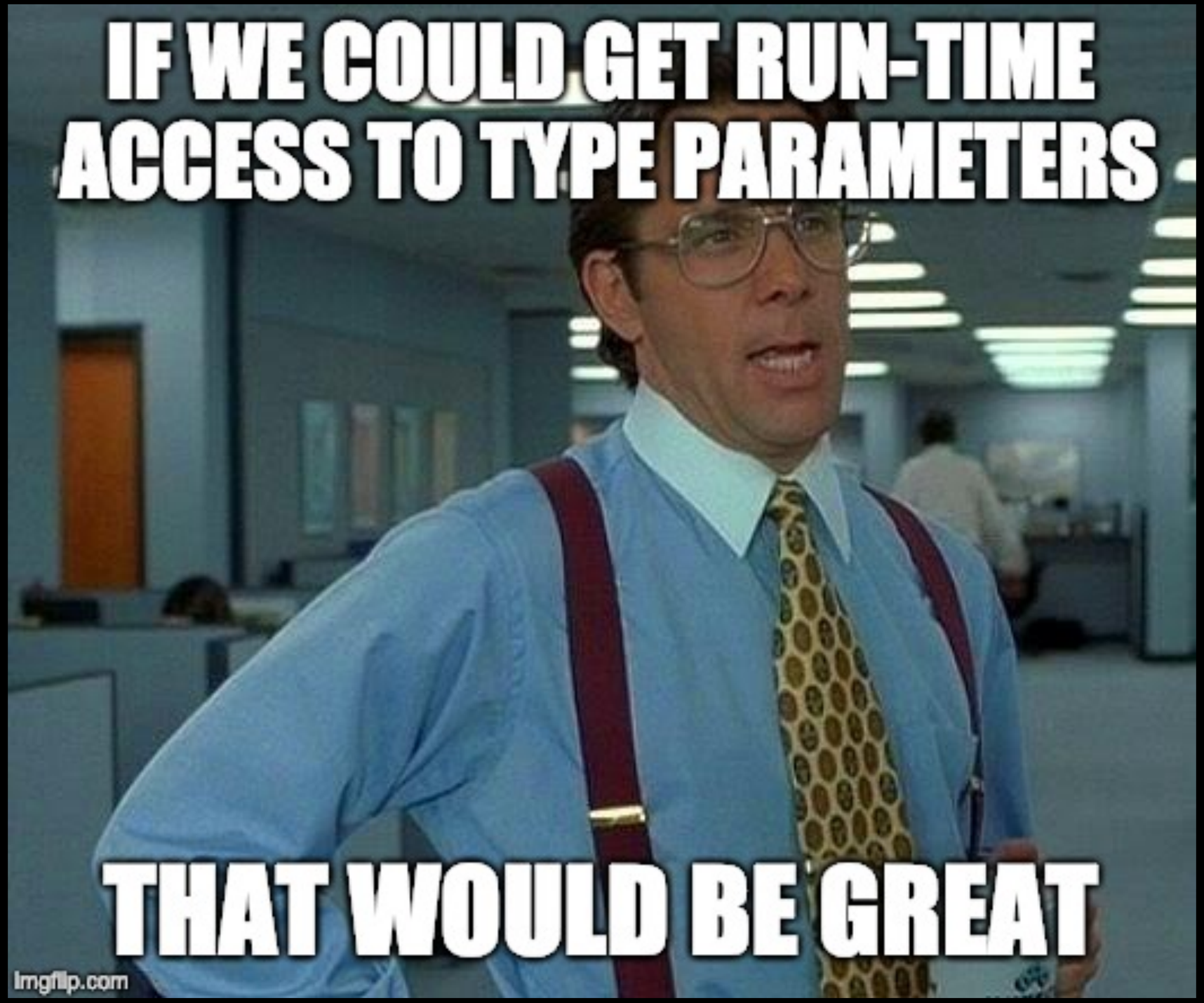
Object Expressions

```
interface ScheduledEventHandler {
    fun handle(event: EventData)
}

val later = ScheduledEventHandler { data -> process(data) }

interface ScheduledEventHandler {

   companion object {
     operator fun invoke(handling: (data: EventData) -> Unit) =
object : ScheduledEventHandler {
            override fun handle(event: EventData) =
                    handling(event)
        }
    }
}
```

Sealed Classes

```
interface ContactDirectory {
    fun findIndividuals(number: ClientNumber): List<ContactCard>
    fun findCompanies(number: SupplierNumber): List<ContactCard>
    fun findStaffMembers(id: ID): ContactCard?
}

sealed class Contact(val contact: ContactCard)

class Individual(val number: ClientNumber, contact: ContactCard): Contact(contact)

class Company(val number: SupplierNumber, contact: ContactCard): Contact(contact)

class StaffMember(contact: ContactCard): Contact(contact) {
    var id: ID? = null
}

fun add(vararg contacts: Contact) {
    contacts.forEach {
        when (it) {
            is Individual -> Insert.into(individuals, it).execute(db)
            is Company -> Insert.into(companies, it).execute(db)
            is StaffMember -> Insert.into(staff, it).execute(db)
        }
    }
}
```

Reified Types


IF WE COULD GET RUN-TIME ACCESS TO TYPE PARAMETERS
THAT WOULD BE GREAT
imgflip.com

Java

```java
public class Problem {
    private static final Gson gson = new GsonBuilder().create();

    public final int status;
    public final String title;
    public final Map<String, List<String>> errors;

    public Problem(int status, String title, Map<String, List<String>> errors) {
        this.status = status;
        this.title = title;
        this.errors = errors;
    }

    public static Problem fromJson(String json) {
        return gson.fromJson(json, Problem.class);
    }

    public String toJson() {
        return gson.toJson(this);
    }
}

Problem problem = Problem.fromJson(Objects.requireNonNull(response.body()));
```

Kotlin

```kotlin
val contact: ContactObject = request.body().fromJson()

val contact = request.body().fromJson<ContactObject>()

val response = Response.ok()
                       .header("Content-Type", "application/json")
                       .body(contact.toJson())
                       .done()


val gson: Gson = GsonBuilder().create()

fun Any.toJson(): String = gson.toJson(this)


inline fun <reified T> String.fromJson() = gson.fromJson<T>(this)

inline fun <reified T> Gson.fromJson(json: String): T =
       this.fromJson<T>(json, T::class.java)
```

Java

```
public class ConsentForm {
    private boolean agreement;
    private String token;

    public static ConsentForm from(Request request) {
        ConsentForm form = new ConsentForm();
        form.token = request.parameter("token");
        form.agreement = request.parameter("agreement") != null;
        return form;
    }
...
}
```

```
CookieJar cookies = request.attribute(CookieJar.class);
```

```
public class Session implements Serializable {

    public static Session get(Request request) {
        return request.attribute(Session.class);
    }
}
```

Kotlin

```kotlin
val id = request["id"]?.toInt() ?: return
         Response.of(HttpStatus.BAD_REQUEST)

val cookies = request.attribute<CookieJar>()


operator fun Request.get(parameterName: String): String? =
         this.parameter(parameterName)

inline fun <reified T> Request.attribute(): T =
         this.attribute(T::class.java)
```

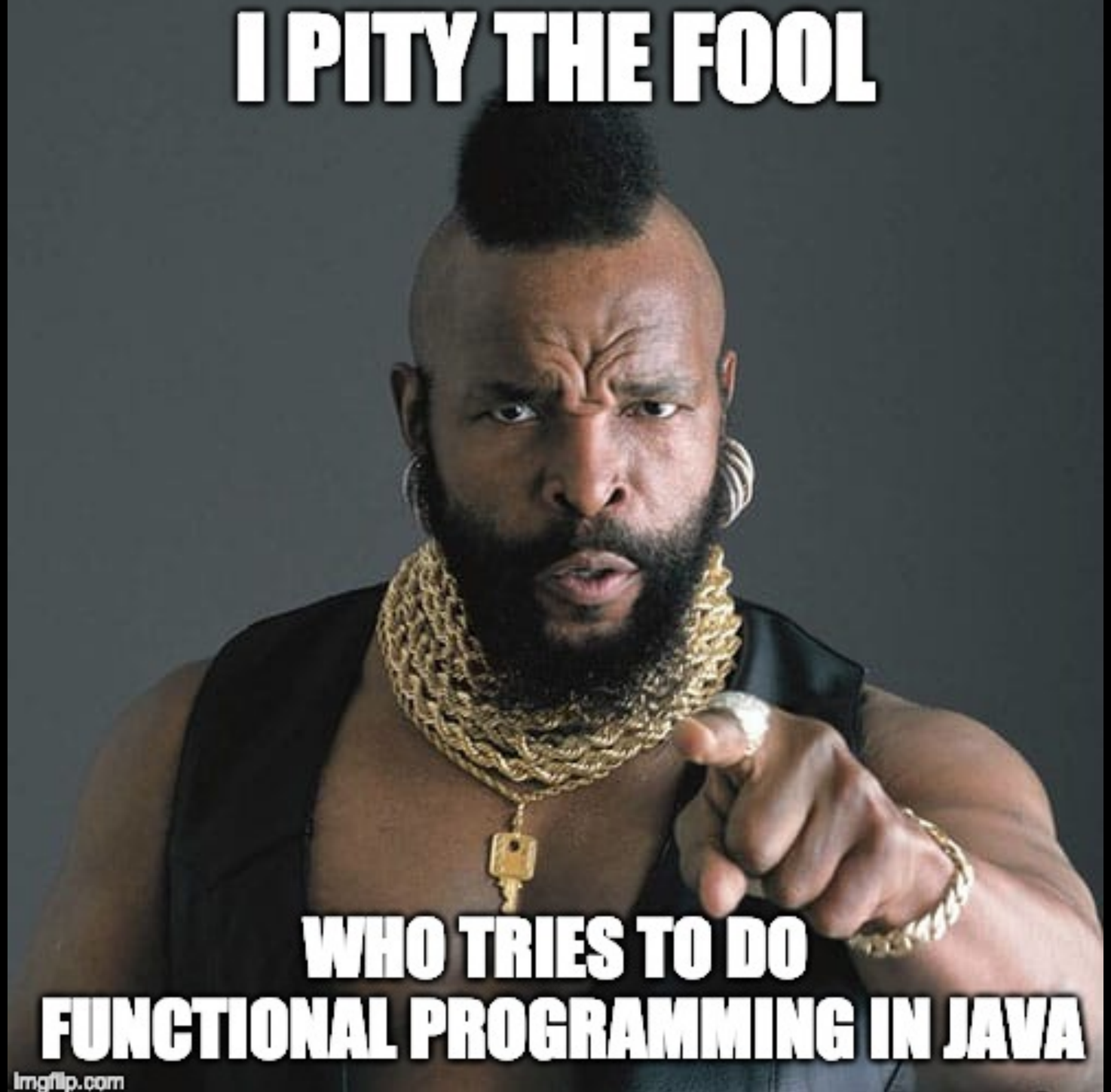
Delegated Properties

```
by map // storing properties in a map, instead of a separate
field for each property;

val claimNumber: ClaimNumber by data


by lazy // the value gets computed only upon first access;

val quotes: QuotesDatabase by lazy { QuotesDatabase(connection)
}


by Delegates.observable("initial value”) // listeners get
notified about changes to this property;
```

Kotlin

```
typealias EventData = Map<String, Any?>


typealias ClientNumber = String


class RegistrationFollowUp(data: EventData) {
    val clientNumber: ClientNumber by data

    companion object {
        const val type = “registration-follow-up"

        fun data(client: ClientFile) = mapOf("clientNumber" to client.number)

        fun schedule(client: ClientFile) = ScheduledEvent(triggerDate(client),
type, data(client))

        private fun triggerDate(client: ClientFile) =
            client.registrationDate.atMidnight().plus(15, ChronoUnit.DAYS)
    }
}
```

Functions


I PITY THE FOOL
WHO TRIES TO DO
FUNCTIONAL PROGRAMMING IN JAVA
imgflip.com

Functional Programming

- Data Classes
  - Immutability and copy()
- Package level functions
- Companion Objects
- Functions and Lambdas

Java

```
Router routes = Router.draw(new Routes() {{
    get("/").to(renderStatic(views.named("home")));
    get("/join").to(accounts::new_);
    get("/join/plan").to(authenticated(plan::join));
    get("/join/payment").to(authenticated(payment::join));
    post("/accounts").to(accounts::create);
    get("/account/plan").to(authenticated(plan::edit));
    post("/account/plan").to(authenticated(plan::create));
    get("/account").to(authenticated(accounts::get));
    get("/account/payment").to(authenticated(payment::new_));
    post("/account/payment").to(authenticated(payment::create));
    delete("/account/payment").to(authenticated(payment::delete));
    get("/login").to(sessions::open);
    post("/session").to(sessions::create);
    delete("/session").to(sessions::delete);
    get("/download").to(renderStatic(views.named("download")));
    get("/reset-password").to(resets::new_);
    get("/password-resets/created").to(renderStatic(views.named("password-resets/
show")));
    get("/password-resets/done").to(renderStatic(views.named("password-resets/done")));
    get("/password-resets/:token").to(resets::edit);
    post("/password-resets").to(resets::create);
    delete("/password-resets/:token").to(resets::complete);
}});
```

Kotlin

```
class WebApp(private val app: ApplicationScope) : Application {

    val diagnostics = routes {
        get("/status").to { Response.ok().contentType("text/plain").done("All green") }
    }

    override fun handle(request: Request): Response {
        val scope: RequestScope = app.scopeOf(request)

        val router = Router.draw(
                diagnostics then
                scope.client.routes then
                scope.calendar.routes
        )

        return router.handle(request)
    }
}

fun routes(definition: Routes.() -> Unit) = Routes().apply(definition)

infix fun RouteBuilder.then(others: RouteBuilder): RouteBuilder = CombinedRoutes(this, others)

private class CombinedRoutes(private val first: RouteBuilder, private val second: RouteBuilder)
              : RouteBuilder {
    override fun build(routes: RouteSet?) {
        first.build(routes)
        second.build(routes)
    }
}
```

Java

```
@Rule
public JUnitRuleMockery context = new JUnitRuleMockery();

ChartOfAccounts chart = context.mock(ChartOfAccounts.class);

AccountingRules rules = new AccountingRules(chartOfAccounts);

@Before
public void givenPayableAccounts() {
    context.checking(new Expectations() {{
        allowing(chart).payableAccounts(); will(returnValue(
                List.of(anAccountWithNumber("40000"),
                        anAccountWithNumber("50480"))));
    }});
}
```

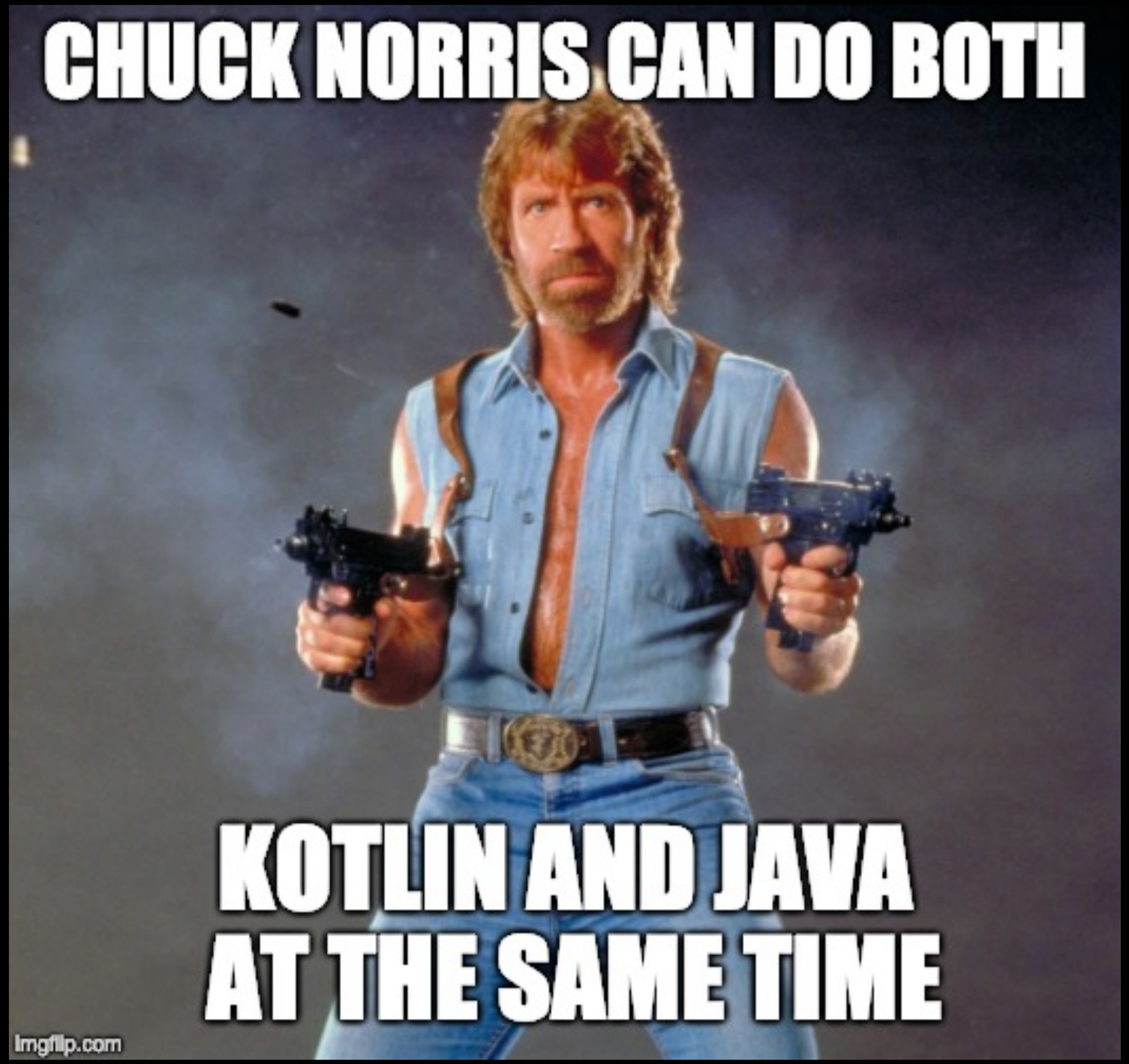
Kotlin

```
val mockery = JUnitRuleMockery()

val schedule = mockery.mock<Schedule>()

val dispatcher = ScheduledEventsDispatcher(schedule)


@Test fun `dispatches due events in order`() {
    val handler = FakeEventHandler()

    dispatcher.subscribe("any", handler)

    mockery.check {
        allowing(schedule).listEventsDueAt(dueDate) will returnListOf(
            ScheduledEvent(at = dueDate, type = "any", data = mapOf("order" to "first")),
            ScheduledEvent(at = dueDate, type = "any", data = mapOf("order" to "second")),
            ScheduledEvent(at = dueDate, type = "any", data = mapOf("order" to "third"))
        )
   }

    dispatcher.executeEventsDueAt(dueDate)

    assertThat(
        handler.events, contains(
            hasData("order" to "first"),
            hasData("order" to "second"),
            hasData("order" to "third")
        )
    )
}
```

Kotlin

```kotlin
inline fun <reified T> Mockery.mock() = this.mock<T>(T::class.java)!!


inline fun <reified T> Mockery.mock(name: String) =
    this.mock<T>(T::class.java, name)!!


fun Mockery.check(expectations: Expects.() -> Unit) =
    this.checking(Expects().apply(expectations))


class Expects : Expectations() {

    infix fun Any?.will(action: Action) = super.will(action)

    fun Any?.willReturn(value: Any?) = super.will(returnValue(value))

    fun Any?.willReturnList(vararg items: Any?) = super.will(returnListOf(items))

    fun returnListOf(vararg items: Any?) = returnValue(listOf(*items))
}
```

Tooling


CHUCK NORRIS CAN DO BOTH
KOTLIN AND JAVA
AT THE SAME TIME
imgflip.com

Java Interoperability

- Calling Java from Kotlin

```
!, !!, @NonNull, Getters, Setters, Unit, Unchecked Exceptions
```

- Calling Kotlin from Java

```
@JvmStatic, @JvmField, @JvmName
```

- IntelliJ
- Gradle

Spek

```
object CalculatorSpec: Spek({
    describe("A calculator") {
        val calculator by memoized { Calculator() }

        describe("addition") {
            it("returns the sum of its arguments") {
                assertThat(3, calculator.add(1, 2))
            }
        }
    }
})
```

Konfig

```
val options: Array<CommandLineOption> = arrayOf(
    CommandLineOption(env, short = "e"),
    CommandLineOption(server.host),
    CommandLineOption(server.port, short = "p"),
    CommandLineOption(quiet, short = "q")
)

var (options, _) = parseArgs(arrayOf(*args), *options) overriding
        EnvironmentVariables() overriding
        ConfigurationProperties.fromResource("etc/${options[env]}.properties")
overriding
        ConfigurationProperties.fromResource("etc/defaults.properties")

val server = Server(options[server.host], options[server.port])

val env = Key("env", stringType)
val quiet = Key("quiet", booleanType)

object Settings {
    val env = Key("env", stringType)

    object server : PropertyGroup() {
        val host by stringType
        val port by intType
    }
}
```

Ktor

```
fun main(args: Array<String>) {
    val server = embeddedServer(Netty, port = 8080) {
        routing {
            get("/") {
                call.respondText("Hello World!",
                                 ContentType.Text.Plain)
            }
            get("/demo") {
                call.respondText("HELLO WORLD!")
            }
        }
    }
    server.start(wait = true)
}
```

Futurs JEP

Switch expressions: http://openjdk.java.net/jeps/325

Pattern matching for instanceof: http://openjdk.java.net/jeps/305

Pattern matching for Switch: http://openjdk.java.net/jeps/8213076

Raw String Literals: https://openjdk.java.net/jeps/326

Concise method bodies: http://openjdk.java.net/jeps/8209434

Lazy Static Final Fields: http://openjdk.java.net/jeps/8209964

Type Operators Expressions: http://openjdk.java.net/jeps/8204937

Project Valhalla: https://wiki.openjdk.java.net/display/valhalla/Main

Data Classes for Java: https://cr.openjdk.java.net/~briangoetz/amber/datum.html

Références

- https://github.com/testinfected/molecule
- https://github.com/testinfected/tape
- https://ktor.io/
- https://spekframework.org/
- https://github.com/npryce/konfig

Kotlin

.software