# Devoxx Antwerpen

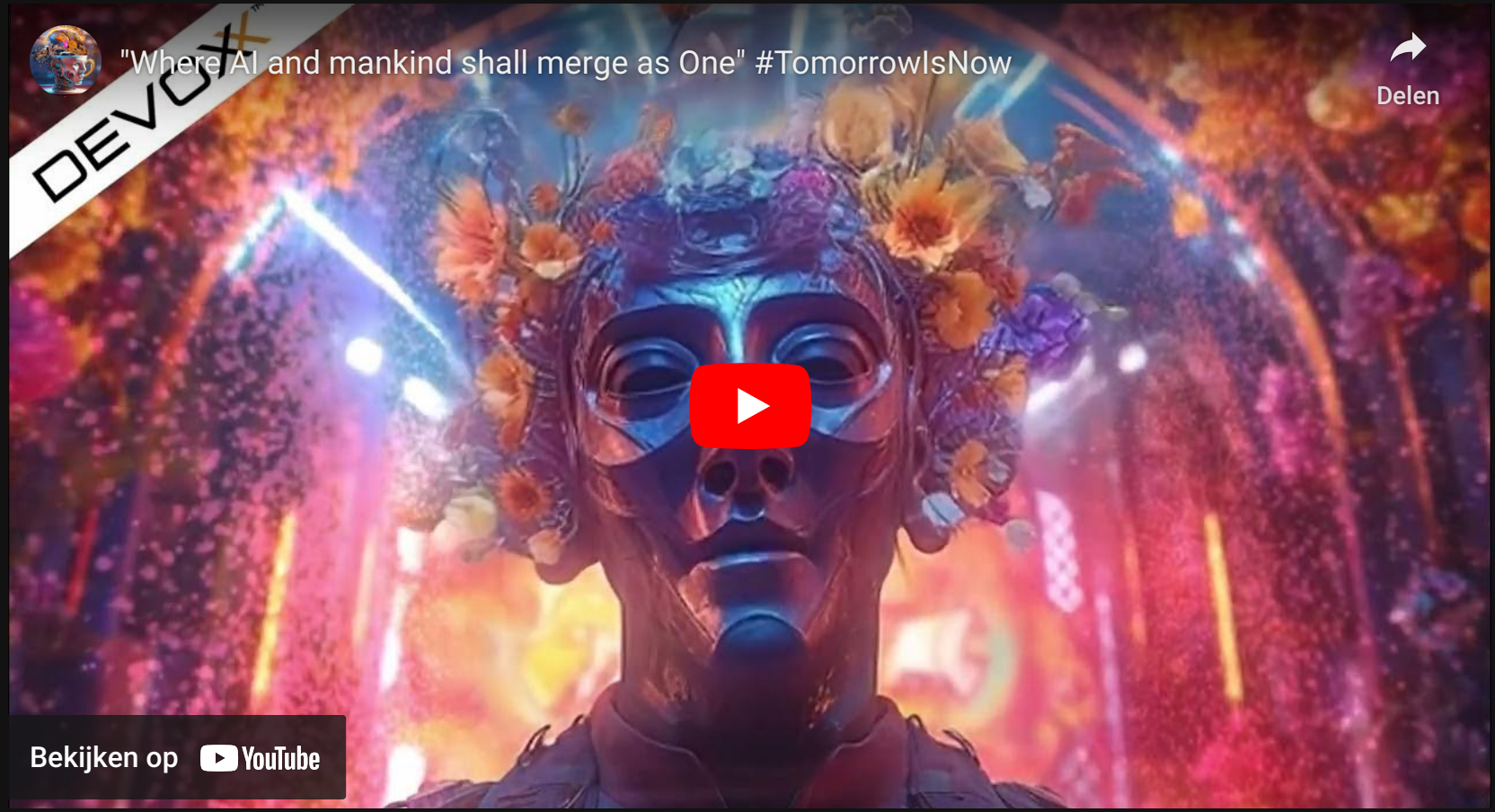
Marcel Schutte - Groningen - 3 november 2023

# Who am I?

- **Title** - Lead Solution Architect
- **Assignment** - KOOP in Den Haag
- **Sogeti** - Member of Java Architecture Board, Java Education Team and Technology Board NoordOost

**Personal**

- **Age** 41 years
- **Marital status** Married, 2 kids
- **Hobbyies** Running, geocaching, gardening




https://www.youtube.com/watch?v=njs7bq0oftg

# What is Devoxx?

- Biggest Java conference in Europe(the world?)
- Yearly event about all that is Java and techniques/methodologies
- Hundreds of sessions and speakers

# Bonus:

- Catching up with former colleagues
- Networking with big java related companies
- Getting to know current colleagues
- Lots and lots of energy to try out all the new and cool stuff

# Agenda

1. Java 21
2. VS Code Support
3. Sustainability
4. AI assistants
5. Misc. interesting sessions

# Java 21

1. Accessability 👉
2. Virtual Threads
3. Miscellaneous features

## Unnamed classes & Instance main methods

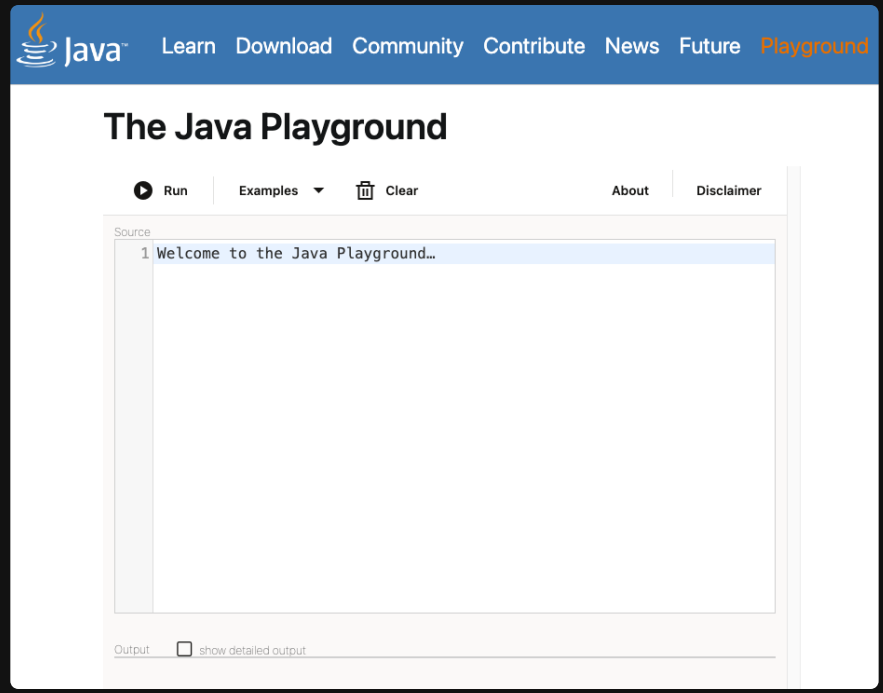
```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}

void main() {
    System.out.println("Hello, World!");
}
```

# Java 21

1. Accessability 👉
2. Virtual Threads
3. Miscellaneous features

Java Playground at Dev.java




https://dev.java/playground

# Java 21

1. Accessability
2. Virtual Threads 👉
3. Miscellaneous features

## Project Loom

Virtual threads

- Lighter threads
- min. size -> 200-300 bytes
- scales to 1M+ threads on commodity hardware

Virtual Threads are real Threads

- implement java.lang.Thread and support ThreadLocal
- clean stacktraces
- sequential step debugging
- threaded code just works
- Thread without the bagage

# Java 21

1. Accessability
2. Virtual Threads 👉
3. Miscellaneous features

## Virtual threads (JEP 444)

```
try (var executor =
        Executors.newVirtualThreadPerTaskExecutor()) {
    IntStream.range(0, 10_000).forEach(i → {
        executor.submit(() → {
            Thread.sleep(Duration.ofSeconds(1));
            return i;
        });
    });
}  // executor.close() is called implicitly, and waits
```

# Java 21

1. Accessability
2. Virtual Threads
3. Miscellaneous features 👉

## String templates

```
// Embedded expressions can be strings
String firstName = "Bill";
String lastName  = "Duck";
String fullName  = STR."\{firstName} \{lastName}";
| "Bill Duck"
String sortName  = STR."\{lastName}, \{firstName}";
| "Duck, Bill"

// Embedded expressions can perform arithmetic
int x = 10, y = 20;
String s = STR."\{x} + \{y} = \{x + y}";
| "10 + 20 = 30"

// Embedded expressions can invoke methods/access fields
String s =
  STR."You have a \{getOfferType()} waiting for you!";
| "You have a gift waiting for you!"
```

# Java 21

1. Accessability
2. Virtual Threads
3. Miscellaneous features 👉

## Pattern matching for switch

```java
// As of Java 21
static String formatterPatternSwitch(Object obj) {
    return switch (obj) {
        case Integer i -> String.format("int %d", i);
        case Long l    -> String.format("long %d", l);
        case Double d  -> String.format("double %f", d);
        case String s  -> String.format("String %s", s);
        default        -> obj.toString();
    };
}
```

# Java 21

1. Accessability
2. Virtual Threads
3. Miscellaneous features 👉

## Switches and null

```java
// Prior to Java 21
static void testFooBarOld(String s) {
    if (s == null) {
        System.out.println("Oops!");
        return;
    }
    switch (s) {
        case "Foo", "Bar" -> System.out.println("Great");
        default           -> System.out.println("Ok");
    }
}

// As of Java 21
static void testFooBarNew(String s) {
    switch (s) {
        case null         -> System.out.println("Oops");
        case "Foo", "Bar" -> System.out.println("Great");
        default           -> System.out.println("Ok");
    }
}
```

# Java 21

1. Accessability
2. Virtual Threads
3. Miscellaneous features 👉

## Case refinement

```
// Prior to Java 21
static void testStringOld(String response) {
    switch (response) {
        case String s → {
            if (s.equalsIgnoreCase("YES"))
                System.out.println("You got it");
            else
                System.out.println("Sorry?");
        }
    }
}

// As of Java 21
static void testStringNew(String response) {
    switch (response) {
        case String s
        when s.equalsIgnoreCase("YES") → {
            System.out.println("You got it");
        }
        case String s → {
            System.out.println("Sorry?");
        }
    }
}
```

# Visual Studio Code

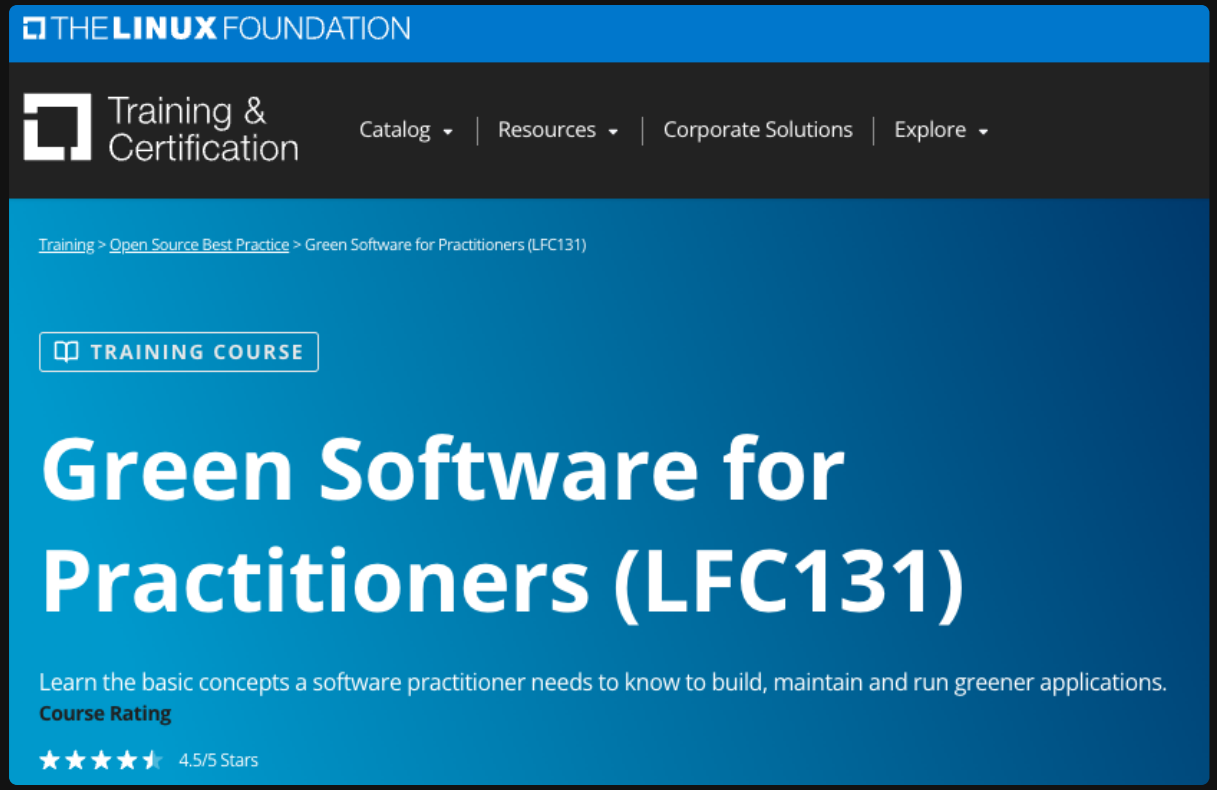
Oracle Java Platform Extension for Visual Studio Code

## Features

- project view
- auto-complete
- error highlighting
- jump to definition
- some forms of automated refactoring
- JavaDoc-on-hover
- debugging support
- unit-testing support for JUnit
- support for Gradle and Maven projects.


Backspace
Hom
4

# Sustainability certification




https://training.linuxfoundation.org/training/green-software-for-practitioners-lfc131/

# AI Coding assistants

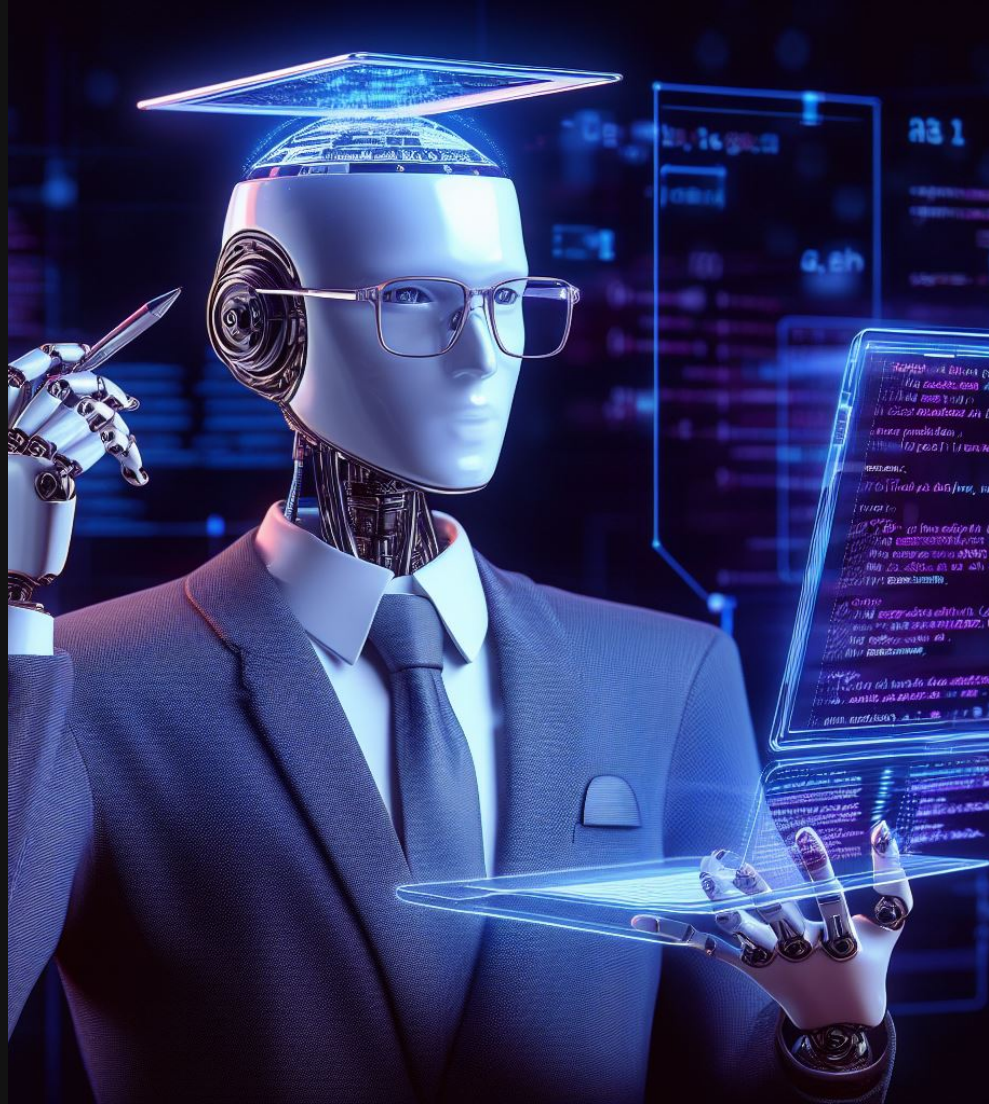
Coding will never be the same

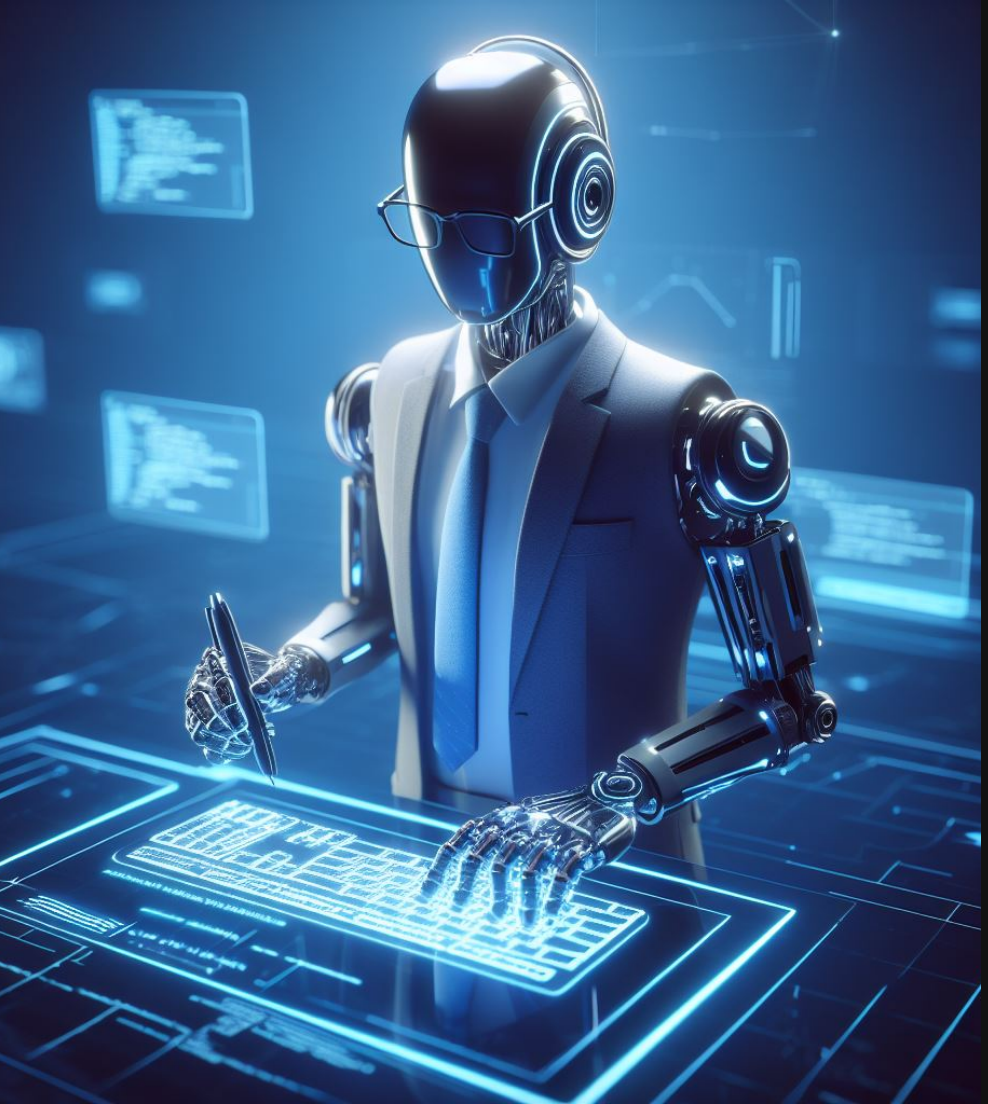
## Already available

- Amazon CodeWhisperer (Free)
- Github Copilot
- Tabnine

## Available soon

- Github Copilot Chat
- Tabnine Chat
- JetBrains AI assistant





# JetBrains AI Assistant

Technical preview with limited access

## Functionality

- AI Chat
- Generate documentation
- Generate commit messages
- Code explanation
- Refactoring
- Suggest bugfixes
- Write unit tests

# Imposter Syndrome

Who?
Everyone!
Is that a problem?
No!

Embracing Imposter Syndrome
By Dom Hodgson

https://devoxx.be/talk/?id=70101

# C4 models as Code

Advantages:

- multiple diagrammes from one model
- alternative visualisations
- versioning

C4 models as code

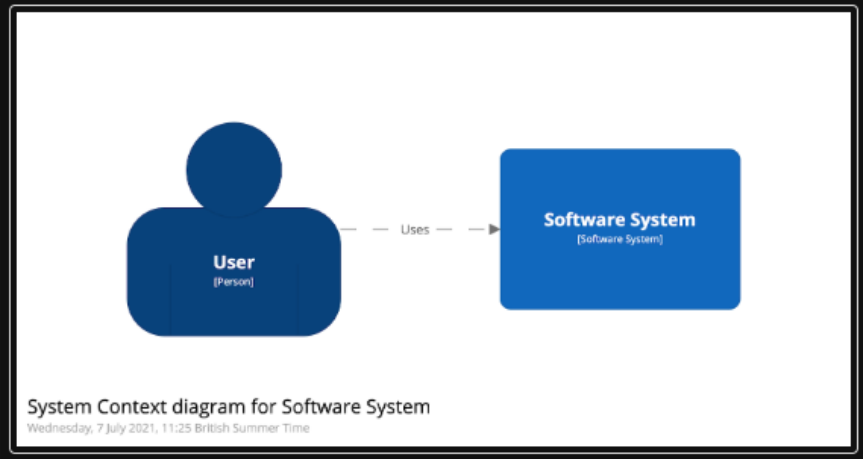
By Simon Brown

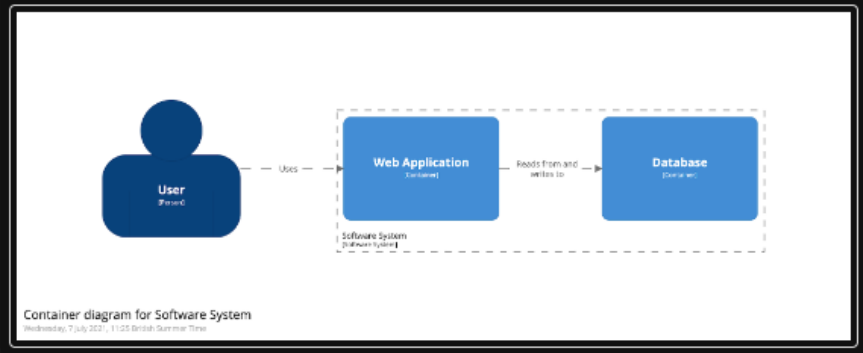
https://devoxx.be/talk/?id=28357

```
workspace {
    model {
        user = person "User"
        softwareSystem = softwareSystem "SW System" {
            webapp = container "Web Application" {
                user → this "Uses"
            }
            container "Database" {
                webapp → this "Reads from and writes to"
            }
        }
    }
    views {
        systemContext softwareSystem {
            include *
            autolayout lr
        }
        container softwareSystem {
            include *
            autolayout lr
        }
    }
}
```

# C4 models as Code

```
workspace {
    model {
        user = person "User"
        softwareSystem = softwareSystem "SW System" {
            webapp = container "Web Application" {
                user → this "Uses"
            }
            container "Database" {
                webapp → this "Reads from and writes to"
            }
        }
    }
    views {
        systemContext softwareSystem {
            include *
            autolayout lr
        }
        container softwareSystem {
            include *
            autolayout lr
        }
    }
}
```



# Building a Bullsh*t Language

Annotationscript

## What?

- based on Java annotations
- turing complete

## Why?

- Why not?
- Because we can!

```
@Zero("begin")
@Zero(list={
  @One("define"),
  @One("twice"),
  @One(list={
    @Two("lambda"),
    @Two(list={@Three("x")}),
    @Two(list={@Three("*"), @Three("x"), @Three("2")})})})
@Zero(list={
  @One("+"),
  @One(list={@Two("twice"), @Two("x")}),
  @One("1")})
class Example {}
```

Building a Bullsh*t Language

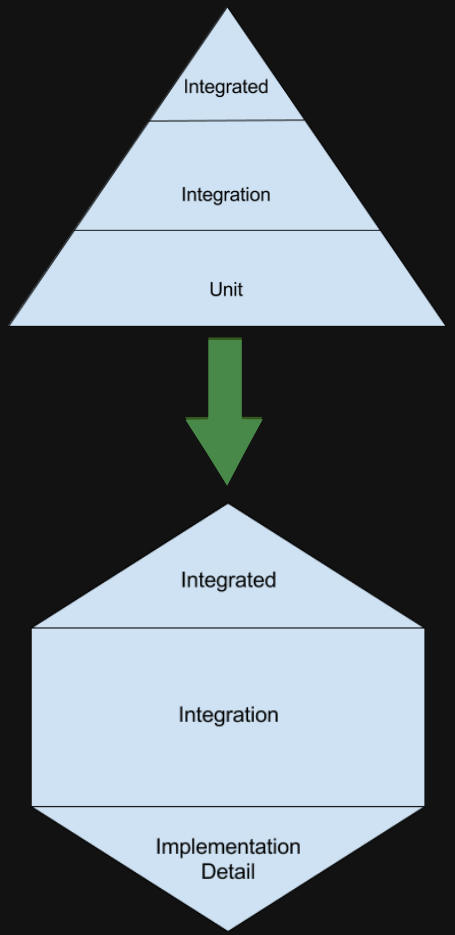
By Jan Ouwens

https://devoxx.be/talk/?id=31405

# Testing microservices

## Integration testing challenges

- Test Coupling
- Flaky tests
- Slow tests
- Cognitive load
- Investment -> CI, libs, training




Testing Microservices - Join the Revolution

By Victor Rentea

https://devoxx.be/talk/?id=30503

# Tests need architecting

- Naming conventions
- Object Mother

Your tests also need some architecting

By Jonas Geiregat

https://devoxx.be/talk/?id=2914

```
Address billingAddress = new Address.Builder()
        .streetAddress("123 Main St")
        .city("Anytown")
        .state("CA")
        .zipCode("12345")
        .country(Country.US)
        .build();
Address shippingAddress = new Address.Builder()
        .streetAddress("456 Oak Ave")
        .city("Othertown")
        .state("CA")
        .zipCode("67890")
        .country(Country.US)
        .build();
List<InvoiceItem> items = new ArrayList<>();
items.add(new InvoiceItem("Product A", Amount.EUR(100.00),
items.add(new InvoiceItem("Product B", Amount.EUR(100.00),
Customer customer = new Customer.Builder()
        .name("John Doe")
        .email("john.doe@example.com")
        .phoneNumber("555-123-4567")
        .build();
Invoice invoice = new Invoice(
        new InvoiceNumber("001"),
        customer,
        billingAddressm,
        shippingAddress,
        LocalDate.now(),
```

# Tests need architecting

```
Address billingAddress = new Address.Builder()
        .streetAddress("123 Main St")
        .city("Anytown")
        .state("CA")
        .zipCode("12345")
        .country(Country.US)
        .build();
Address shippingAddress = new Address.Builder()
        .streetAddress("456 Oak Ave")
        .city("Othertown")
        .state("CA")
        .zipCode("67890")
        .country(Country.US)
        .build();
List<InvoiceItem> items = new ArrayList<>();
items.add(new InvoiceItem("Product A", Amount.EUR(100.00), Tax.vatPercentage(21)));
items.add(new InvoiceItem("Product B", Amount.EUR(100.00), Tax.vatPercentage(21)));
Customer customer = new Customer.Builder()
        .name("John Doe")
        .email("john.doe@example.com")
        .phoneNumber("555-123-4567")
        .build();
Invoice invoice = new Invoice(
        new InvoiceNumber("001"),
        customer,
```

# Tests need architecting

```
public class InvoiceMother {

    private InvoiceMother() { }

    public static Builder invoice() {
        return new Builder();
    }

    public static class Builder {

        InvoiceNumber invoiceNumber = new InvoiceNumber("001");
        Customer customer = CustomerMother.customer().build();
        Address billingAddress = AddressMother.address().build();
        Address shippingAddress = AddressMother.address().build();
        LocalDate creationDate = LocalDate.now();
        List<InvoiceItem> items = List.of(InvoiceItemMother.item().build());

        public Builder withItems(List<InvoiceItem> items) {
            this.items = items;
            return this;
        }

        // other setters are left out for brevity

        public Invoice build() {
```

# Tests need architecting

```
Invoice invoice = InvoiceMother.invoice()
        .withInvoiceItems(
            new InvoiceItem("Product A", Amount.EUR(100.00), Tax.vatPercentage(21)),
            new InvoiceItem("Product B", Amount.EUR(100.00), Tax.vatPercentage(21)))
        .build();

Amount amount = invoice.getVatAmount();

assertThat(amount).isEqualTo(Amount.EUR(42))
```

# SoCode 2023

Yearly funprogging context

- Starts december 1st at 6.00AM
- 2 puzzles per day

Extras

- Discordserver for discussion and showing off cool solutions
- Leaderboard especially for participants of Sogeti and CapGemini
- Cool prizes!

Kickoff: monday 27 november at 16:00 PM

Registration:
https://forms.microsoft.com/e/u8zxgvT0fW

# Thank you for your attention!


linkedin.com/in/marceljschutte

@PragArchitect

marcel.schutte@sogeti.com

+31 652 659 277


https://github.com/marceljschutte/slidedecks/blob/main/20231103-java-friday-noordoost-devoxx-eng.pdf

END