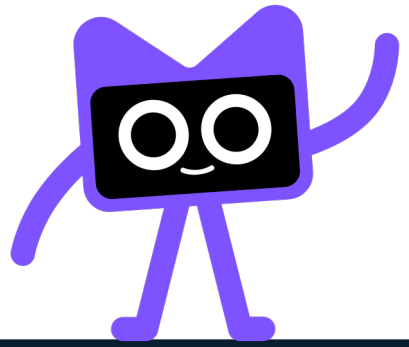
/sit schwarz IT



Web development in Kotlin

Suman Venkat
Sebastian Muskalla

42 / TUM Heilbronn | July 31, 2024

Agenda

01 Introduction
Introduction of SIT, trainers and participants

02 About Kotlin

03 Morning session
Kotlin basics

04 Lunch break + Q & A

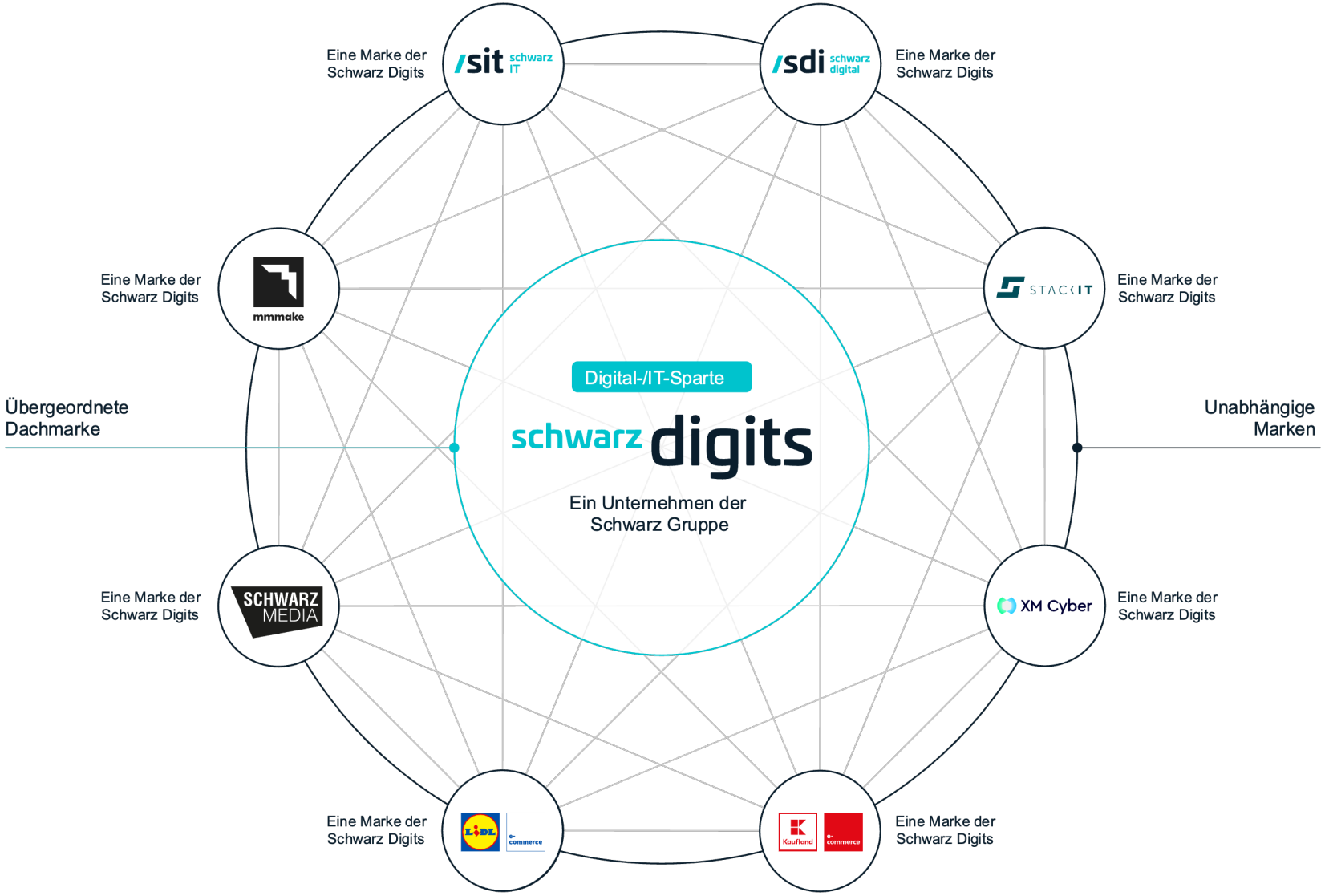
05 Afternoon session
Testing & Web development

06 Q & A

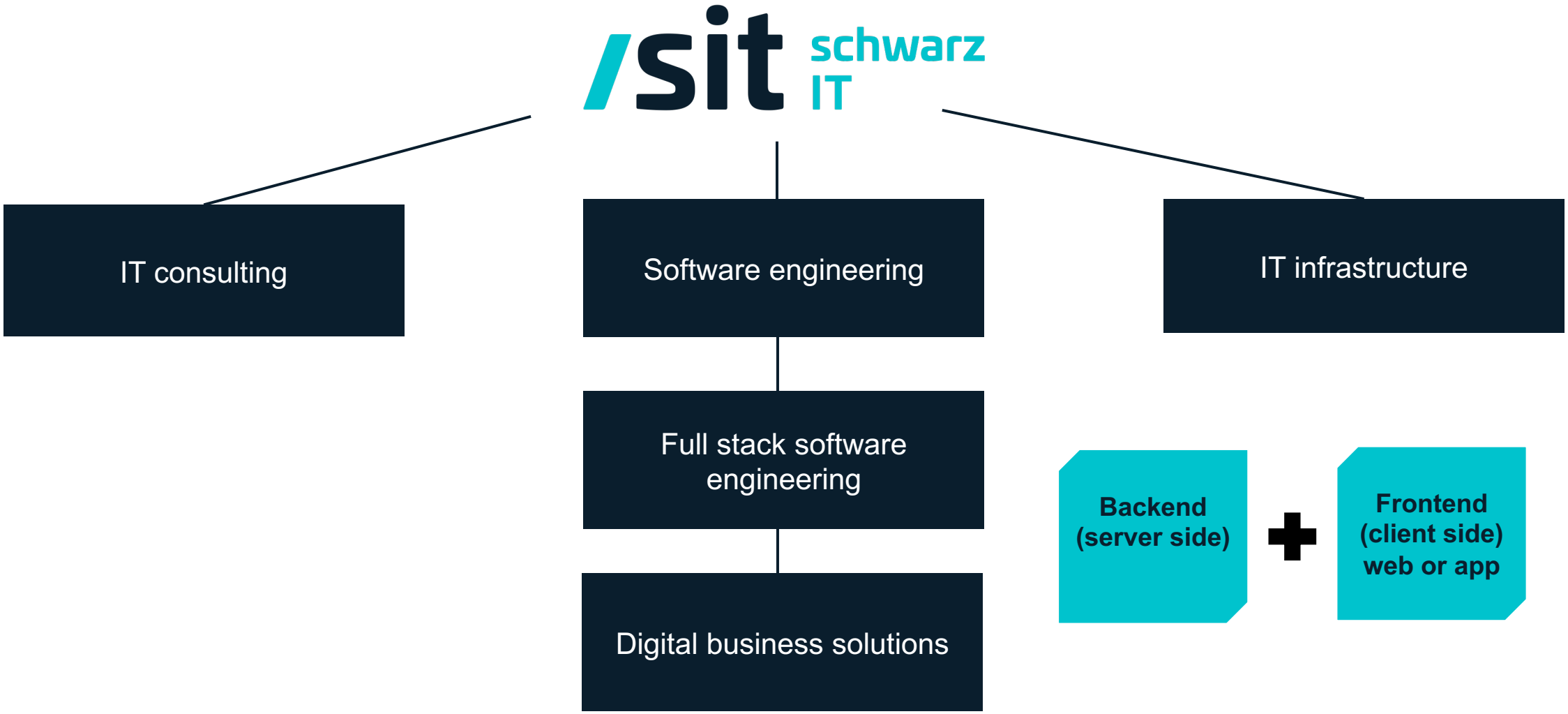
Introduction


SCHWARZ
LIDL
Kaufland
SCHWARZ
Produktion
pre
zero
schwarz
digits

Schwarz Digits



Schwarz IT


/sit schwarz IT
IT consulting
Software engineering
IT infrastructure
Full stack software engineering
Digital business solutions
Backend (server side)
Frontend (client side) web or app

/sit schwarz IT

Sebastian Muskalla

Sebastian.Muskalla@mail.schwarz

2015 MSc. (mathematics)

2022 PhD (theoretical computer science)

2023 Full stack software engineer @ Schwarz IT

Suman Venkat

Suman.Venkat@mail.schwarz

2019 Junior Consultant @ Schwarz IT

2021 Full stack software engineer

Introduction: You!

- How long have you been at 42 / TUM Heilbronn?
- With which technologies / programming languages do you have experience?
- What do you hope to learn in this workshop?

/sit schwarz IT

About Kotlin

Compiled and interpreted languages

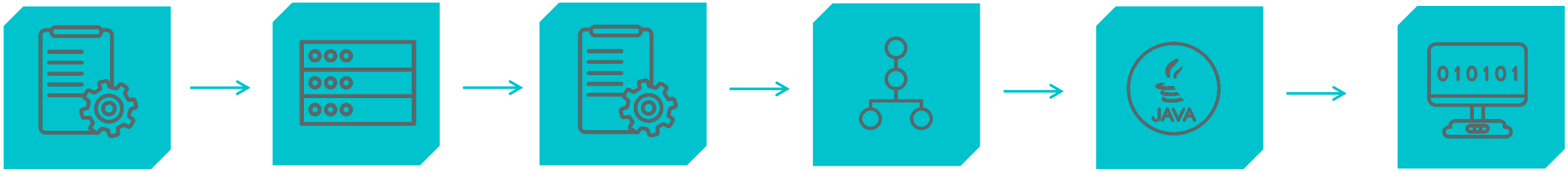
Compiled languages

Interpreted languages

Modern languages use

JIT compilation

Java and the Java virtual machine (JVM)



- Originally released in 1996
- The most popular programming language of the last 25 years [1]
  - Business sector
  - Android
- Failed attempts to replace it: Groovy, Scala, …
- Dropping popularity in the last few years:
  - JavaScript/TypeScript, Python
  - **Kotlin**

[1] https://www.tiobe.com/tiobe-index/

/sit schwarz IT

HelloWorld.java

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

/sit schwarz IT

Kotlin

2010: Start of development

2011: Public announcement by JetBrains

2012: Open source‘d (Apache license)

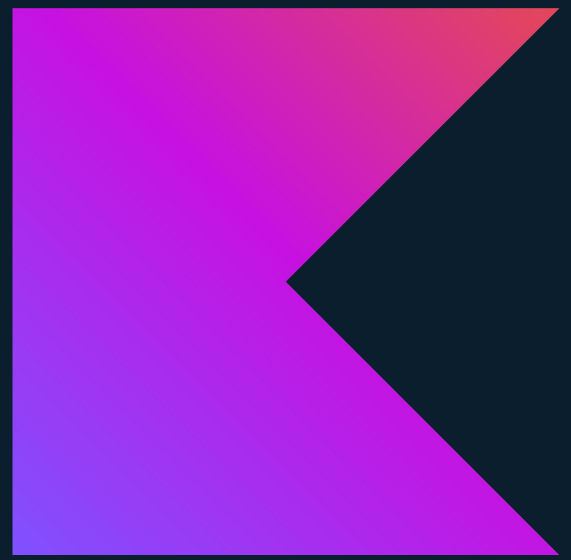
2016: First stable release

2017: Kotlin Multiplatform beta

2019: Declared by Google as preferred language for Android apps

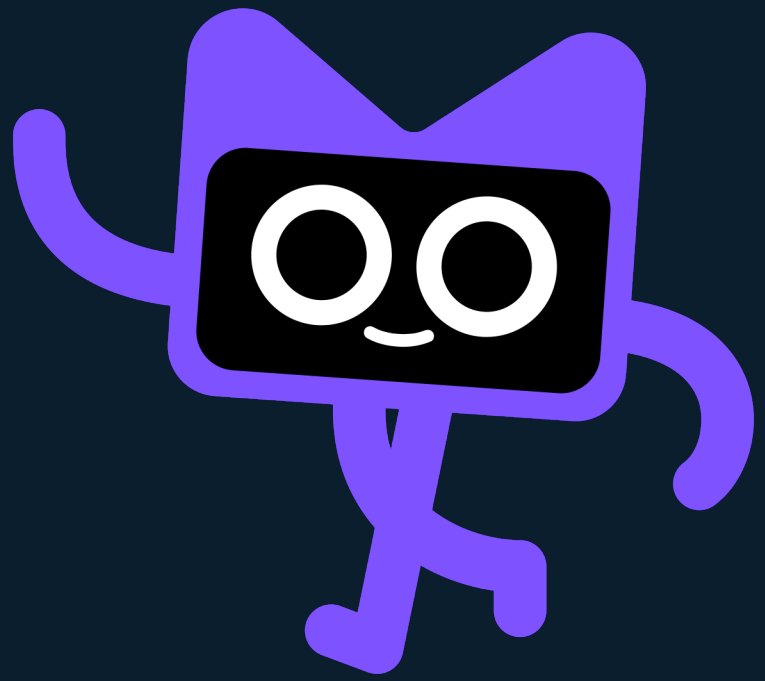
2023: Kotlin Multiplatform stable

2024: Kotlin 2.0



Why Kotlin?

- Less verbose
- Full Java interoperability
- Null-safety
- Declarative programming



Kodee

Programming paradigms

Imperative programming

- **Structured programming**
- **Procedural programming**
- **Object-oriented programming**

…

Declarative programming

- **Functional programming**

…

SquaresImperative.kt

```kotlin
val squares = mutableListOf<Int>()

for (i in 0..100) {
    squares.add(i * i)
}

return squares
```

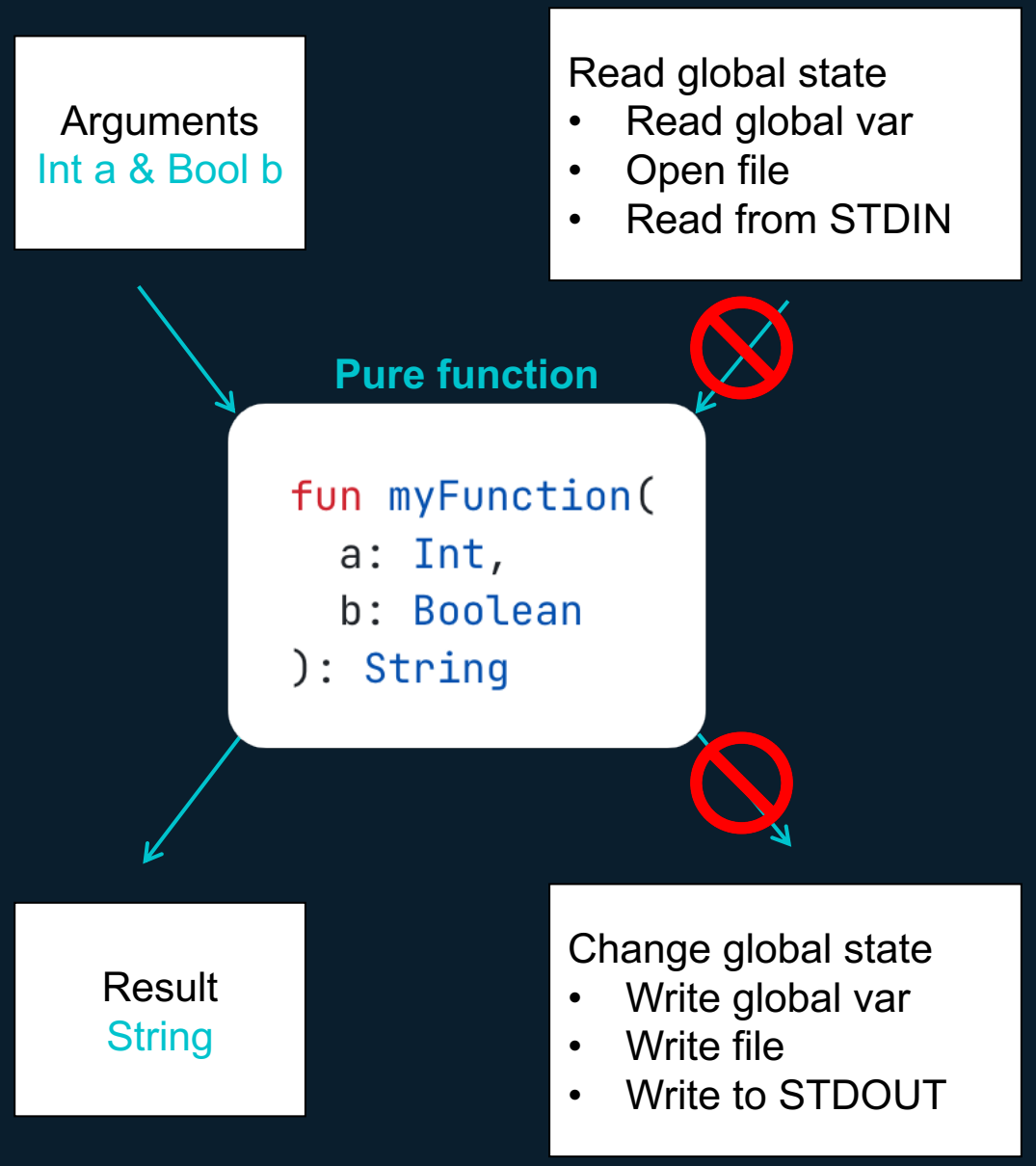
SquaresDeclarative.kt

```kotlin
val allNumbers = generateSequence(0) { it + 1 }

val allSquares = allNumbers.map { it * it }

val firstSquares = allSquares.take(100)

return firstSquares.toList()
```

Declarative programming

- Telling the computer what to do,
  not how to do it
- Focus on immutability
- Functional programming with pure functions
- Higher-order functions

Makes code easier to read, test, debug, maintain



/sit schwarz IT

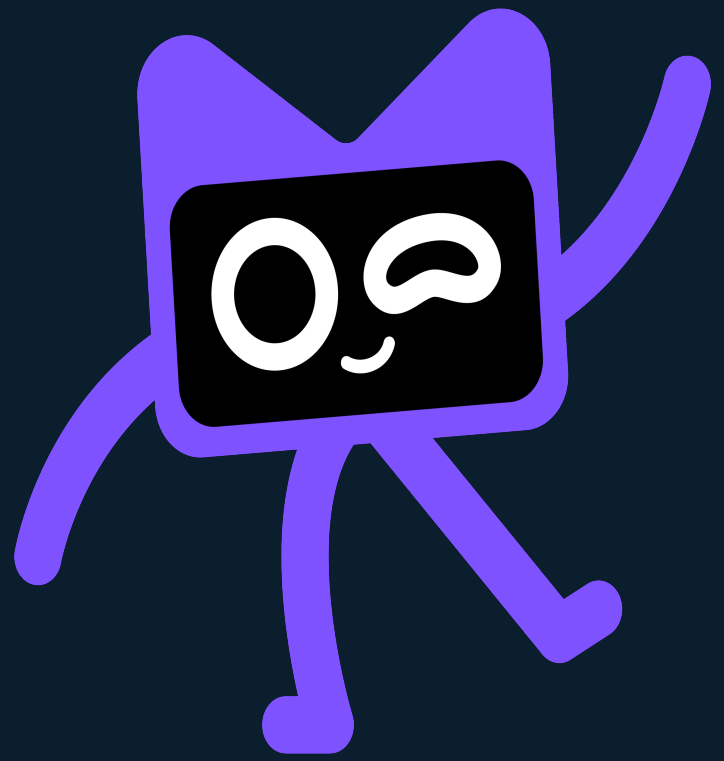
Kotlin setup check

https://github.com/SIT-Kotlin-Workshop

Morning session

Basic syntax overview

- If-Then
- When
- For Loop
- While Loop
- Functions
  - With body
  - With expression



If-Else

```
1  val number = 5
2  val result = if (number > 3) {
3    "Greater"
4  } else {
5    "Lesser"
6  }
```

When

```
1  val x = 1
2  when (x) {
3    1    → println("One")
4    2    → println("Two")
5    else → println("Other")
6  }
```

For

```
1  val names = listOf("Sebastian", "Suman")
2
3  for (name in names) {
4    println("Hello, $name")
5  }
```

While

```
1  var timeLeft = 10
2  while (timeLeft > 0) {
3    println("Time Left: $timeLeft seconds")
4    timeLeft--
5  }
6  println("Time's up!")
```

Function with body

```
1  fun calculateDiscount(
2    price: Double,
3    discountRate: Double
4  ): Double {
5    val discountAmount = price * discountRate / 100
6    return price - discountAmount
7  }
```

Function as expression

```
1  fun calculateDiscount(
2    price: Double,
3    discountRate: Double
4  ): Double =
5    price - (price * discountRate / 100)
```

Basic syntax

Exercises

Immutable data in Kotlin

- **Immutable:** Whose state cannot be modified after they are created
- Mutable vs Immutable
  - Mutable `var`
  - Immutable `val`
- `val` keyword used to declare read only variables
- Lists, Sets and Maps can be immutable while using collections
- **Data classes:** Simplifies the creation of immutable classes with automatic `equals()`, `hashcode()`, `toString()`, and `copy()`

Immutable

```
1  val immutableString: String = "Immutable"
2  immutableString = "Mutable" // Error
```

Immutable list

```
1  val immutableList = listOf("A", "B", "C")
2  immutableList.add("D") // Error: Unresolved reference: add
```

Immutable class

```
1  data class MyClass(val primary: String)
2  val immutable = MyClass("Hello")
3  immutable.primary = "World" // Error
```

```
1  val updatedString = immutable.copy(primary = "World")
```

Mutability and references

- Mutability of references

- Mutability of contents

MutableList.kt

```
1  val mutableList = mutableListOf(1, 2, 3)
2  mutableList.add(4) // allowed
```

ImmutableList.kt

```
1  var immutableList = listOf(1, 2, 3)
2  immutableList.add(4) // not allowed
3  immutableList = listOf(4) // allowed
```

Immutable Data

Exercises

Nullability

In Java and other languages, the special value null is allowed for (almost) all types.

It can mean

- not initialized
- undefined / not provided (as an argument of a function)
- error (as a return value of a function)
- …

Problems:

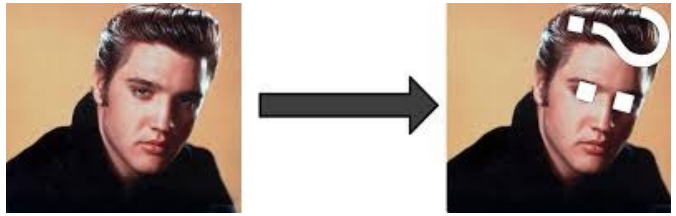
- Don‘t know which of the above
- Don‘t know when objects are null
- Accessing null crashes with NPE (NullPointerException)

Nullability.kt

```kotlin
data class Person(
    val name: String,
    val birthday: LocalDate,
) {
    fun format(): String = "$name ($birthday)"
}

fun main() {

    val person: Person = null

    // Property access would crash with NPE
    println(person.name)

    // Function call would crash with NPE
    println(person.format())
}
```

Null safety

- In Kotlin, null is only allowed as value for nullable types of shape Type?
- Compiler enforces that no NPEs happen at runtime
- Safe navigation ?.
  - only call property / function if object is not null
- Elvis operator ?:
  - Take alternative when object is null
- Smart casting



Null safety.kt

```kotlin

data class Company(
    val companyName: String,
    val numberOfEmployees: Int,
)

data class Worker(
    val name: String,
    val company: Company,
)

private fun main() {
    // val worker : Worker = null

    val worker: Worker? = null

    // println(worker.name)

    println(worker?.name)
    println(worker?.company?.companyName)

    println(worker?.name ?: "Unknown")

    if (worker ≠ null) {
        println(worker.name)
    }
}
```

Nullability

Exercises

Functional programming

- **Functions as First-Class citizens**
  - Functions can be assigned to variables, passed as arguments, and returned from other functions
- **Higher-order Functions**
  - Functions that take other functions as parameters or return them
  - Common Higher-Order functions `map`, `filter`,`fold`, `reduce`
- **Lambdas**
  - Anonymous functions that can be treated as values

```kotlin
val greet: (String) -> String = { name -> "Hello, $name" }
println(greet("World")) // Outpur: Hello, World
```

```kotlin
fun <T, R> higherOrderFunction(func: (T) -> R, value: T): R {
  return func(value)
}
```

HigherOrderFunction.kt

```kotlin
fun main() {
  val double: (Int) -> Int = { it * 2 }
  println(higherOrderFunction(double, 5)) // Result: 10
}

fun <T, R> higherOrderFunction(func: (T) -> R, value: T): R {
  return func(value)
}
```

/sit schwarz IT

Functional Programming

Exercises

/sit schwarz IT

04

Lunch break + Q&A

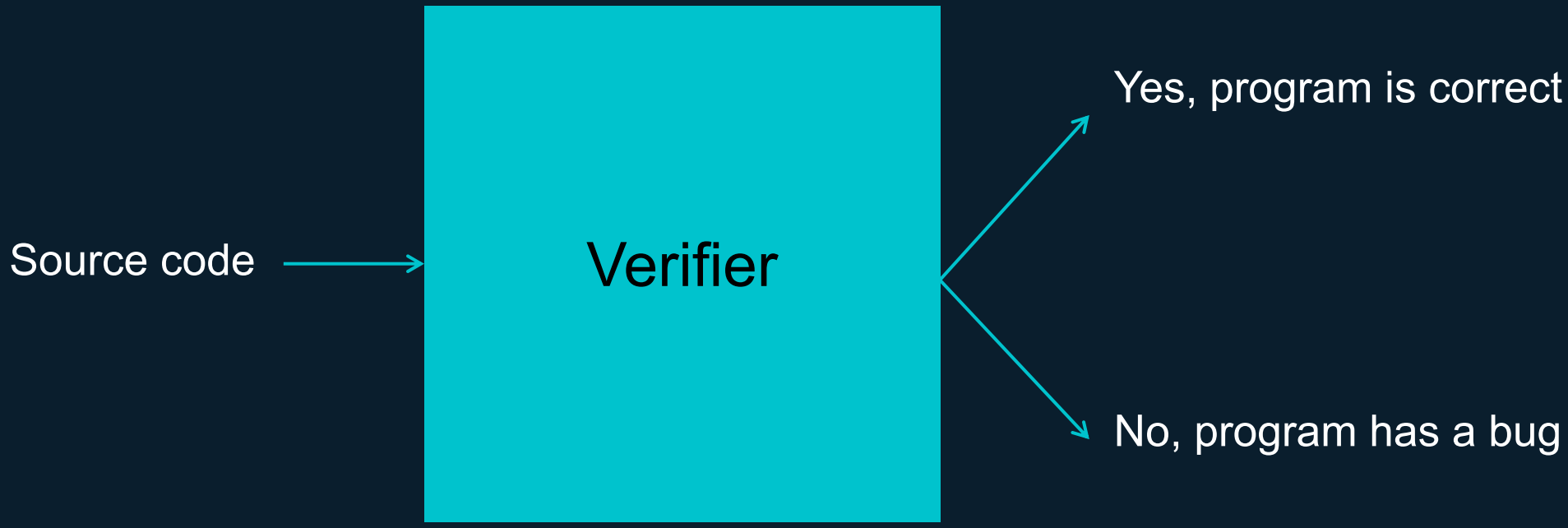
05

Afternoon session

/sit schwarz IT

05.a

Testing


Source code
Verifier
Yes, program is correct
No, program has a bug


Source code
Test scenario
Expected outcome
Test framework
Yes, outcome is as expected
No, program has a bug

SimpleTest.kt (1)

```kotlin
// Built-in assertions
// Only enabled with special flag
assert("kotlin".uppercase(Locale.GERMAN) ==
    "kotlin".uppercase(Locale.ENGLISH))

// Using Kotest testing framework
("kotlin".uppercase(Locale.GERMAN) ==
    "kotlin".uppercase(Locale.ENGLISH)) shouldBe true
```

SimpleTest.kt (2)

```
1  (
2    "kotlin".uppercase(Locale.of("tr", "TR")) ==
3      "kotlin".uppercase(Locale.ENGLISH)
4  ) shouldBe true
```

```
18
19      @Test  new *
20      fun `simple assert - false`() {
21          ("kotlin".uppercase(Locale.of( language: "tr",  country: "TR")) == "kotlin".uppercase(Locale.ENGLISH)) shouldBe true
22      }
```

✕ Stopped. Tests failed: 1, ignored: 6 of 7 tests – 21 ms

```
expected:<true> but was:<false>
Expected :true
Actual   :false
<Click to see difference>

io.kotest.assertions.AssertionFailedError: expected:<true> but was:<false>
    at schwarz.it.kotlin.workshop.basic.testing.SimpleTest.simple assert - false(SimpleTest.kt:21) <2 internal lines>
```

SimpleTest.kt (3)

```kotlin
"kotlin".uppercase(Locale.of("tr", "TR")) shouldBe
    "kotlin".uppercase(Locale.ENGLISH)
```

```kotlin
    @Test  new *
    fun `better assert - false`() {
        "kotlin".uppercase(Locale.of( language: "tr",  country: "TR")) shouldBe "kotlin".uppercase(Locale.ENGLISH)
    }
```

✕ Stopped. Tests failed: 1, ignored: 6 of 7 tests – 31 ms

```
/Users/muskalla/Library/Java/JavaVirtualMachines/corretto-21.0.3/Contents/Home/bin/java ...
Testing started at 16:05 ...

expected:<"KOTLIN"> but was:<"KOTLİN">
Expected :"KOTLIN"
Actual   :"KOTLİN"
<Click to see difference>

io.kotest.assertions.AssertionFailedError: expected:<"KOTLIN"> but was:<"KOTLİN">
    at schwarz.it.kotlin.workshop.basic.testing.SimpleTest.better assert - false(SimpleTest.kt:26) <2 internal lines>
```

Testing

- Use testing framework + assertion framework to write expressive tests (Here: kotest)
- Detecting regressions
- Code coverage
- Test driven development
- Automatic test execution as part of CI/CD

Several types of tests

- Unit tests: Test component in isolation
- Integration tests: Test interaction between components (requires starting web framework)

SimpleTest.kt (4)

```kotlin
class SimpleTest : AnnotationSpec() {

    @Test
    fun `another test`() {

        val mascots = listOf(
            "Duke",
            "Ferris",
            "Kodee",
            "Gopher"
        )

        "kodee" shouldBeIn
            mascots.map { it.lowercase() }
    }

}
```

WithDependency.kt

```kotlin
class DateService {
    fun getToday() = LocalDate.now()
}

class WorkingInfoService(
    private val dateService: DateService,
) {
    fun getWorkingsHours(day: DayOfWeek) =
        when (day) {
            MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY -> 9..17
            SATURDAY, SUNDAY -> IntRange.EMPTY
        }

    fun getWorkingHoursToday() =
      getWorkingsHours(dateService.getToday().dayOfWeek)
}

```

WithDependencyTest.kt

```kotlin
class WithDependencyTest : AnnotationSpec() {

    private val mockedDateService = mockk<DateService>()

    private val workingInfoService =
        WorkingInfoService(
            dateService = mockedDateService,
        )

    @Test
    fun `getWorkingHoursToday - weekend`() {

        every { mockedDateService.getToday() } returns
            LocalDate.of(2024, 7, 6)

        workingInfoService.getWorkingHoursToday() shouldHaveSize 0
    }
}
```

Testing

Exercises

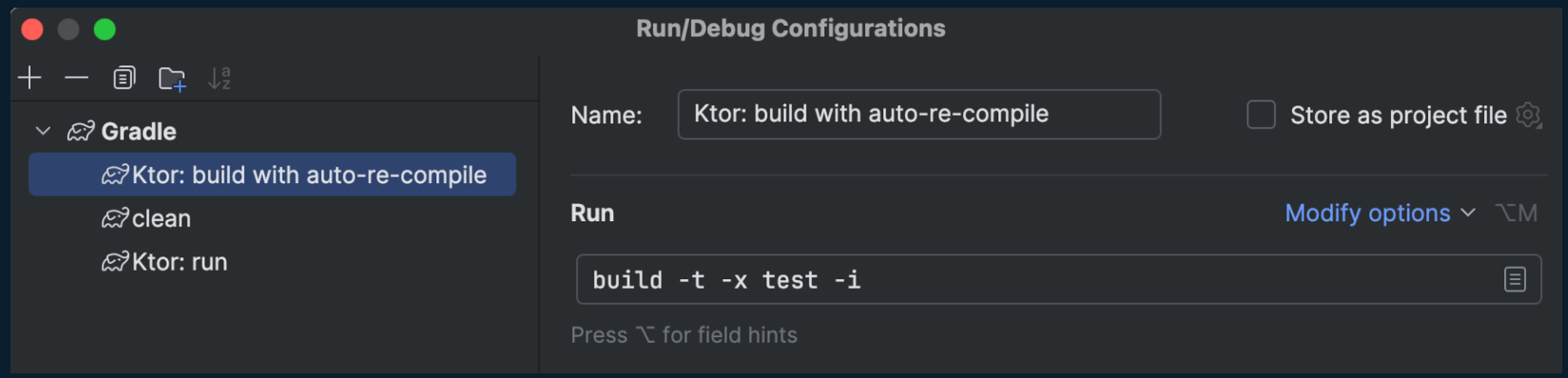
/sit schwarz IT

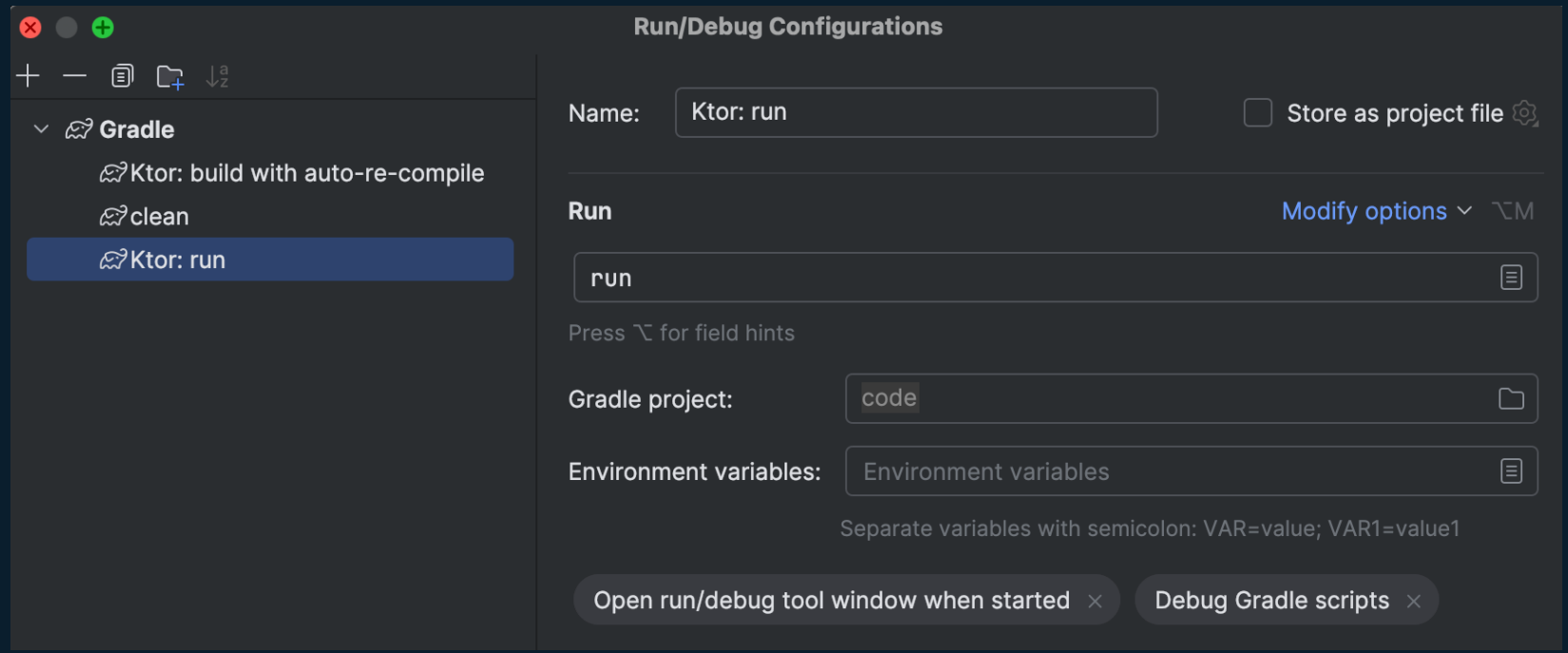
05.b

Web development

/sit schwarz IT

Ktor demo in IDE


Run/Debug Configurations
Gradle
Ktor: build with auto-re-compile
clean
Ktor: run
Name:
Ktor: build with auto-re-compile
Store as project file
Run
Modify options
⌥M
build -t -x test -i
Press ⌥ for field hints


Run/Debug Configurations
Gradle
Ktor: build with auto-re-compile
clean
Ktor: run
Name:
Ktor: run
Store as project file
Run
Modify options
⌥M
run
Press ⌥ for field hints
Gradle project:
code
Environment variables:
Environment variables
Separate variables with semicolon: VAR=value; VAR1=value1
Open run/debug tool window when started
Debug Gradle scripts

StaticRoutes.kt

```kotlin
fun Route.staticRoutes() {
    staticResources("/frontend", "static") {
        default("index.html")
    }
}
```

http://localhost:8080/frontend/index.html

BasicRoutes.kt (1)

```kotlin
fun Route.basicRoutes() {
    route("/basic") {
        get("/static.html") {
            call.respondText(
              contentType = ContentType.Text.Html,
              status = HttpStatusCode.OK
            ) {
                """
                <html>
                <body>
                    <h1>Welcome at Kodee's online shop</h1>
                </body>
                </htmL>
                """.trimIndent()
            }
        }
    }
}
```

http://localhost:8080/basic/static.html

BasicRoutes.kt (2)

```kotlin
get("/dynamic.html") {
    val totalOrders = Random.nextInt(from = 5, until = 20)
    val inDelivery = Random.nextInt(from = 2, until = totalOrders / 2)

    call.respondText(
      contentType = ContentType.Text.Html,
      status = HttpStatusCode.OK
    ) {
        """
        <html>
        <body>
            <h1>Welcome at Kodee's online shop</h1>
            Open orders at ${LocalDateTime.now()}:
                $totalOrders open,
                $inDelivery of which in delivery
        </body>
        </html>
        """.trimIndent()
    }
}
```

http://localhost:8080/basic/dynamic.html

Web technologies

The beginnings:

- Web server delivers static HTML

Later:

- Dynamic content: Web server renders page on request

15-25 years ago: AJAX etc.

- Browser and backend communicate asynchronously via API
- REST
- JSON

ShopStatisticsDTO.kt

```kotlin
@Serializable
data class ShopStatisticsDTO(
    val timestamp: String,
    val totalOrders: Long,
    val ordersInDelivery: Long,
)
```

BasicRoutes.kt (3)

```kotlin
get("/decoupled") {
    val totalOrders = Random.nextLong(from = 5, until = 20)
    val inDelivery = Random.nextLong(from = 2, until = totalOrders / 2)

    val response =
        ShopStatisticsDTO(
            timestamp = LocalDateTime.now().toString(),
            totalOrders = totalOrders,
            ordersInDelivery = inDelivery,
        )

    call.respond(HttpStatusCode.OK, response)
}
```

http://localhost:8080/basic/decoupled

http://localhost:8080/frontend/basic.html

BasicRoutes.kt (4)

```kotlin
get(
    "/decoupled",
    {
        description = "get current shop statistics"
        response {
            HttpStatusCode.OK to {
                description = "Successful request"
                body<ShopStatisticsDTO> {
                    description = "the current statistics"
                }
            }
        }
    },
) {
  // ...
}
```

http://localhost:8080/swagger

BasicRoutesTest.kt

```kotlin
class BasicRoutesTest : AnnotationSpec() {

    @Test
    fun `endpoint decoupled - returns statistics`() = testApplication {

          val client = createClient {
              install(ContentNegotiation) {
                  json()
              }
          }

          val response = client.get("/basic/decoupled")

          response.status shouldBe HttpStatusCode.OK

          val body = response.body<ShopStatisticsDTO>()

          body.totalOrders shouldBeLessThan 20
          body.ordersInDelivery shouldBeLessThan body.totalOrders
          body.ordersInDelivery shouldBeGreaterThanOrEqual 2
      }
}
```

Advanced project demo in IDE / browser

http://localhost:8080/frontend/advanced.html

AdvancedRoutes.kt (1)

```kotlin
get("/order/{identifier}") {
    val identifier = call.parameters["identifier"]?.toLongOrNull()

    if (identifier == null) {
        call.respond(HttpStatusCode.BadRequest)
        return@get
    }

    val orderInfo = orders[identifier]

    if (orderInfo == null) {
        call.respond(HttpStatusCode.NotFound)
        return@get
    }

    call.respond(HttpStatusCode.OK, orderInfo.toDTO())
}
```

http://localhost:8080/frontend/advanced.html

AdvancedRoutes.kt (2)

```kotlin
post("/order") {
    val dto = call.receive<OrderCreationDTO>()

    if (dto.cost < 0 || dto.cost > 1_000) {
        call.respond(HttpStatusCode.BadRequest)
        return@post
    }

    if (dto.deliveryDay == DayOfWeek.SUNDAY) {
        call.respond(HttpStatusCode.BadRequest)
        return@post
    }

    val identifier = // ...

    orders[identifier] = // ...
    call.respond(HttpStatusCode.Created, identifier)
}
```

http://localhost:8080/frontend/advanced.html

Web development

Exercises

/sit schwarz IT

05.c

ORM

ORM

- Backend typically uses a database as persistent storage
- Problem: Structure of entries in databases ≠ structure of Kotlin objects
- Use ORM (object–relational mapping) framework

Here:

- H2: Lightweight, in-memory database
- Exposed: lightweight Kotlin-native ORM framework

Article.kt

```kotlin
data class Article(
    val identifier: String,
    val name: String,
    val description: String,
)

object Articles : Table("articles") {
    val identifier = varchar("identifier", 255).uniqueIndex()
    val name = varchar("name", 255)
    val description = text("description")

    override val primaryKey = PrimaryKey(identifier, name = "PK_Articles")
}

fun ResultRow.toArticle() =
    Article(
        identifier = this[Articles.identifier],
        name = this[Articles.name],
        description = this[Articles.description],
    )
```

http://localhost:8080/frontend/orm.html

Article.kt (Usage)

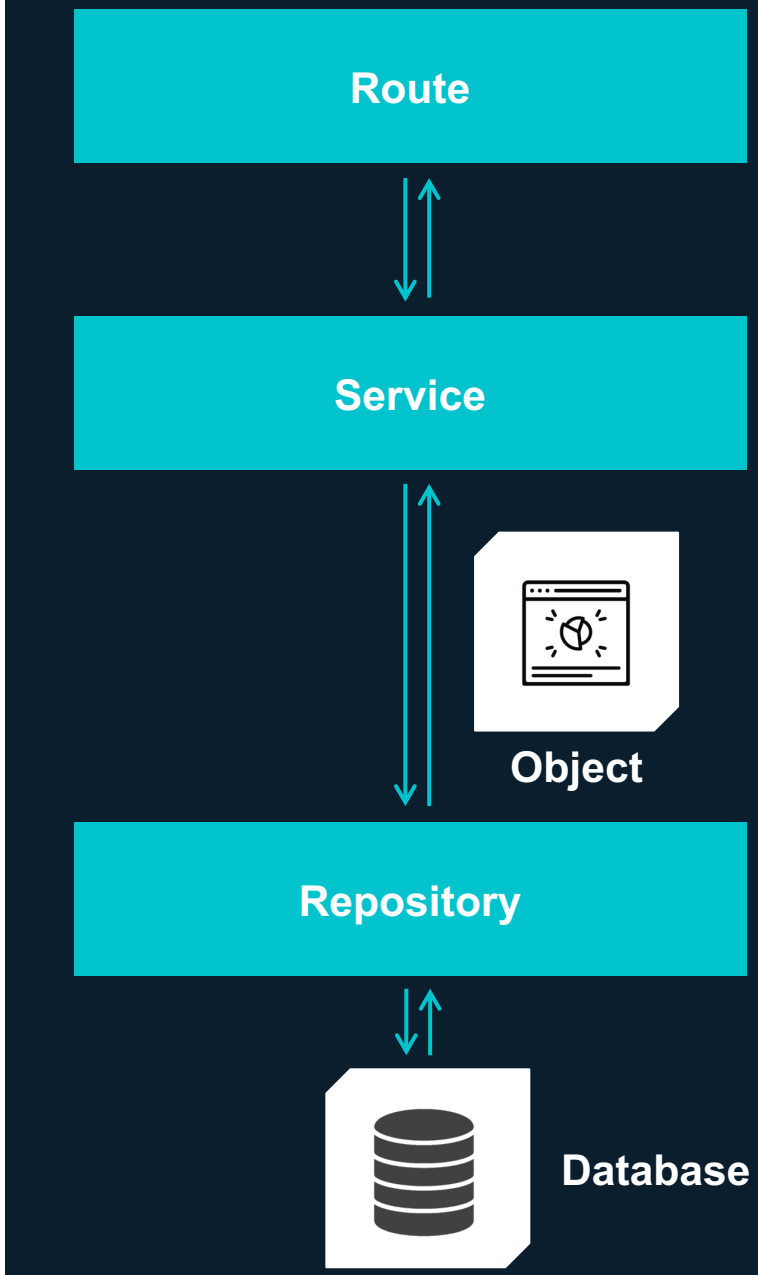
```kotlin
fun findArticleByIdentifier(identifier: String) =
    transaction {
        Articles
            .selectAll()
            .where { Articles.identifier eq identifier }
            .map { it.toArticle() }
            .singleOrNull()
    }
```

http://localhost:8080/frontend/orm.html

Layered architecture

Separation of concerns

- Entity: Object that is stored in database
- Repository: Access layer for database
- Service: (Business) logic
  - uses Repository to access data
- Route (Controller / Resource): Provides API
  - uses Service to invoke logic



ORM project demo in IDE / browser

http://localhost:8080/frontend/orm.html

ORM

Exercises

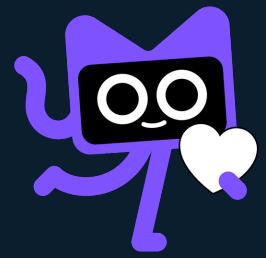
06

Q&A

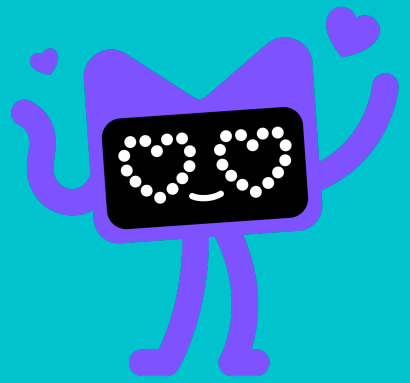
Readings

A Comprehensive Kotlin Learning Guide for All Levels | The JetBrains Academy Blog

Kotlin Onboarding: Introduction | The JetBrains Academy Blog



Thank you for attending!



Sebastian Muskalla
Digital Business Solutions
Schwarz IT

Sebastian.Muskalla@mail.schwarz

Suman Venkat
Digital Business Solutions
Schwarz IT

Suman.Venkat@mail.schwarz