#0

Java EE

Enterprise & Java Features

"Enterprise? Java Versioner? Roadmaps? Överlämning?
Future of Java == NULL?"

INNEHÅLLSFÖRTECKNING

INNEHÅLLSFÖRTECKNING

Kursplan

Förväntningar?

Utbildningsmoment:

- MVC i Java och databaser
- Webb Applikationer
- CRUD-funktionalitet
- Teoretiska och praktiska övningar
- Java EE
- Databaser
- Säkerhet

01

ÖVERSIKT


Is Java Still Relevant?

Talk LAB

3 min

Discuss the following topics:

Vad är Kotlin? Kommer Kotlin slå ut Java?

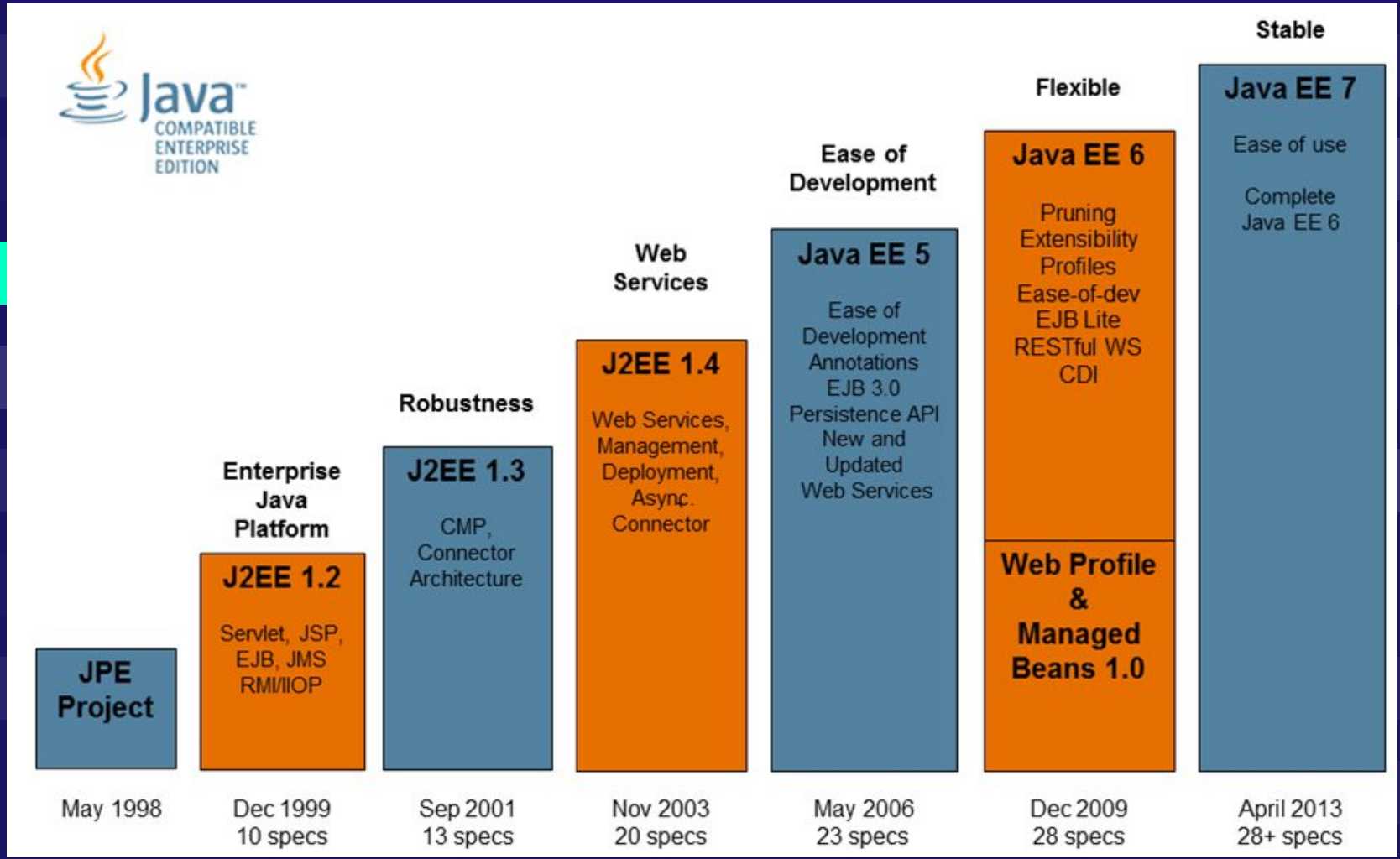
Vad tror ni om framtiden för Java? C# då?

Vad är Enterprise?


Vad är Enterprise?

“Java Platform Enterprise Edition (Java EE), *the standard in community-driven enterprise software, is* developed using the Java Community Process.”

- Oracle

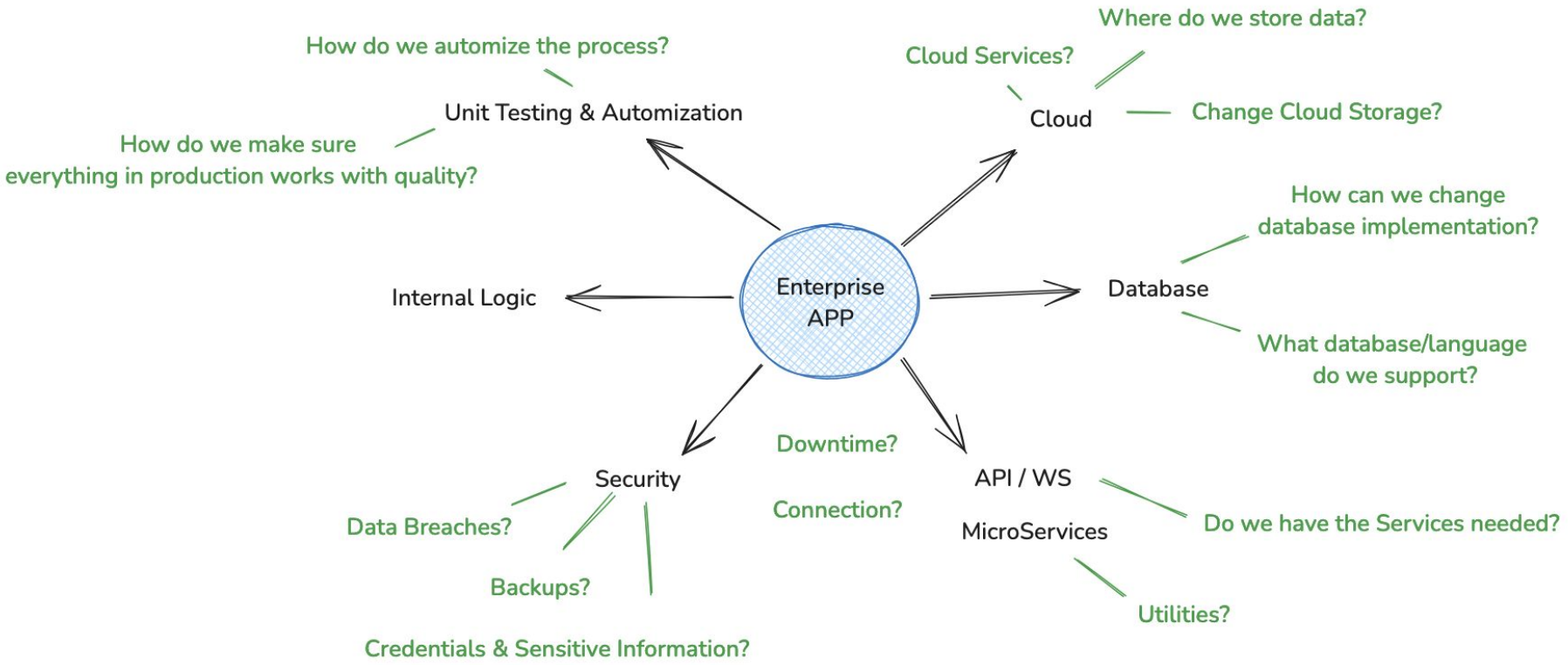

Java
COMPATIBLE
ENTERPRISE
EDITION
JPE
Project
May 1998
Enterprise
Java
Platform
J2EE 1.2
Servlet, JSP,
EJB, JMS
RMI/IIOP
Dec 1999
10 specs
Robustness
J2EE 1.3
CMP,
Connector
Architecture
Sep 2001
13 specs
Web
Services
J2EE 1.4
Web Services,
Management,
Deployment,
Async.
Connector
Nov 2003
20 specs
Ease of
Development
Java EE 5
Ease of
Development
Annotations
EJB 3.0
Persistence API
New and
Updated
Web Services
May 2006
23 specs
Flexible
Java EE 6
Pruning
Extensibility
Profiles
Ease-of-dev
EJB Lite
RESTful WS
CDI
Web Profile
&
Managed
Beans 1.0
Dec 2009
28 specs
Stable
Java EE 7
Ease of use
Complete
Java EE 6
April 2013
28+ specs

Vad är Enterprise?

Projekt i fokus

How can we make this BETTER?

What happens if the postgreSQL database disappears?
What if our data is compromised?
What if our database is down?
What if we get DDOS'd
What if we...



Vad är Enterprise?

Projekt i fokus

How can we make this BETTER?

What happens if the postgreSQL database disappears?
What if our data is compromised?
What if our database is down?
What if we get DDOS'd
What if we...

Where do we store data?

How do w

Cloud Storage?

How do we make sure
everything in production works with qu

How can we change
atabase implementation?

Int

What database/language
do we support?

SKALBARHET
SCALABILITY

Connection?

Data Breaches?

MicroServices

Do we have the Services needed?

Backups?

Utilities?

Credentials & Sensitive Information?


JAVA
What's OLD?
What's NEW?

*“This report **summarizes** how the **InfoQ Java** editorial **team** currently **sees** the **adoption** of **technology** and **emerging trends** within the **Java** space”*

Source:
https://www.infoq.com/articles/java-trends-report-2024/

2020


Software Development
Java and JVM 2020 Q3 Graph
http://infoq.link/java2020
InfoQ
JPMS: Adopting Modularity
Java on ARM
Microprofile
Helidon
Quarkus
Jakarta EE
JUnit 5
Static compilation
CHASM
Java 11
Kotlin
JRuby (niche uptake)
OpenJDK/Non-Oracle builds
Java 8
Clojure (niche uptake)
Groovy
Scala
Spring
Hibernate
Innovators
Early Adopters
Early Majority
Late Majority

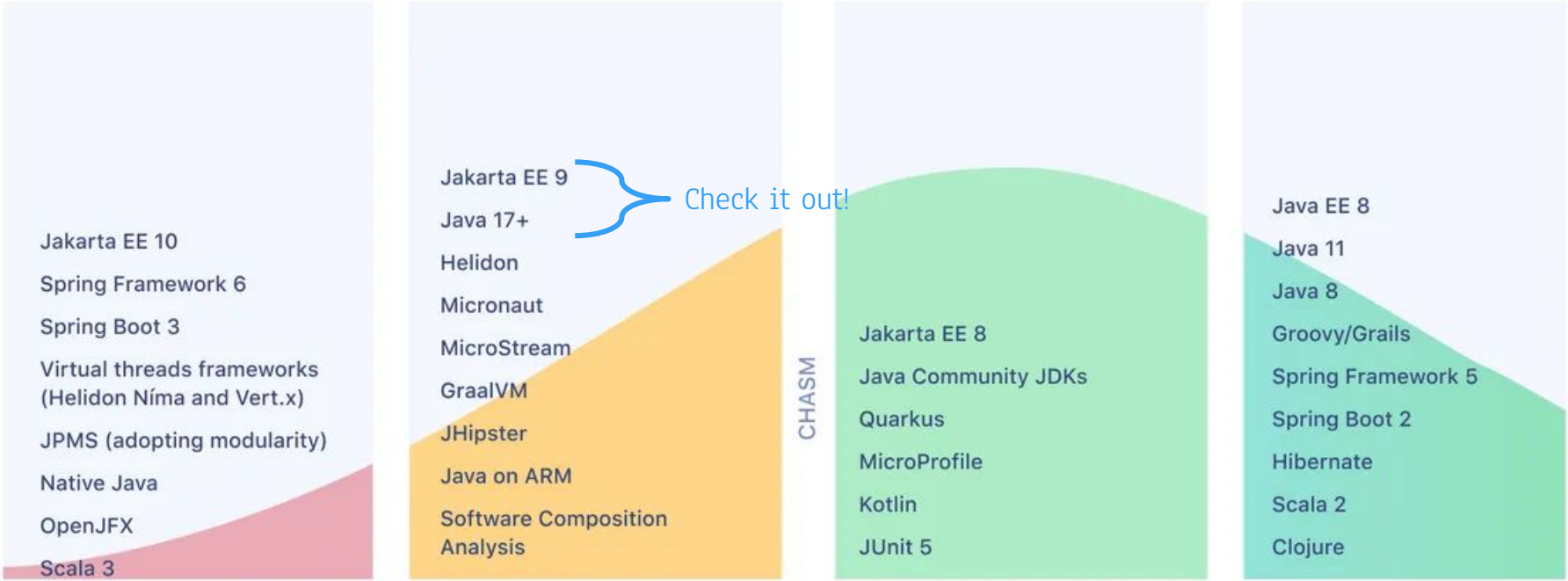
2021

Source: https://www.infoq.com/articles/java-jvm-trends-2021/

2022

Software Development
Java and JVM 2022 Q4 Graph
http://infoq.link/java2022

InfoQ




InfoQ
Software Development
InfoQ Java Trends Report - November 2023
2023
Read the full article on InfoQ.com
Jakarta EE 10
Java 21
Virtual Threads Frameworks (Helidon Níma and Vert.x)
Fast JVM Startup (CRaC)
OpenJFX
Scala 3
Prompt Engineering (Spring AI/Semantic Kernel)
Code Generation (Generative AI)
Jakarta EE 9
Spring Boot 3
Java 17
Helidon
Micronaut
MicroStream
Fast JVM Startup (GraalVM)
JHipster
Java on ARM
Software Composition Analysis
CHASM
Jakarta EE 8
Java Community JDKs
Quarkus
MicroProfile
Kotlin
JUnit 5
Java EE 8
Java 11
Java 8
Groovy/Grails
Spring Boot 2
Hibernate
Scala 2
Clojure
Innovators
Early Adopters
Early Majority
Late Majority


InfoQ
Java Trends Report – 2024
https://www.infoq.com/articles/java-trends-report-2024
2024
Read full article on InfoQ.com
Java 23+
Jakarta EE 11
Prompt Engineering (LangChain4j, Spring AI and Semantic Kernel)
Code Generation (Generative AI)
GraalPy
GraalWasm
Java 21
Jakarta EE 10
Micronaut
Eclipse Store (formerly MicroStream)
Fast JVM Startup (CRaC)
Virtual Threads Frameworks (Helidon Nima and Vert.x)
JHipster
Java on ARM
Software Composition Analysis
CHASM
Java 17
Jakarta EE 9
Spring Framework 6/Spring Boot 3
Quarkus
MicroProfile
Helidon
Java Community JDKs
Fast JVM Startup (GraalVM)
Kotlin
JUnit 5
Java 11
Java 8
Jakarta EE 8
Java EE 8
Groovy/Grails
Spring Boot 2
Hibernate
Scala 3
OpenJFX
Clojure
Innovators
Early Adopters
Early Majority
Late Majority

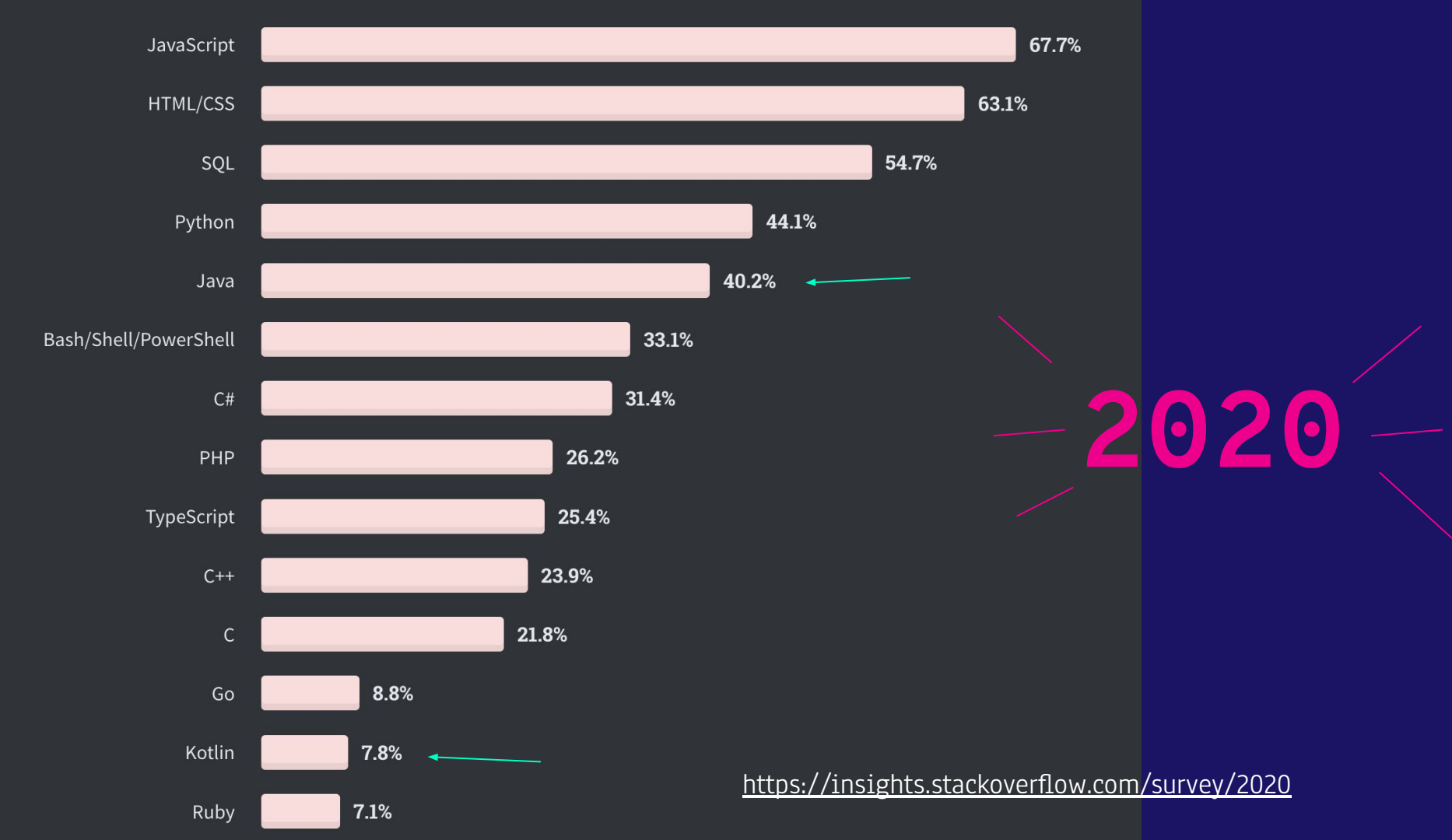
Work Lab

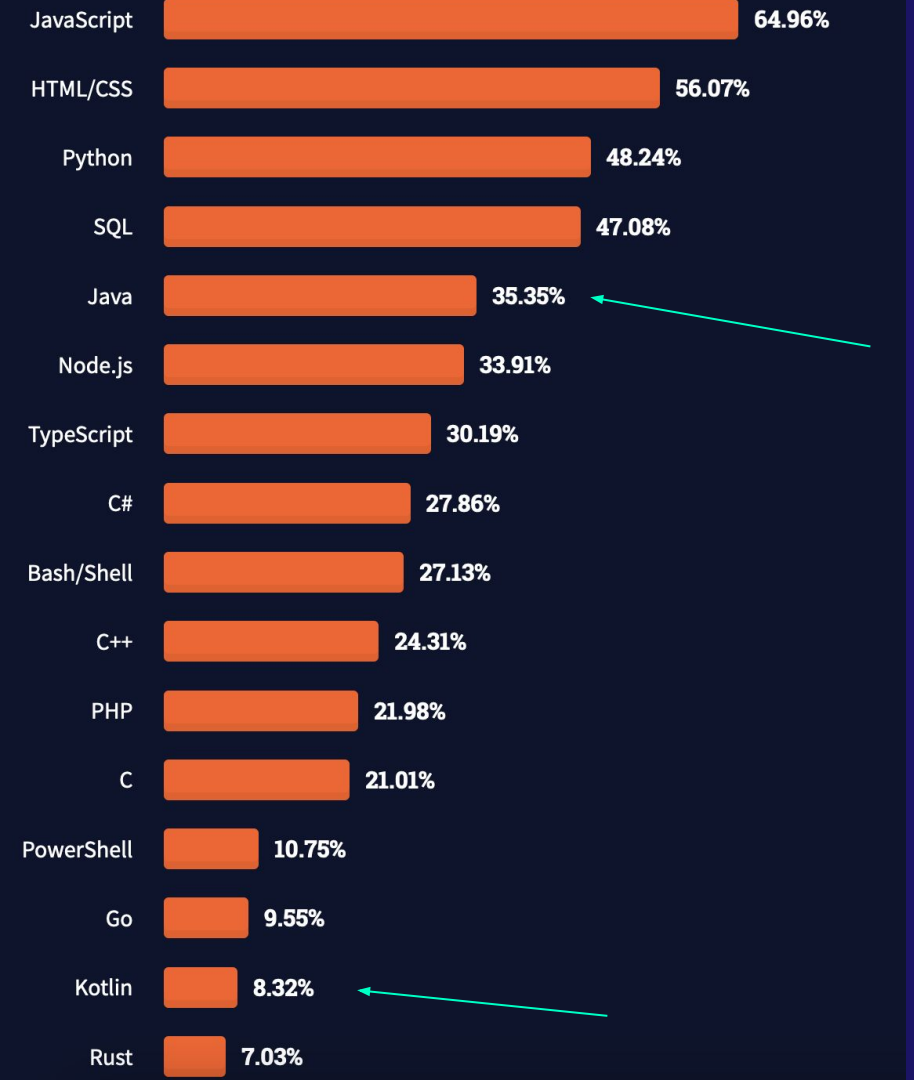
Statistik

Kolla in ‘tech’ på:
https://survey.stackoverflow.co/2024/technology

Vad säger Stackoverflow

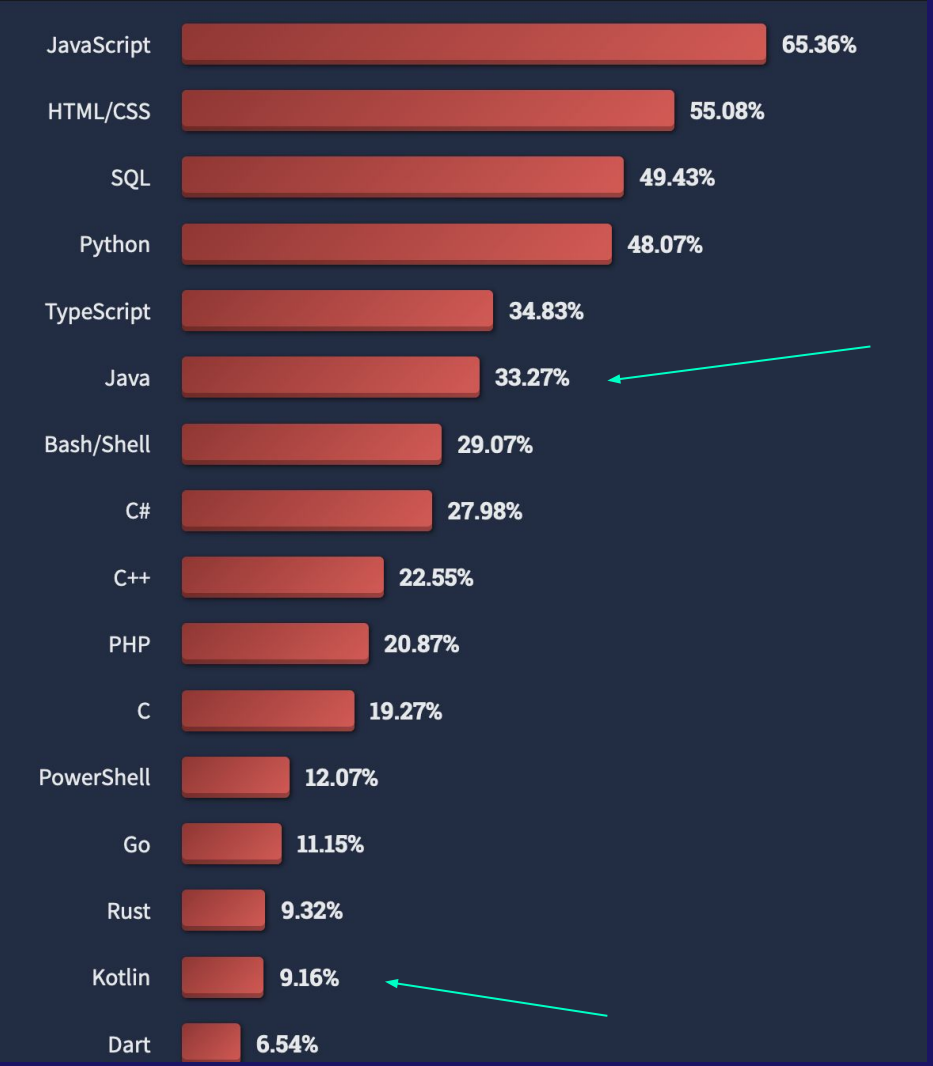
“Här visas data om personer som kan svara på flera frågor. Procenten kan förvirra en utan denna vetskap”


JavaScript
67.7%
HTML/CSS
63.1%
SQL
54.7%
Python
44.1%
Java
40.2%
Bash/Shell/PowerShell
33.1%
C#
31.4%
PHP
26.2%
TypeScript
25.4%
C++
23.9%
C
21.8%
Go
8.8%
Kotlin
7.8%
Ruby
7.1%
2020
https://insights.stackoverflow.com/survey/2020



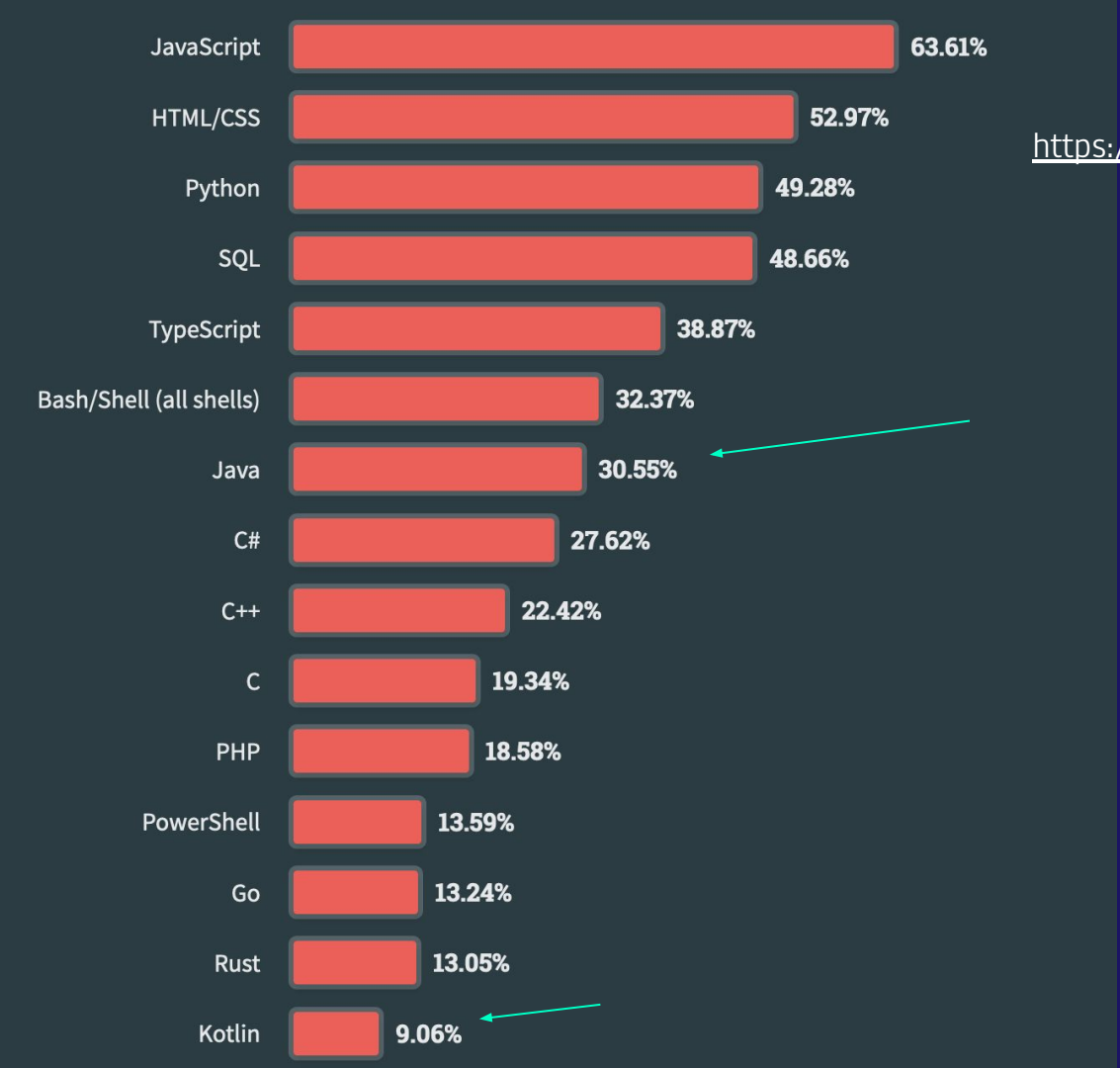
https://insights.stackoverflow.com/survey/2021

2021



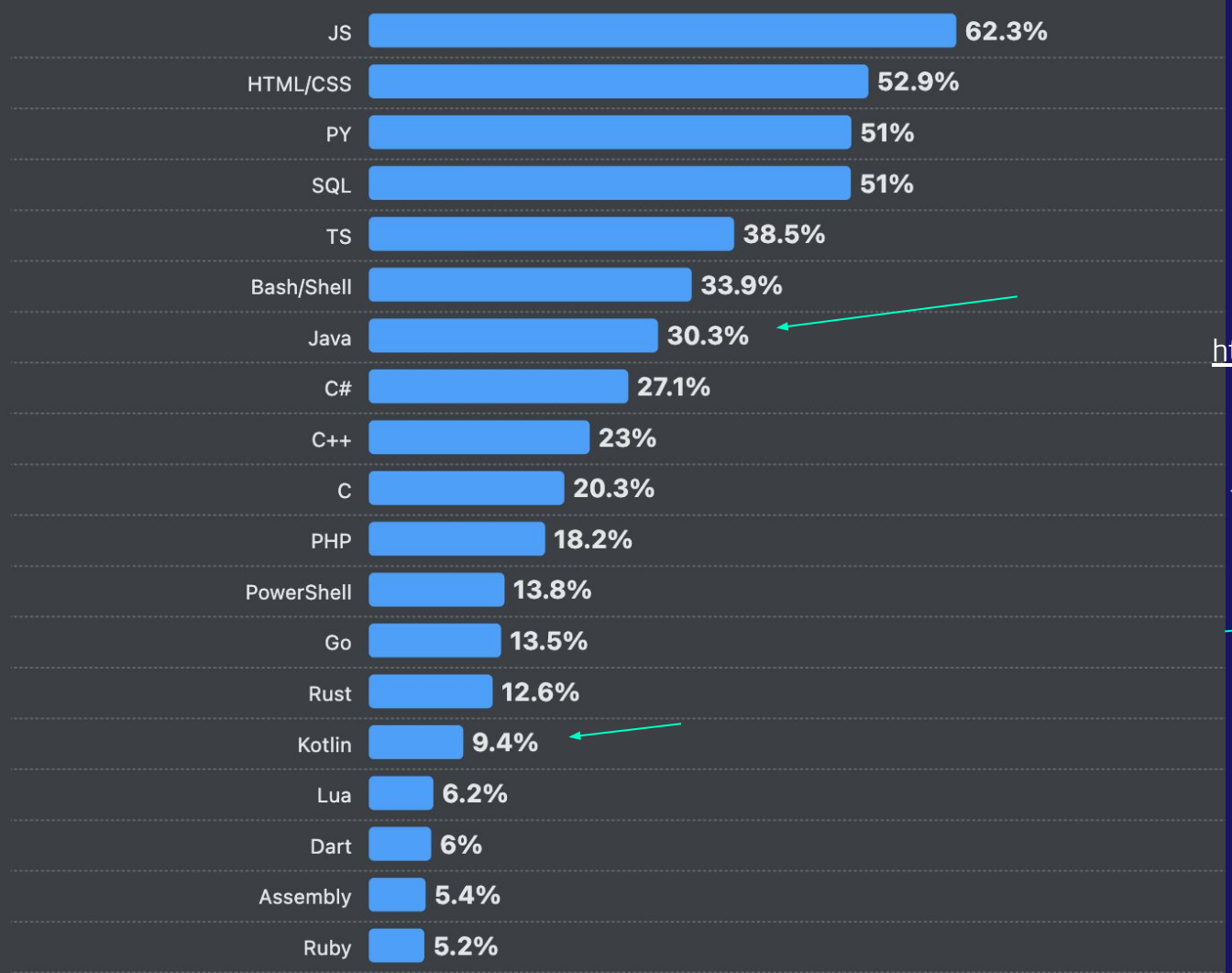
https://survey.stackoverflow.co/2022/

2022



https://survey.stackoverflow.co/2023/

2023

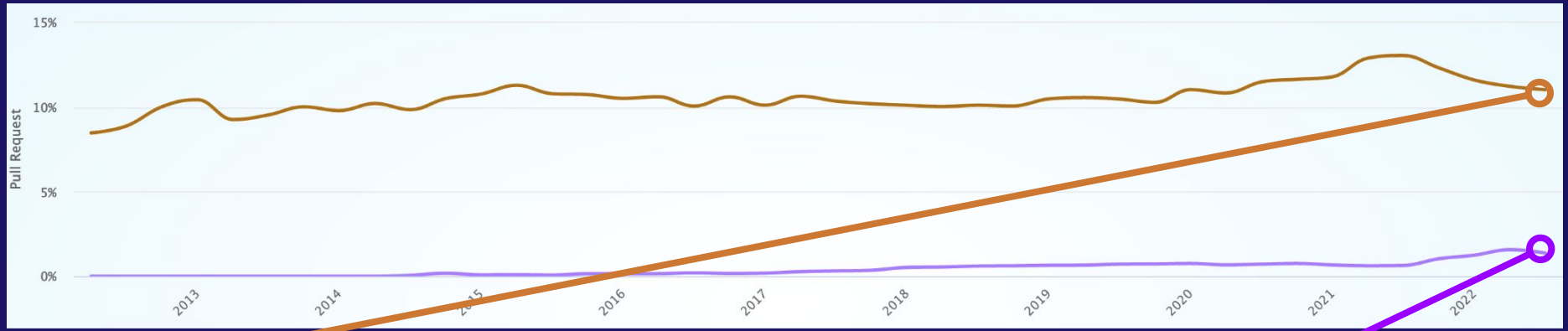


https://survey.stackoverflow.co/2024/

2024

Vad säger GitHut?

Githut utgår från antal Pull Requests



Antal 'Pull Requests'
2022

Out of all languages on projects at GitHub
https://madnight.github.io/githut/#/pull_requests/2022/3

Source:
https://madnight.github.io/githut/#/pull_requests/2024/1

+0.7%

Antal 'Pull Requests' 2024

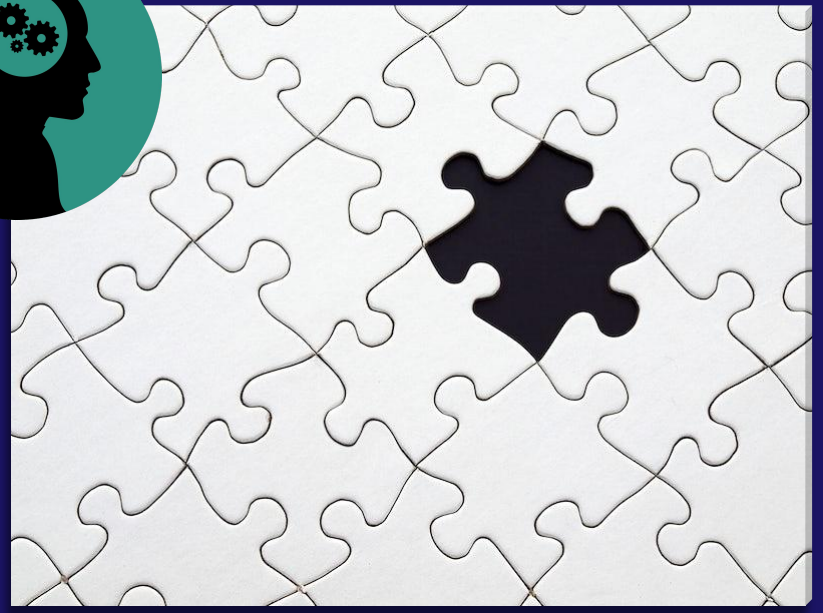
# Ranking	Programming Language	Percentage (YoY Change)	YoY Trend
1	Python	16.925% (-0.284%)	
2	Java	11.708% (+0.393%)	
3	Go	10.262% (-0.162%)	
4	JavaScript	9.859% (+0.306%)	^
5	C++	9.459% (-0.624%)	v
6	TypeScript	7.345% (-0.554%)	
7	PHP	5.665% (+0.357%)	
8	Ruby	4.706% (-0.307%)	
9	C	4.616% (+0.208%)	
10	C#	3.442% (+0.300%)	
11	Nix	2.733% (+0.606%)	^
12	Shell	2.350% (-0.180%)	v
13	Rust	1.872% (+0.144%)	
14	Scala	1.590% (+0.036%)	
15	Kotlin	1.480% (+0.042%)	
16	Swift	1.077% (+0.141%)	
17	Dart	0.713% (-0.211%)	

Java!

Java kommer inte att dö ut någon gång snart!

- **Stort gemenskap med över 10 Mil. utvecklare**
- **Utvecklas än idag**
- **Biljoner enheter använder sig fortfarande utav Java**
- **Java är fortfarande ett otroligt bra språk för byggnation av bl.a. ‘Backends’**

Frågor?

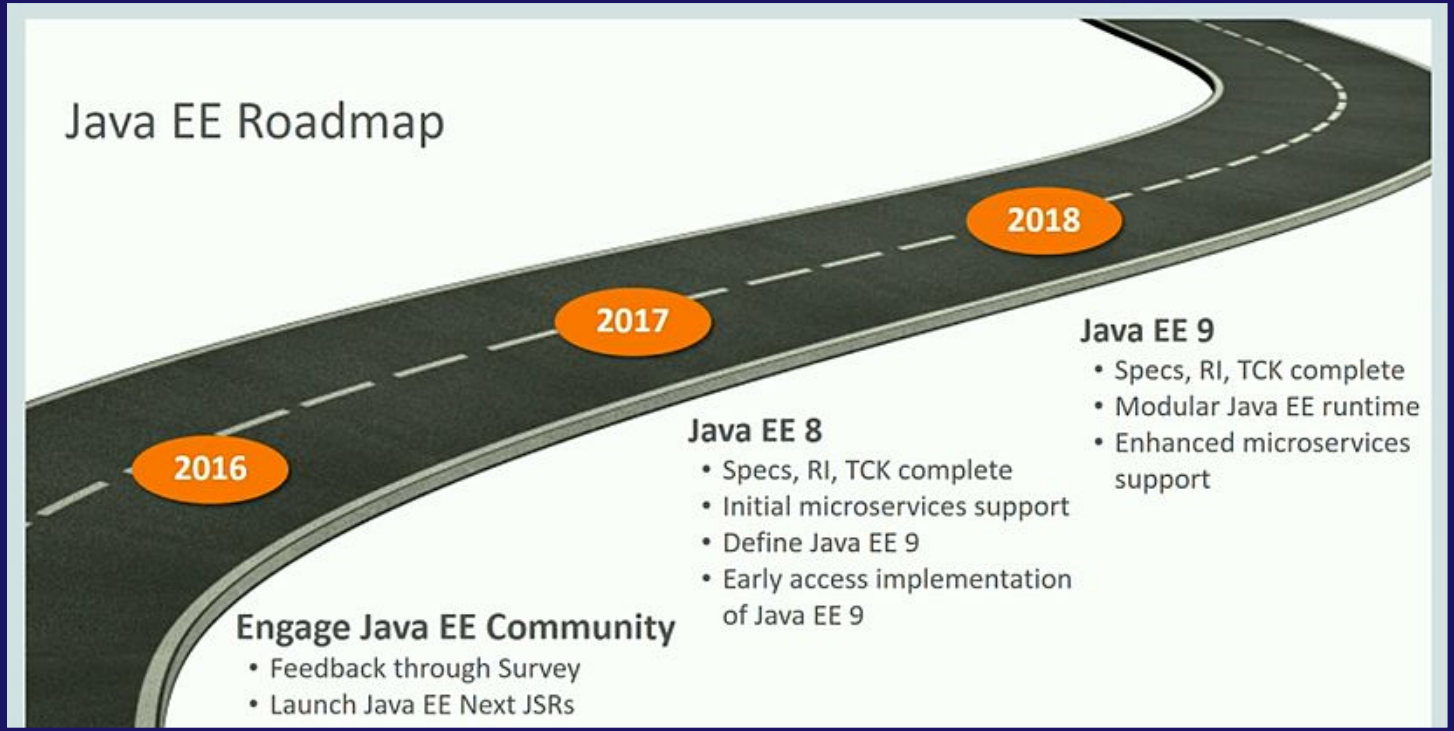


02

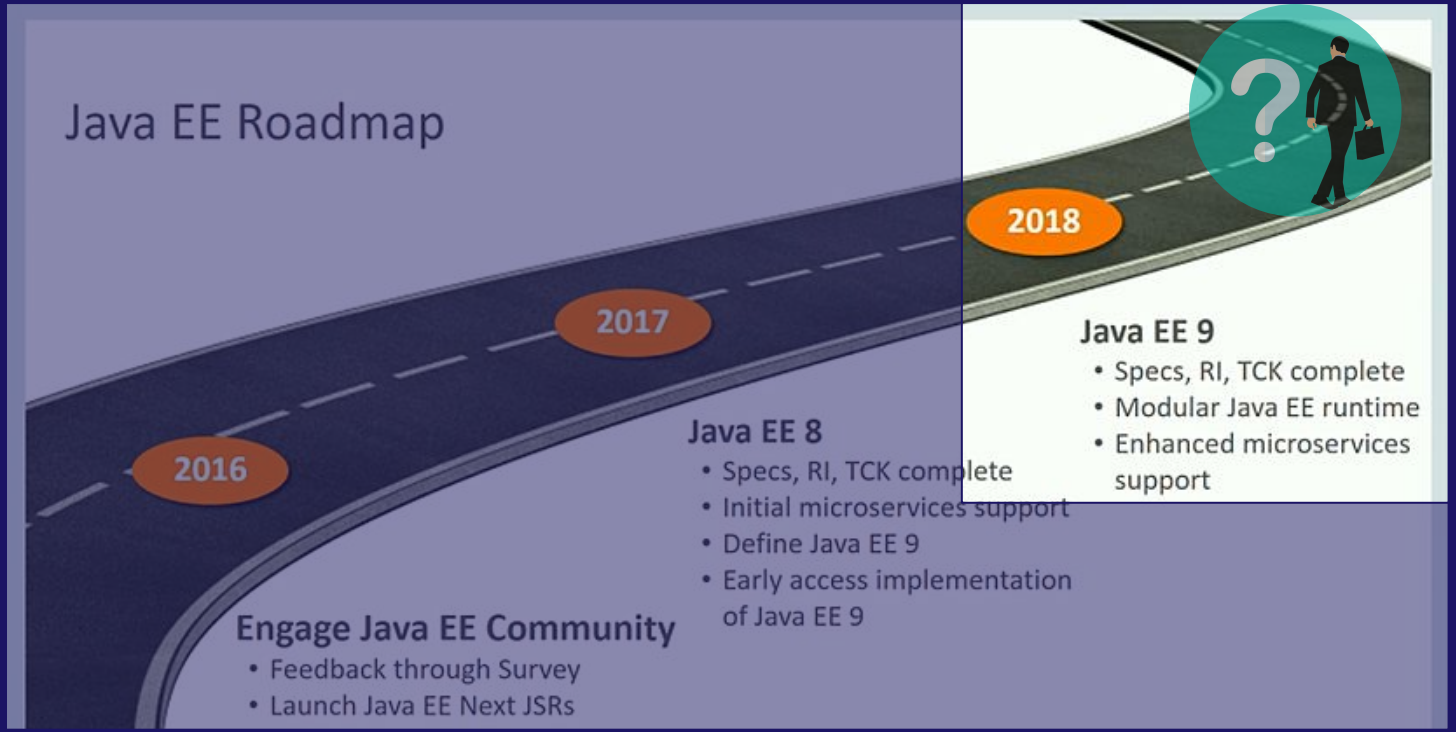
Java’s Version Historik


Oracle’s Roadmap

Oracle Roadmap



Oracle Roadmap



Oracle doesn't want to be the bottleneck for EE

Solution

Oracle open-sourced Java through OpenJDK.
Anyone can contribute and build their own JDK distribution.

The name *Java* is a trademark.. you can't use it in product/package names without Oracle's approval.

Long story short…
–Ever since–

Java has moved beyond its reputation as a slow and legacy-bound platform, and there is now a clear drive for innovation.


Enkel Översikt

Historik

J2EE 1.2	December **1999**
J2EE 1.3	September **2001**
J2EE 1.4	November **2003**
Java EE 5	Maj **2006**
Java EE 6	December **2009**
Java EE 7	April **2013**
Java EE 8	Augusti **2017**
Jakarta EE 8	September 10, **2019**
Jakarta EE 9	December 8, **2020**
Jakarta EE 10	September 22, **2022**

DEFINITIONER

ORACLE

Java

Java är ett språk som utvecklats genom åren och släppts ut via **Oracle**

JAKARTA EE

Jakarta

Jakarta är fortsättningen på Java med ett namnbyte.

Detta släpps av Eclipse Foundation.

Är även open source!


Java
SE, EE, ME, FX

ÖVERSIKT – MAIN FILEN

Java

Vilka typer finns?

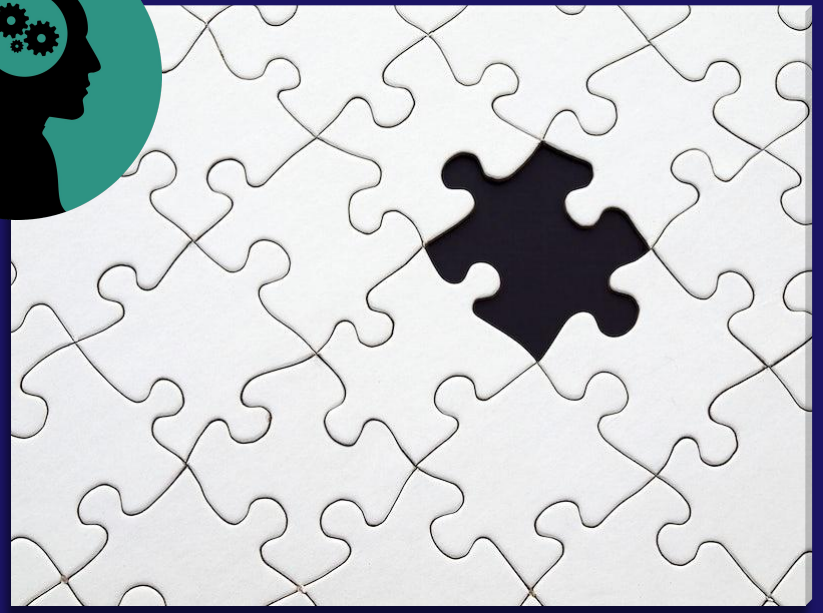
SE
Standard Edition

EE
Enterprise Edition
(Är uppbyggt uppe på SE)

ME
Micro Edition
(Inbäddade och Mobila enheter)

FX
Special Effects
Skapa ‘native’ applikationer

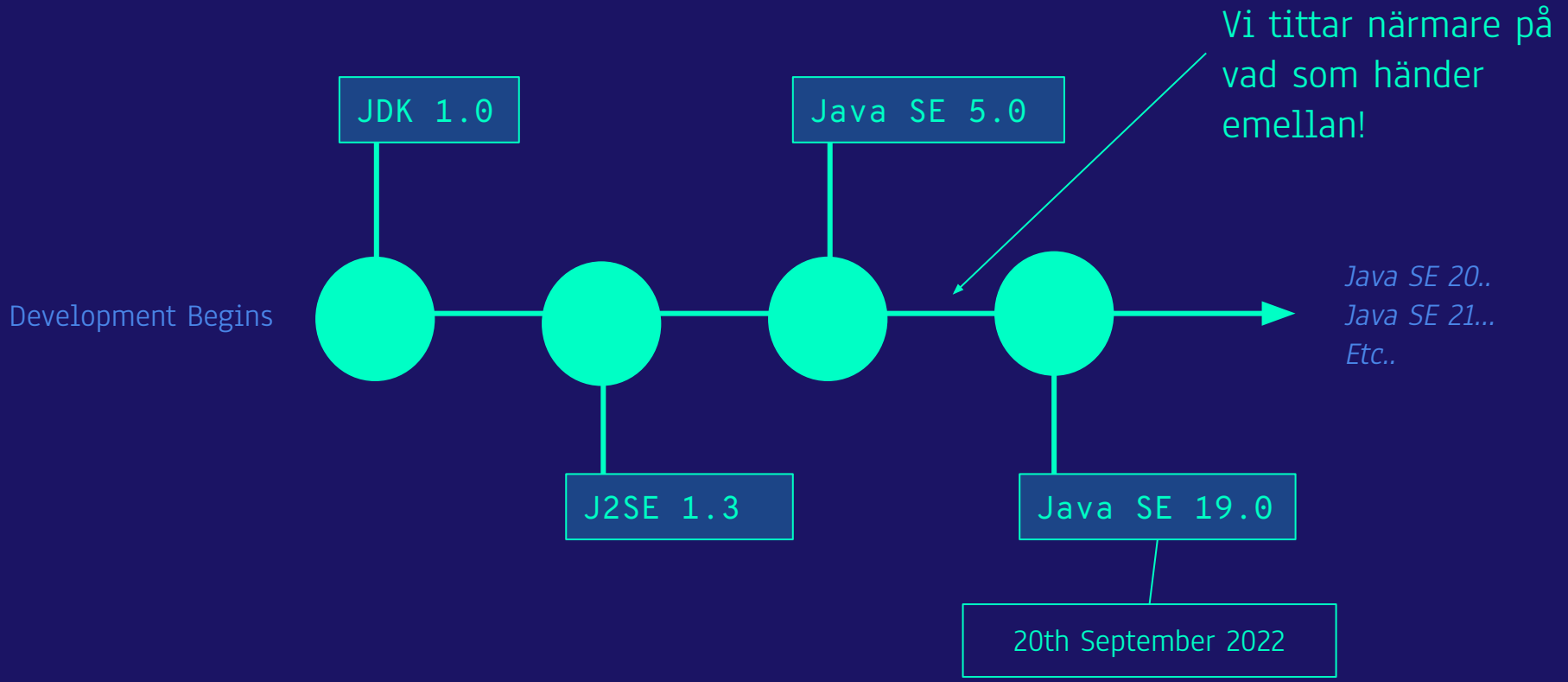
Frågor?

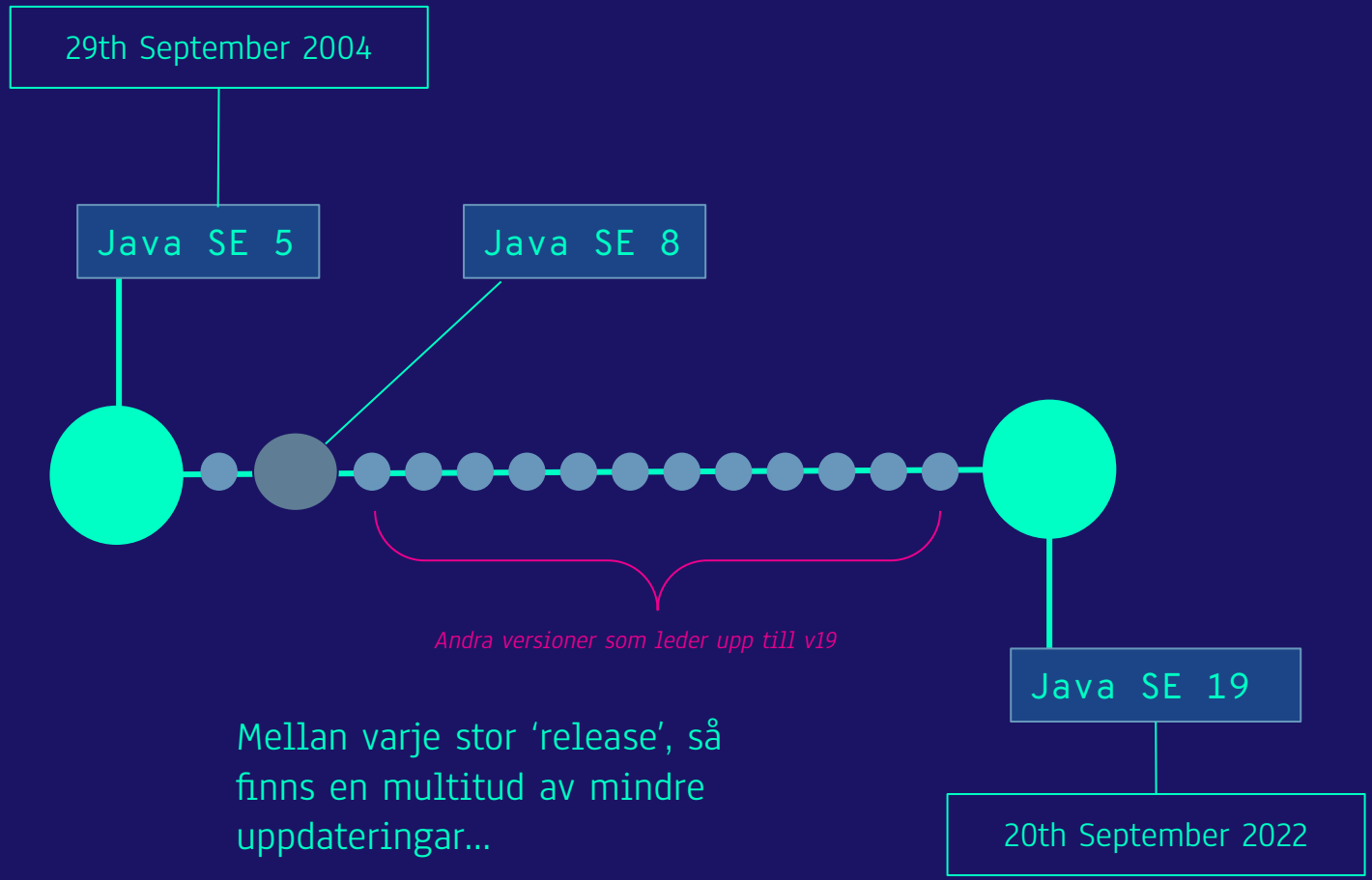


03

Spring 3.0.0


Big deal…?
Why?


Vi tittar närmare på vad som händer emellan!
JDK 1.0
Java SE 5.0
Development Begins
Java SE 20..
Java SE 21...
Etc..
J2SE 1.3
Java SE 19.0
20th September 2022



Mellan varje stor ‘release’, så finns en multitud av mindre uppdateringar…

Varför är **Spring 3.0.0** detta en ‘big deal’?

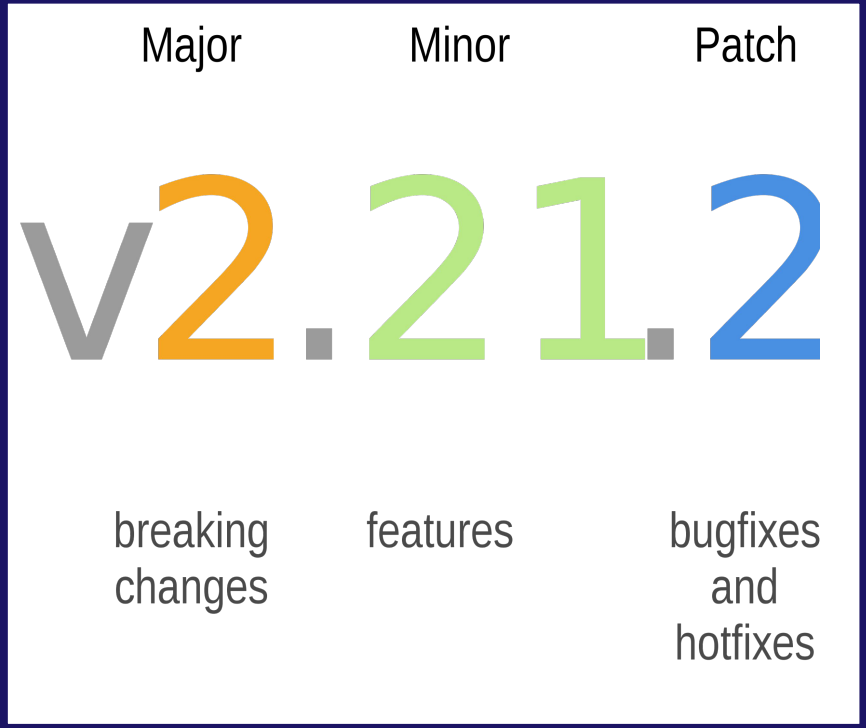
Java 17 Baseline and Java 19 Support

Spring Boot 3.0 requires Java 17 as a minimum version. If you are currently using Java 8 or Java 11, you’ll need to upgrade your JDK before you can develop Spring Boot 3.0 applications.

Spring Boot 3.0 also works well, and has been tested with JDK 19.

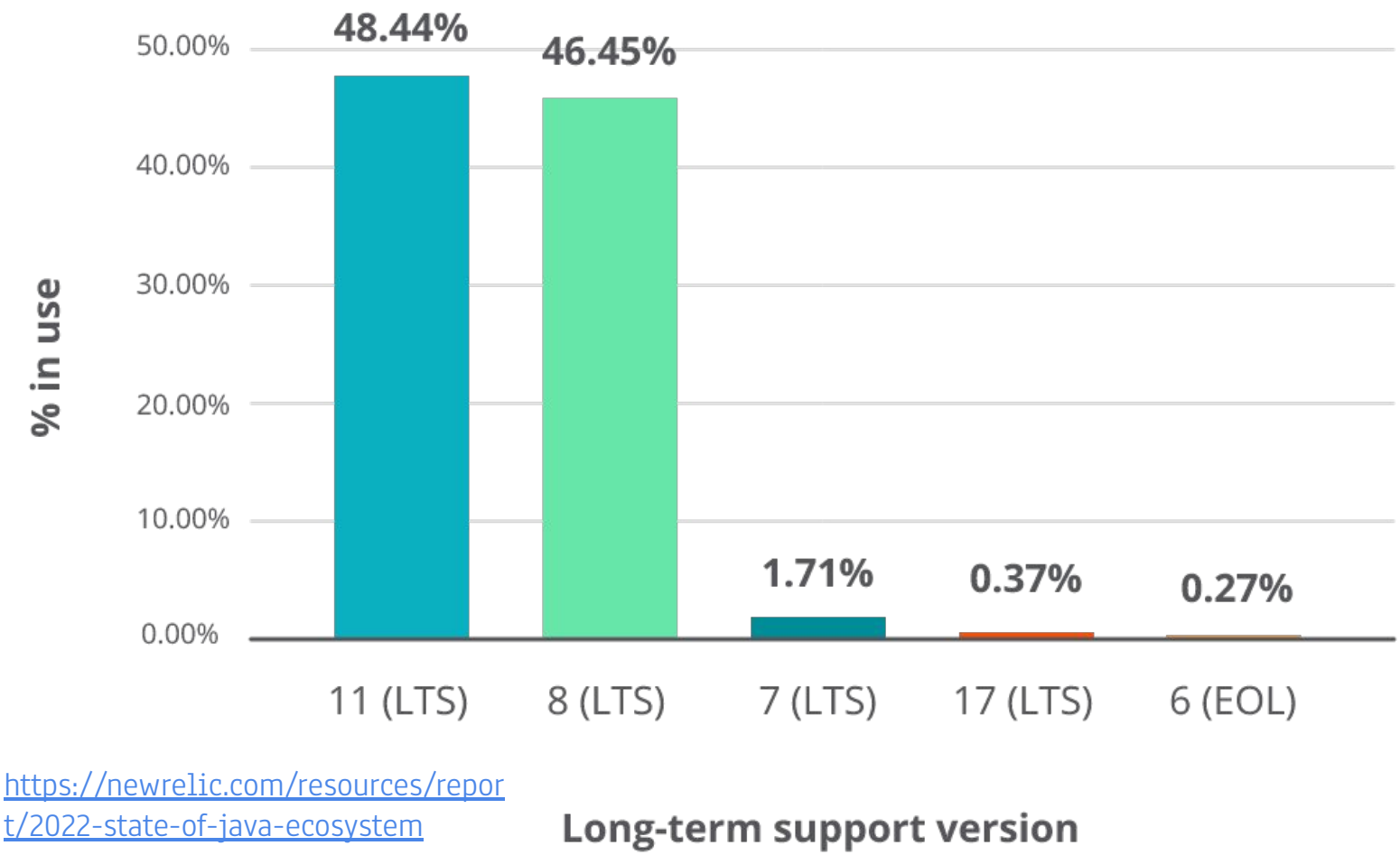
Tydligare struktur på versioner kan ni hitta på Wikipedia:

https://en.wikipedia.org/wiki/Java_version_history



https://www.jetbrains.com/lp/devecosystem-2021/java/

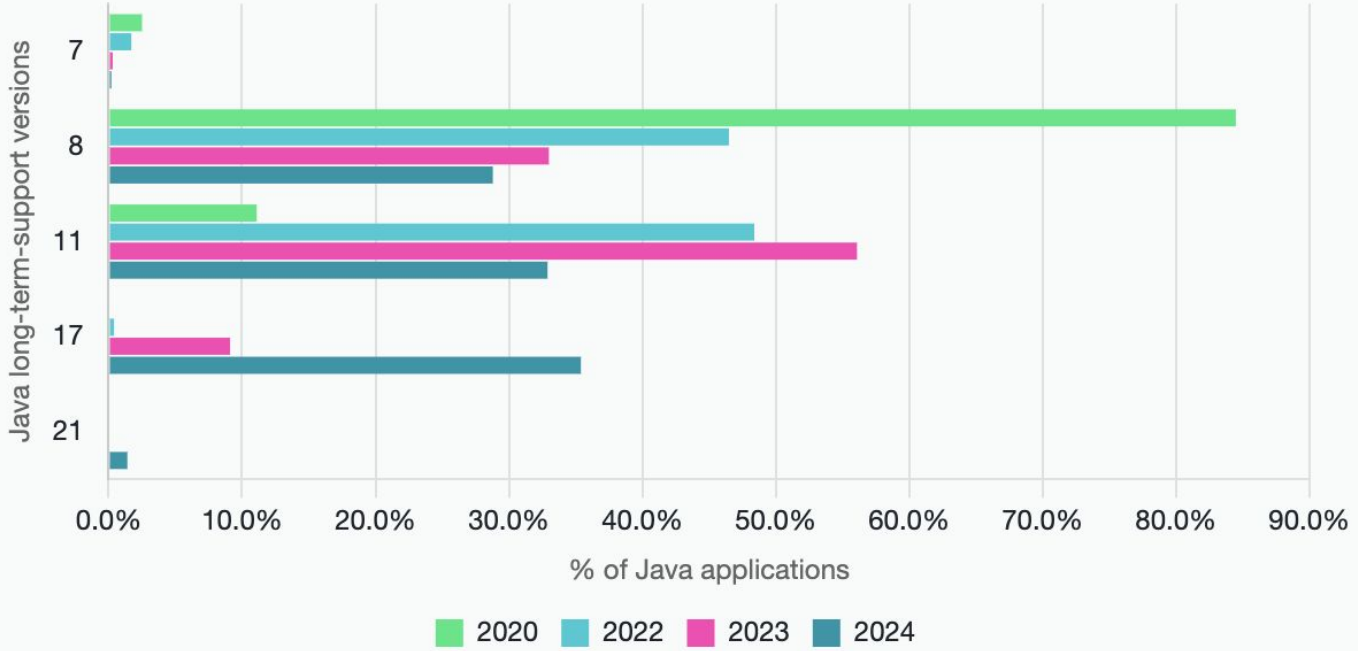
Deep dive into 2021!



https://newrelic.com/resources/report/2022-state-of-java-ecosystem

Source:
https://newrelic.com/resources/report/2024-state-of-the-java-ecosystem

Share



Java long-term-support version adoption by year

In addition, the adoption rate of Java 17 far exceeded what the developer world saw when Java 11 was introduced. About a tenth (9%) of applications were using Java 17 in production in 2023, and now 35% of applications are using Java 17, representing a nearly 300% growth rate in one year. It took years for Java 11 to reach anywhere near that level.

Source:
https://newrelic.com/resources/report/2024-state-of-the-java-ecosystem#new-java-versions-being-adopted-faster

DEFINITIONER

Java 8

Version 8

“ Java Version 8
En ofta förekommande version inom företag än idag.”

Java 17

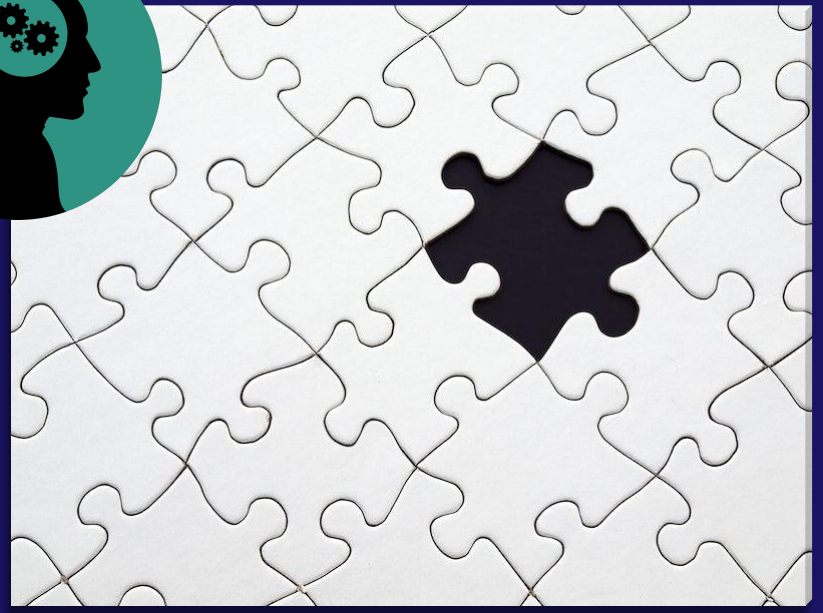
Version 17

“ En utav de nyare varianterna

Används i mindre utsträckning. Men tider förändras numera!

Vi tittar närmre”

Frågor?



Problem

Tidsfråga innan något
Som är ‘LTS’ dör ut.
(Long term support)

Solution

Underhålls fortfarande, numera
ej av Oracle.
Spring 3.0.0 adapterar java 17


Security Reasons

Work Lab

Säkerhet

Uppgift…
Navigera till
https://mvnrepository.com/artifact/org.springframework/spring-web

Märker ni några problem med säkerhet på äldre versioner?

Vad innebär detta för ‘hackers’?

Example

Source
https://www.cve.org/CVERecord?id=CVE-2025-41249

From Spring Web 6.2.7


Java 8
VS
Java 17

```java
if (obj instanceof MyObject myObject) {
  // … the same logic
}
```

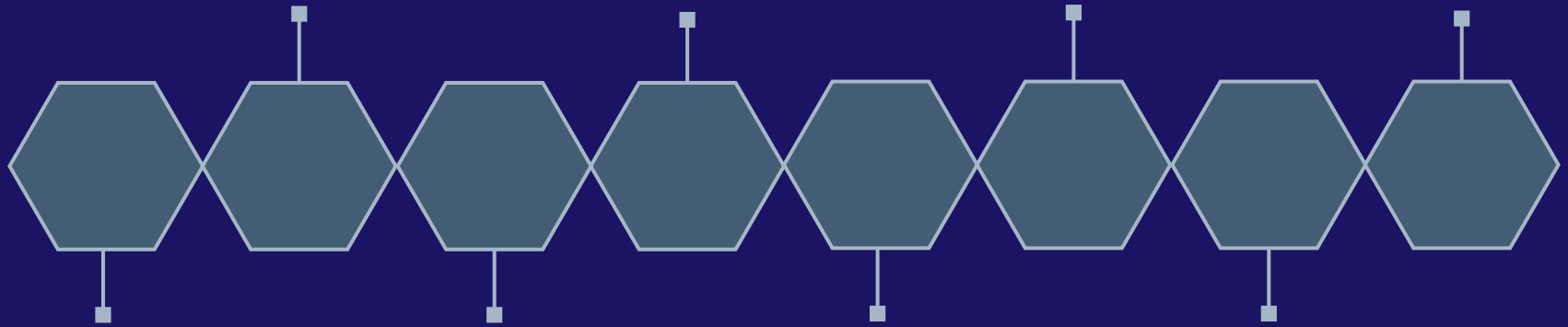
```java
public sealed class Animal permits Dog {

}
```

InstanceOf

Sealed Classes

Better Nullpointer Exceptions

More...



Records

Enhanced Switch

```java
switch (new Scanner(System.in).next()) {
    case "0" -> System.out.println("0");
    case "1" -> System.out.println("1");
    case "2" -> System.out.println("2");
}
```

TextBlocks

```
String myWallOfText = """

 _____        _   _
|  __ \      | | (_)
| |__) |__ _ | |_ _  _ __
|  ___/ _` || __| || '_ \
| |  | (_| || |_| || | | |
|_|   \__,_| \__|_||_| |_|

"""
```

Optional OrElseThrow


Checking out
Jakarta
(Theory)

Notera 3.0.0

Project

- (●) Gradle - Groovy
- ( ) Gradle - Kotlin
- ( ) Maven

Language

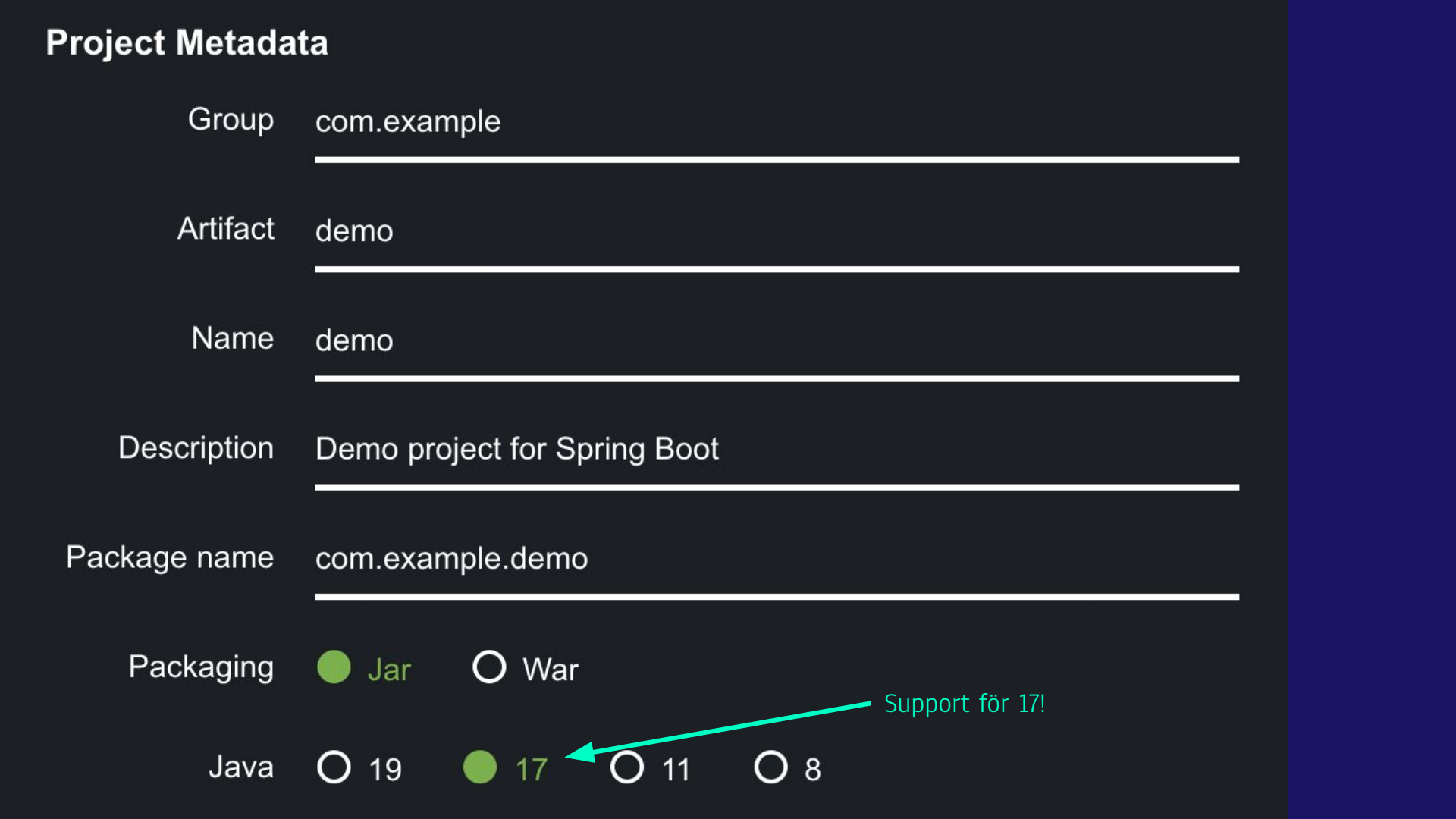
- (●) Java
- ( ) Kotlin
- ( ) Groovy

Spring Boot

- ( ) 3.0.1 (SNAPSHOT)
- (●) 3.0.0
- ( ) 2.7.7 (SNAPSHOT)
- ( ) 2.7.6

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	● Jar ○ War
Java	○ 19 ● 17 ○ 11 ○ 8



Dependencies

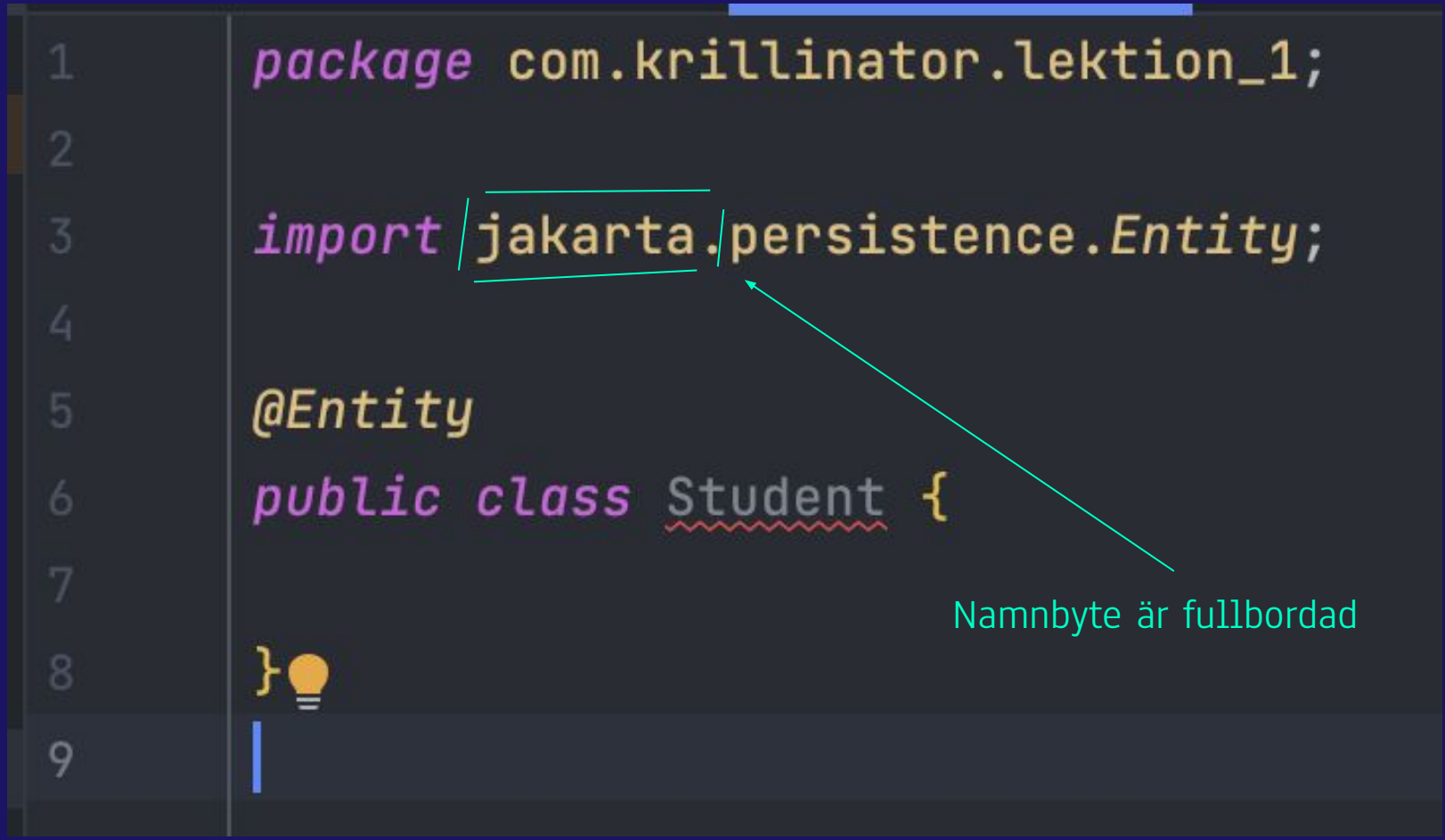
ADD DEPENDENCIES... ⌘ + B

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

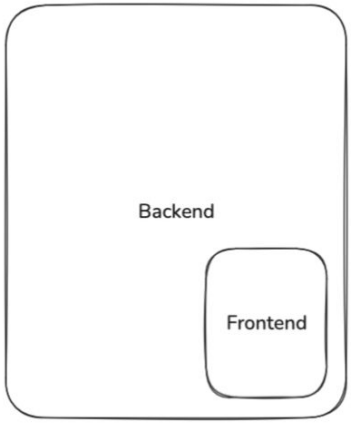
Spring Data JPA SQL

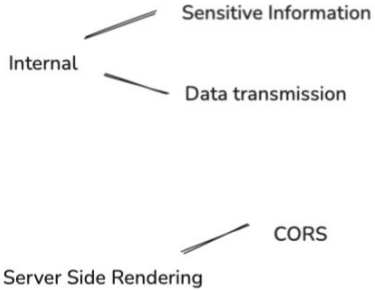
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

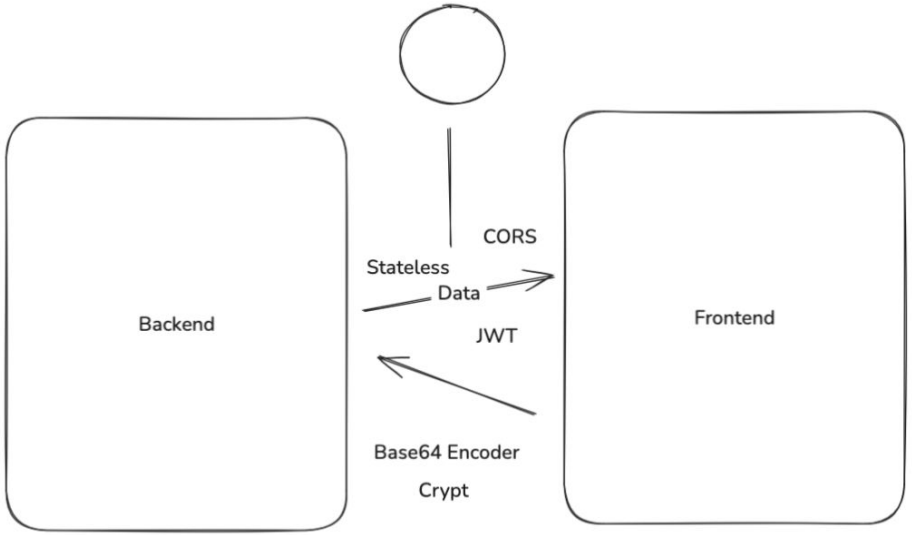

1 package com.krillinator.lektion_1;
2
3 import jakarta.persistence.Entity;
4
5 @Entity
6 public class Student {
7
8 }
9
Namnbyte är fullbordad

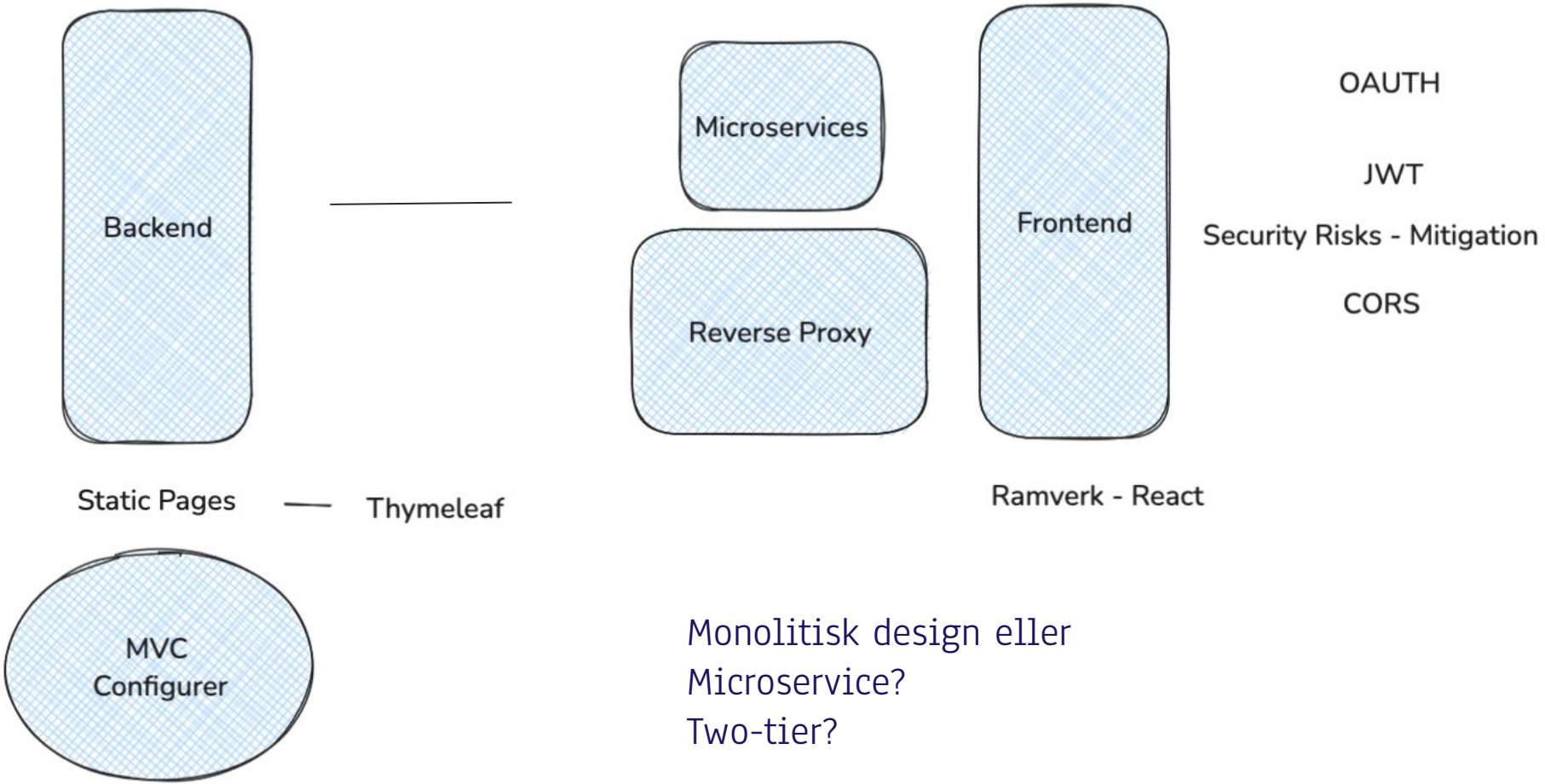

Enterprise
Overview
What do we build?

Vad är det för system vi bygger?









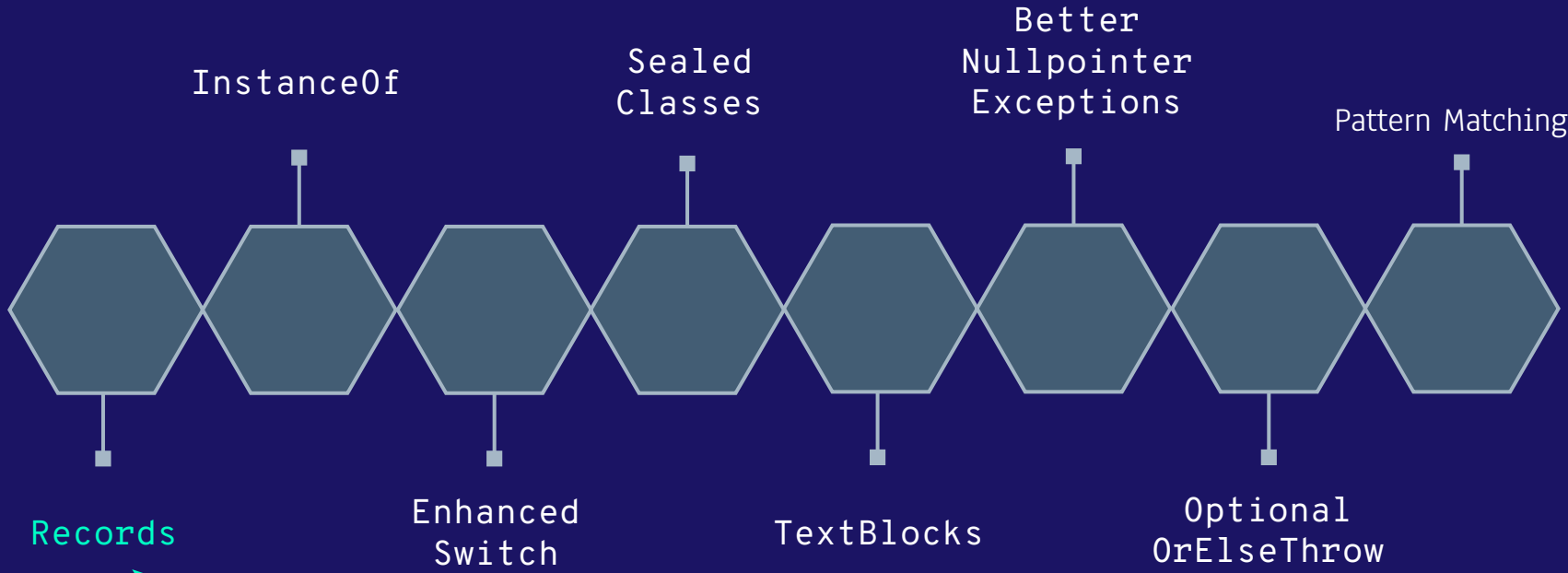
Monolitisk design eller
Microservice?
Two-tier?

04

Java 8+ Features


The many features
of Java 8-21


Record


InstanceOf
Sealed Classes
Better Nullpointer Exceptions
Pattern Matching
Records
Enhanced Switch
TextBlocks
Optional OrElseThrow

Oracle:

“Liksom en ‘enum’ är ‘record’ en begränsad form av en klass. Den är idealisk för "vanliga databärare", klasser som innehåller data som inte är avsedda att ändras och bara de mest grundläggande metoderna som konstruktörer och tillbehör.”

Source:

https://docs.oracle.com/en/java/javase/14/language/records.html

RECORD?!

```
public record Student(String name) {

    // Following is included in a Record:
    // Getters
    // ToString()
    // All args Constructor
    // Equals (hashmap,set,table)
    // Hashcode (hashmap,set,table)

}
```

Record?

Med ett ‘Record’ tillkommer ett par extra delar utan att vi konstant behöver syssla med alla dessa delar själva!

Records kännetecknas med ett ‘R’

Läs mer:
https://www.baeldung.com/java-record-keyword

Record

Record

Vad ingår?

Get

Inkapsling med ‘private’

toString()

Skickar ut data om objektet istället för adress

AllArgsConstructor

Under instansiering krävs inmatning av alla attribut

Equals

Jämför om ett objekt är samma som jämförelsen

Learn more about RECORDS

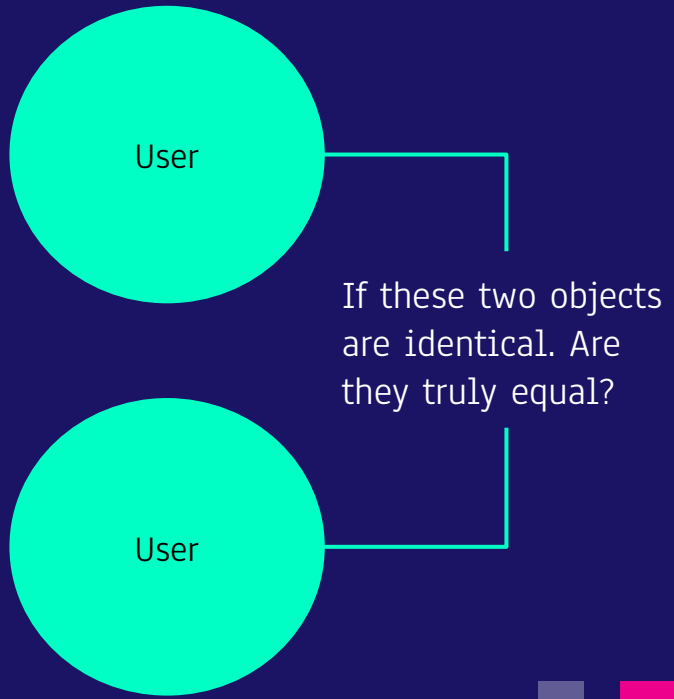
https://www.baeldung.com/java-record-keyword

Object & Equals

Object & Equals:

Varje objekt vi skapar
(Exempelvist: **Student)**

Har per automatik en ‘equals’ metod inbyggd.
Vi tittar närmre..



Equals()

```
Student student = new Student("Benny");
Student student2 = new Student("Benny");

System.out.println(student.equals(student2));
System.out.println(student2.equals(student));
```

False False

När vi skriver på detta vis så får vi tyvärr inte det förväntade resultat…

Här ser vi att objektet är unikt och att dessa ej sparar minnet på samma plats.

Så vi saknar ett välutformat sätt att faktiskt jämföra att objekt är ‘likadana’

Class

```java
package com.krillinator.demo.Enterprise;

import java.util.Objects;

public class Student2 {

    private final String name;

    public Student2(String name) {
        this.name = name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Student2 student2 = (Student2) o;
        return Objects.equals(name, student2.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name);
    }

    public String getName() {
        return name;
    }

}
```

Same result

I detta exempel gör vi precis samma sak som en ‘record’ fast i en vanlig klass.

Vi har alltså skapat en helt ny klass här!

Detta är hur vi hade behövt skriva om vi inte haft tillgång till ‘records’

Märk av att vi override:at!
(Se nästa slide)

Equals()

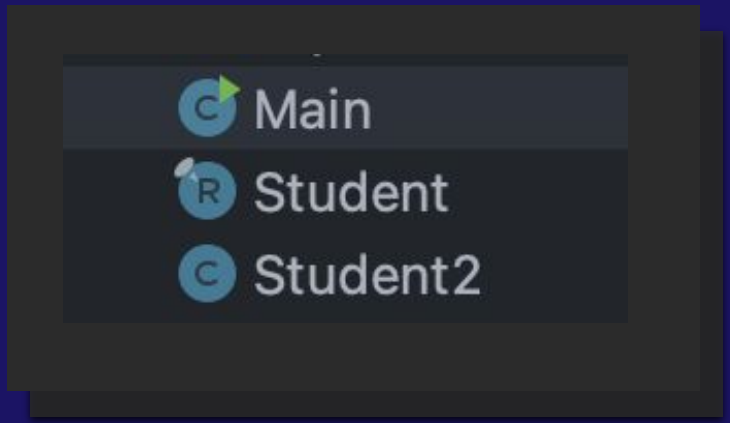
```java
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass())
return false;
    Student2 student2 = (Student2) o;
    return Objects.equals(name, student2.name);
}
```

Jämförelse

Är **Objekt_1,**
samma som **Objekt_2**?

Märk av att detta är en Override!
(Super viktigt)

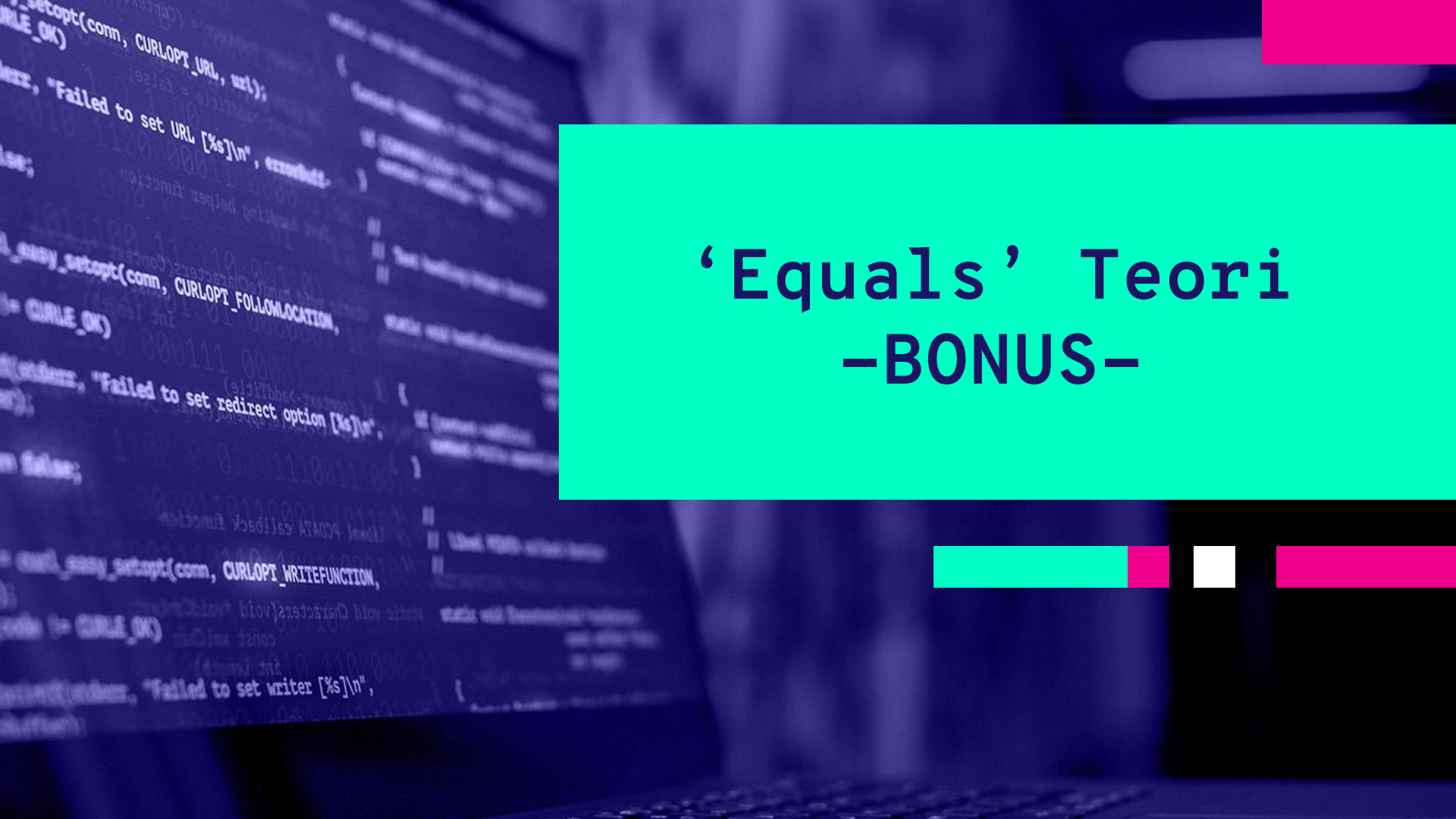
Setup



Låt oss testa teorin och användning!

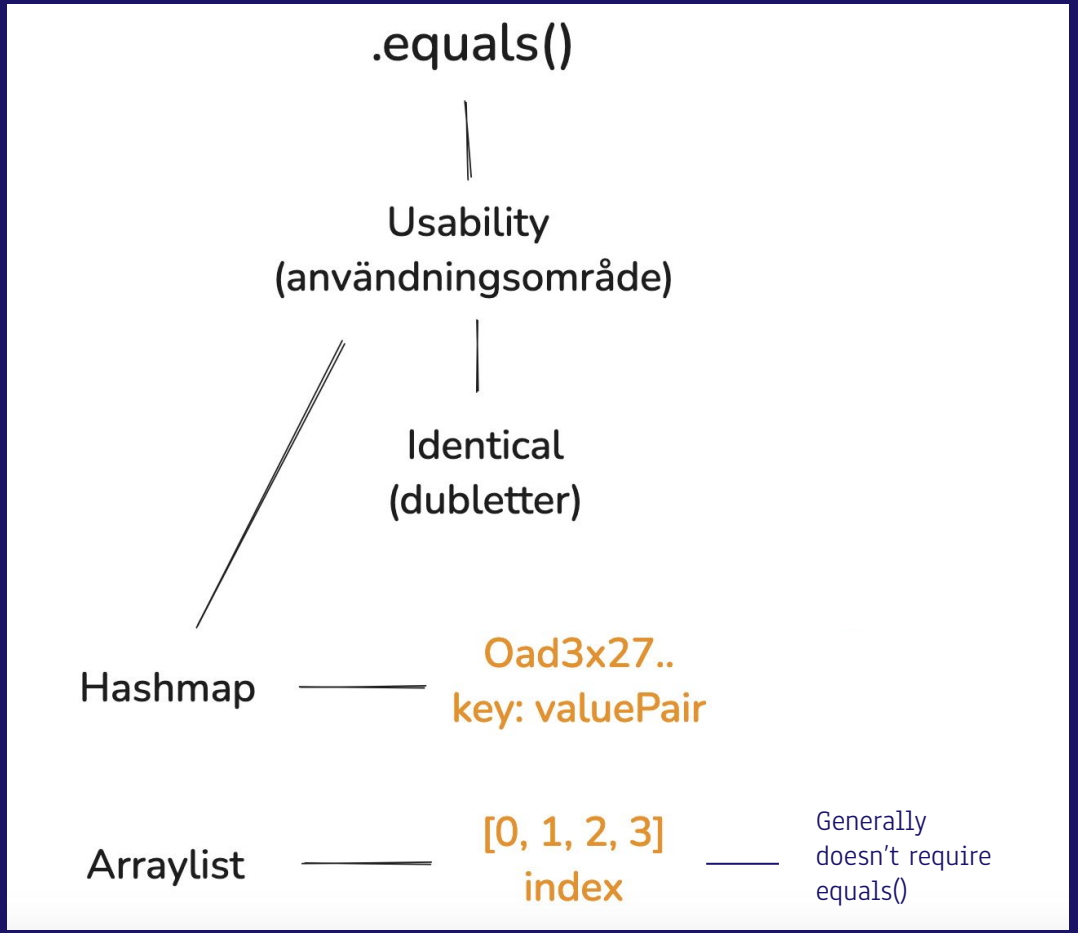
Här har vi vår struktur:

- Student (record)
- Student2 (klass)
- Main (klass)



‘Equals’ Teori
-BONUS-

Varför Equals?
Hantera Dubletter
Eller identiska
objekt



Object & Equals
TEORI

Viktigt att ha översende att detta enbart är för att se ett exempel på hur ‘equals’ fungerar.

Helikopter Perspektiv rekommenderas

https://www.infoworld.com/article/2256967/comparing-java-objects-with-equals-and-hashcode.html

Step #1

Steg ‘UNO’

Här skapar vi två studenter från två olika klasser.

De har samma namn men är ultimat unika.

```
public class Main {
    public static void main(String[] args) {

        Student student = new Student("Benny");
        Student student2 = new Student("Benny");


System.out.println(student2.equals(student));



    }
}
```

// printar “False”

Step #2

Samma klass

Prova nu att skapa en @override för Equals metoden inom STUDENT klasserna.

Vad kommer hända nu...?

```
package com.krillinator.demo.Enterprise ;

public class Main {
    public static void main(String[] args) {

        // Student student = new Student("Benny");
        Student student = new Student("Benny");
        Student student2 = new Student("Benny");

        System.out.println(student.equals(student));
    }
}
```

// printar: “true”

Kontrakt

Equals Contract

Reflexive	An object must equal itself
Symmetric	***x.equals(y)* must return the same result as *y.equals(x)***
Transitive	if *x.equals(y)* and *y.equals(z)*, then also *x.equals(z)*
Consistent	The value of *equals()* should change only if a property that is contained in *equals()* changes (no randomness allowed)

Source:
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Object.html

Kontrakt

Reflexive

Reflexive	An object must equal itself

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
}
```

Kontrakt

Symmetric

Symmetric	***x.equals(y)* must return the same result as *y.equals(x)***

```
Student student = new Student("Benny");
Student student2 = new Student("Benny");

System.out.println(student2.equals(student));
System.out.println(student.equals(student2));
```

Kontrakt

Transitive

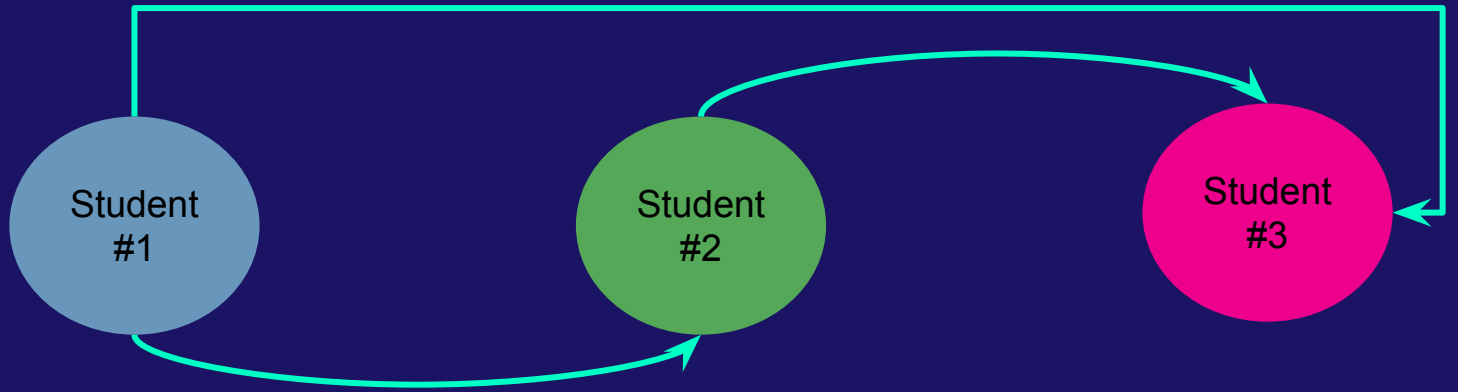
Transitive	if *x.equals(y)* and *y.equals(z),* then also *x.equals(z)*

```
Student student = new Student("Benny");
Student student2 = new Student("Benny");
Student student3 = new Student("Benny");

System.out.println(student.equals(student2));
System.out.println(student2.equals(student3));

System.out.println(student.equals(student3));
```

Kontrakt

Transitive

Transitive	if *x.equals(y)* and *y.equals(z),* then also *x.equals(z)*



Om Student #1 == #2 som är == #3
Då är Student #1 också == #3

Kontrakt

Consistent

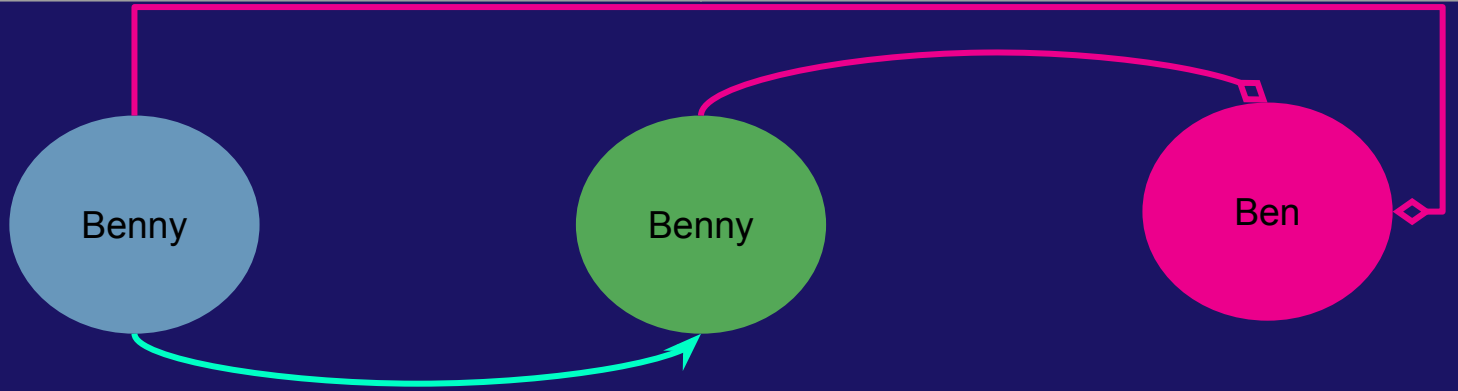
Consistent	The value of *equals()* should change only if a property that is contained in *equals()* changes (no randomness allowed)

```
Student student = new Student("Benny");
Student student2 = new Student("Benny");
Student student3 = new Student("e");

System.out.println(student.equals(student2));    // True
System.out.println(student2.equals(student3));   // False
System.out.println(student.equals(student3));    // False
```

Kontrakt

Consistent

Consistent	The value of *equals()* should change only if a property that is contained in *equals()* changes (no randomness allowed)



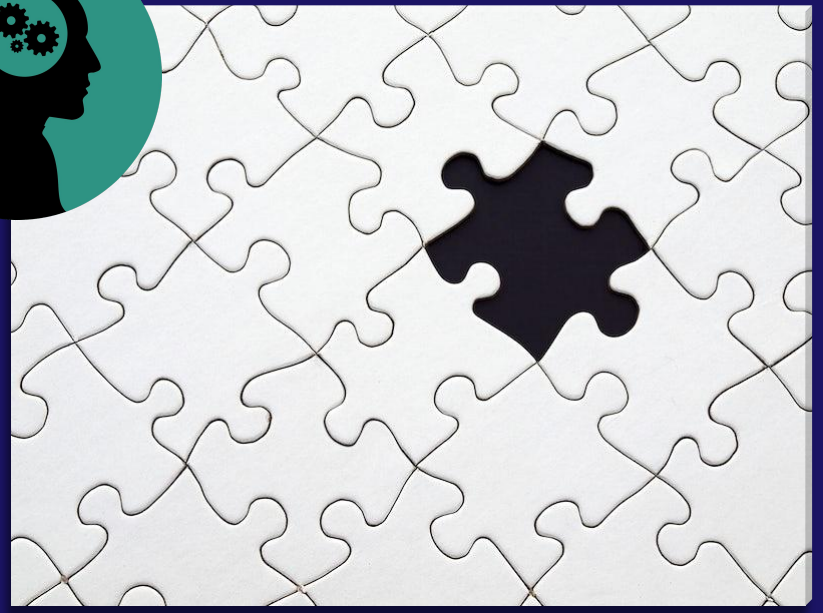
(student 1) Benny är samma som Benny
(student 2) Benny är ej samma som Ben.
(student 3) Därför är ej student 1 benny == ben

Domändriven design

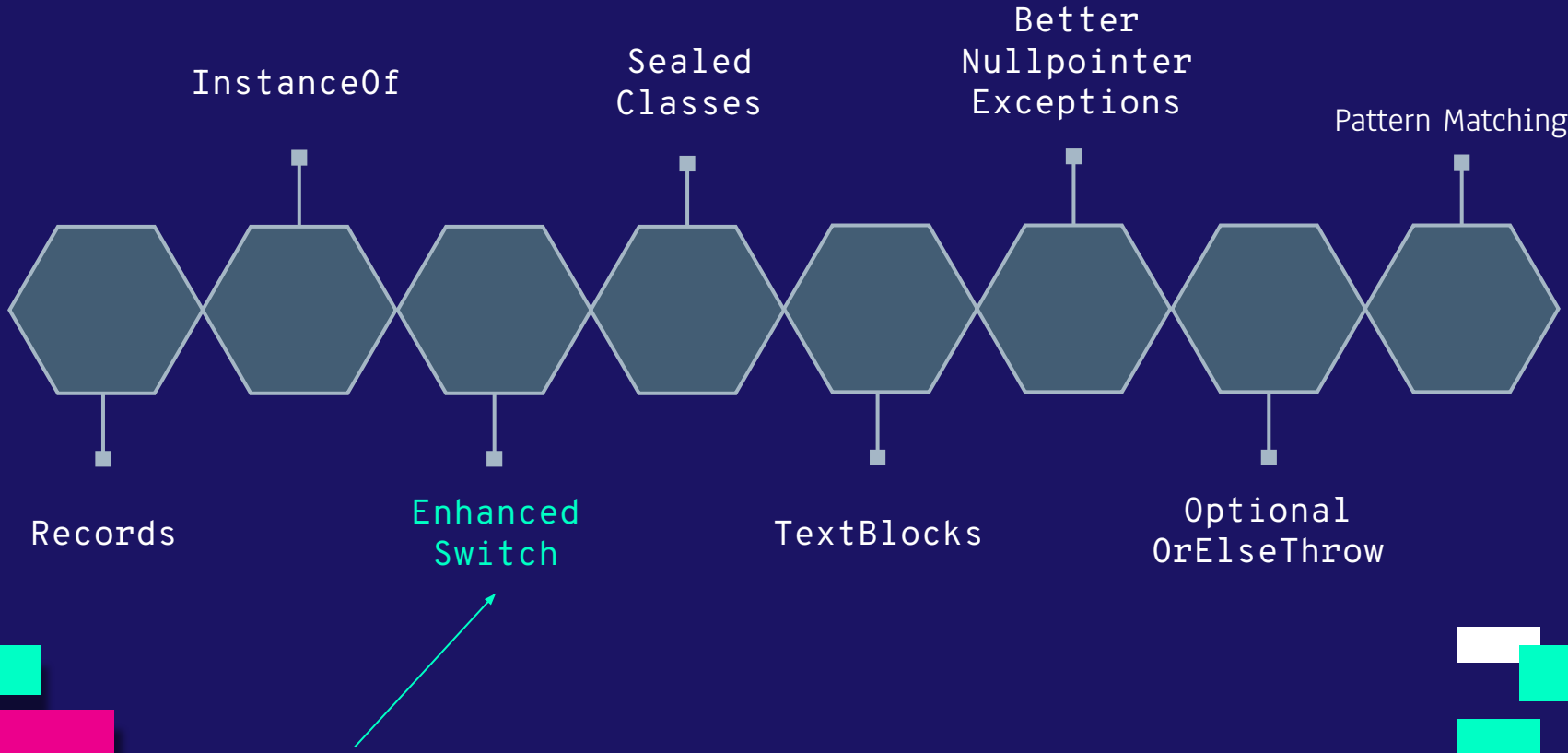
Vi behöver oftast ej tänka på detta då vi arbetar med ‘Entitets objekt’ för domändriven design.

Nu vet vi dock att den finns inbakad!

Frågor?




Enhanced Switch


InstanceOf
Sealed Classes
Better Nullpointer Exceptions
Pattern Matching
Records
Enhanced Switch
TextBlocks
Optional OrElseThrow

Easy Setup

Börja såhär

Märk av att vi har nyckelord

```
int dayOfTheWeek = 0;

switch (dayOfTheWeek) {
    case 0:
    case 1:
    case 2:
    case 3:
    case 4:
}
```

The bad

```
int dayOfTheWeek = 0;

switch (dayOfTheWeek) {
    case 0:
        System.out.println("Monday");
    case 1:
        System.out.println("Tuesday");
    case 2:
        System.out.println("Wednes");
}
```

Gör INTE såhär

När vi skapar en switch utan ‘breaks’ körs alla ‘cases’ under den ‘case’ som kallats.

Exempelvist…

DayOfTheWeek = 1

Case 0 - körs inte

Case 1 - körs

Case 2 - körs

Better!

Gör såhär

Nyckelordet ‘break’ avslutar ett ‘case’ helt.
Därför har vi ‘break’ under ‘SOUT’.
‘SOUT’ körs inte annars.

```
switch (dayOfTheWeek) {
    case 0:
        System.out.println("Monday");
        break;
    case 1:
        System.out.println("Tuesday");
        break;
    case 2:
        System.out.println("Tuesday");
        break;
}
```

Standard

```
switch (dayOfTheWeek) {
    default:
        System.out.println("Weekend!");
        break;
}
```

Default

Med default säger vi:

“Om fel värde matas in - hoppa direkt inuti ‘default’”

Hack

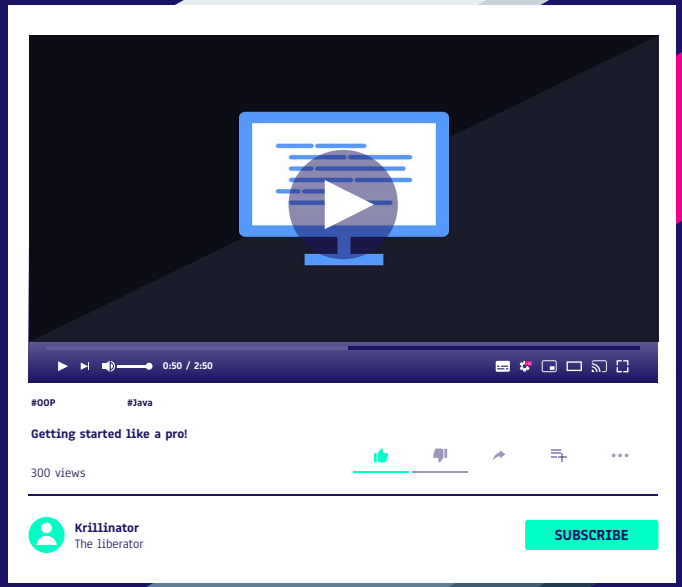
Hack utan break

Här tar vi faktiskt vara på ‘case’ utan ‘breaks’.

Då kommer den alltid rulla till 4 om vi matar in 0 –> 3

```
switch (dayOfTheWeek) {
    case 0:
    case 1:
    case 2:
    case 3:
    case 4:
        System.out.println("Weekday");
        break;
    default:
        System.out.println("Weekend!");
        break;
}
```

Switch - demo



Switch
Switch Body + param
Cases
Break;
Default;
Hack;

Problem

If elseif elseif elseif else….

Solution

Switch!

Vi underlättar debugging, hantering och underhåll med en ‘switch’!

Frågor?


Lambda

Lambda

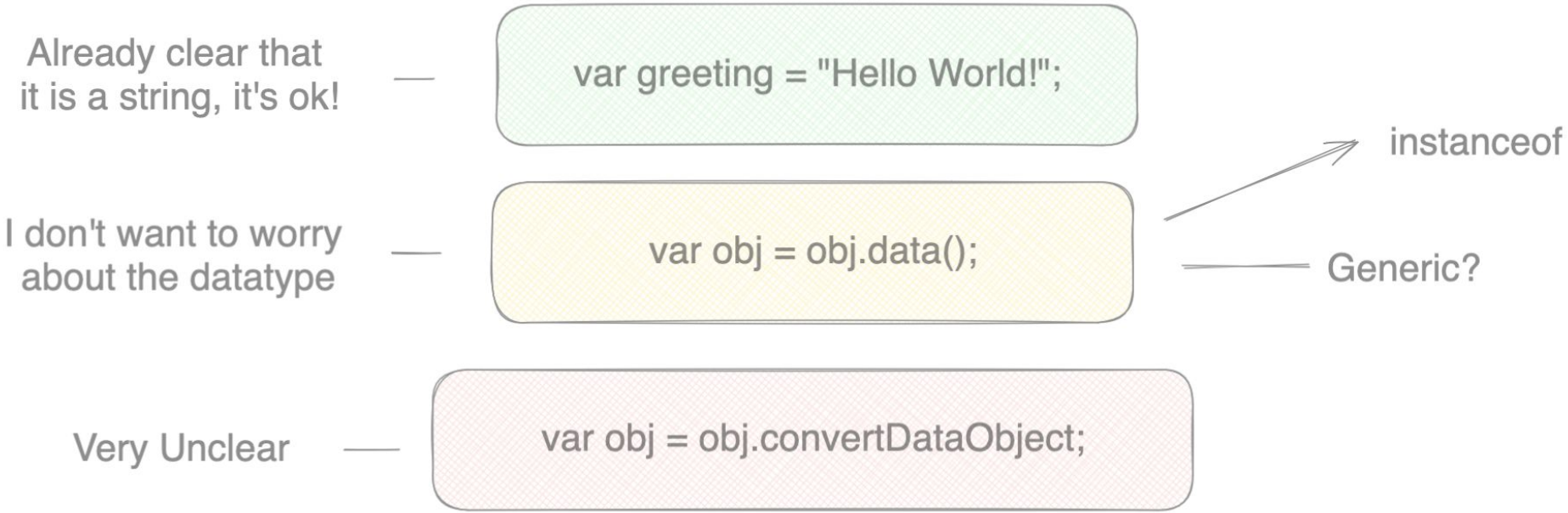
Likt metoder fast utan namn!

(argument-list) -> {body}

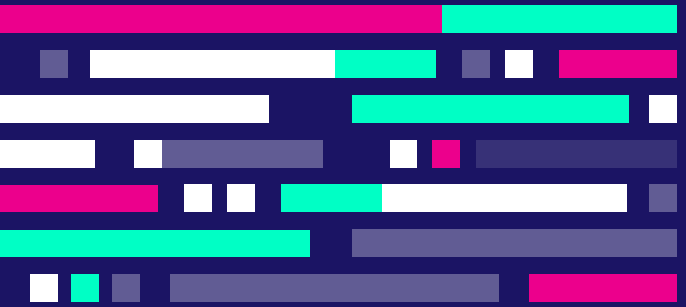
(index) -> { System.out.println(“Hello!”) }

https://www.w3schools.com/java/java_lambda.asp


Var keyword


Already clear that it is a string, it's ok!
var greeting = "Hello World!";
I don't want to worry about the datatype
var obj = obj.data();
instanceof
Generic?
Very Unclear
var obj = obj.convertDataObject;


Switch – Java 13
JDK 13



Java 13 - Switch

Med Java 13 kom ett nytt sätt att skriva switch satser!

Liknande lambda uttryck, så får vi ut mer av vår kod med mindre syntax!

https://openjdk.org/jeps/354

Lambda

Switch Lambda

```
switch (weekday) {
    case 0 -> System.out.println("0");
    case 1 -> System.out.println("1");
    case 2 -> System.out.println("2");
    default ->
System.out.println("Default");
};
```

Märk av att vi inte längre behöver ‘break’, mindre syntax och enklare att läsa!

Return

Return

Du kan även returnera ett resultat genom att definiera switch som en variabel!

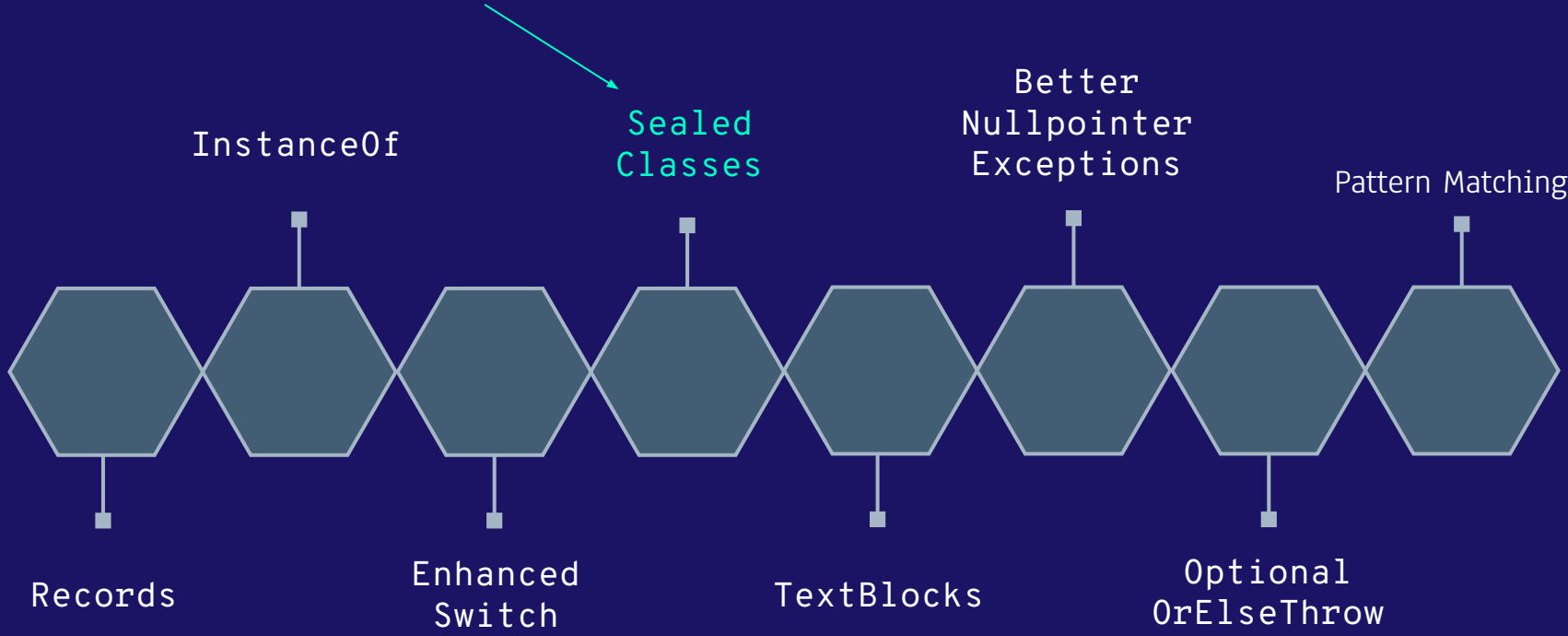
```
public void setupSwitchLambda() {

    var result = switch (weekday) {
        case 0 -> 0;
        case 1 -> 1;
        case 2 -> 2;
    };

    System.out.println(result);

}
```

Frågor?

05

Sealed Classes, Text Blocks & Pattern Matching

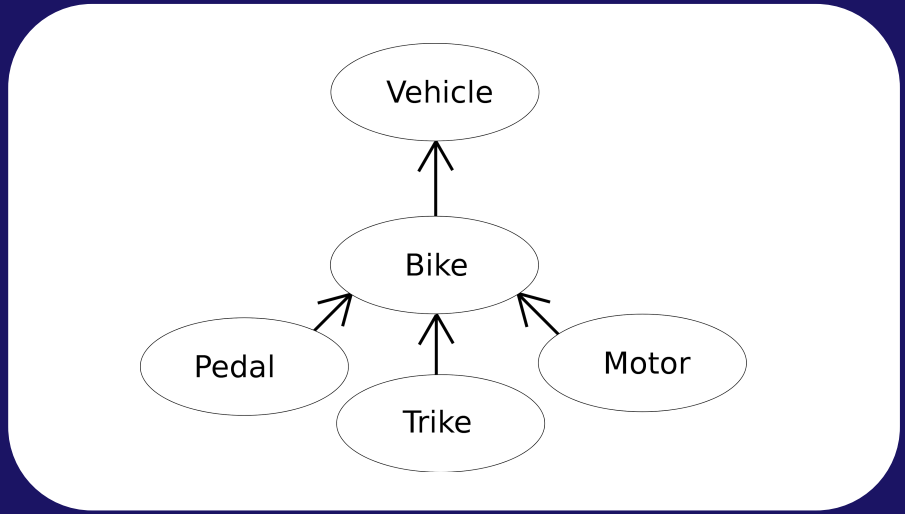

Sealed Classes


InstanceOf
Sealed Classes
Better Nullpointer Exceptions
Pattern Matching
Records
Enhanced Switch
TextBlocks
Optional OrElseThrow

Hierarki

En klasshierarki gör det möjligt för oss att återanvända kod via arv.

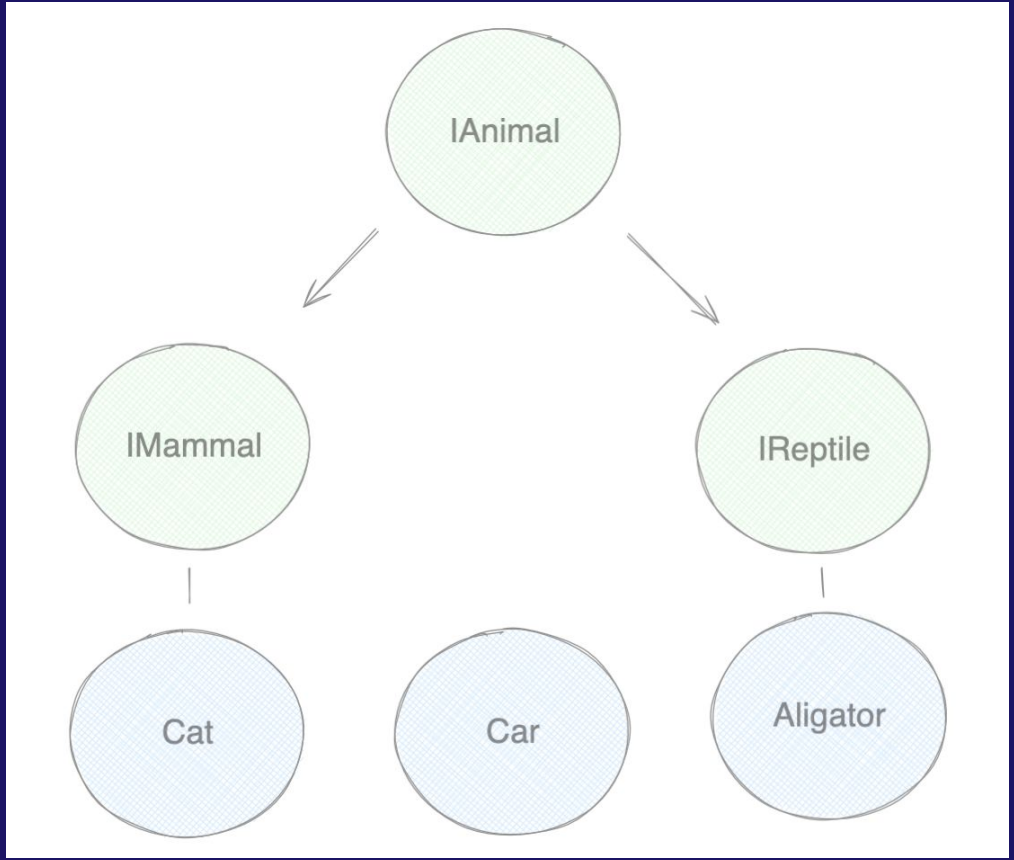
Klasshierarkier kan dock också ha andra syften.
Återanvändning av kod är bra men är inte alltid vårt primära mål.



Problematik

Car kan bli **Animal**, **Mammal** & **Reptile**

Om vi inte begränsar...



Deep dive

https://www.baeldung.com/java-sealed-classes-interfaces

Nyckelord

Keywords

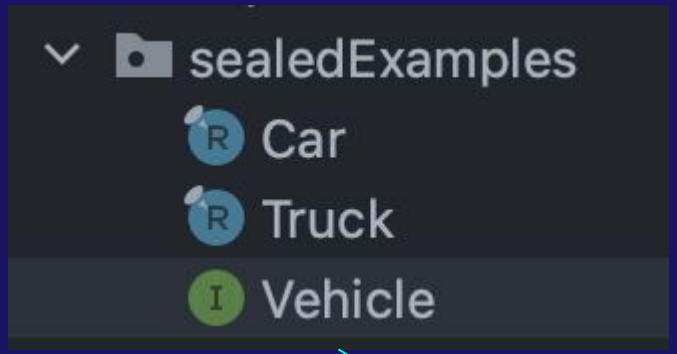
Sealed	Begränsar arv till nämnda klasser
Non-sealed	Vem som helst får ärva denna klass
Permits	Tillåter sagda klasser att få lov att ärva

Det går att lägga till nyckelordet på både **klasser** & **interface**

Setup

Sealed - fungerar utmärkt med records!

Låt oss testa detta med ett enkelt exempel



Problem?

Efter att vi tillämpat nyckelordet ‘sealed’ och ‘permits’

```
3  public sealed interface Vehicle permits Car, Truck {
4  }
5
6
7
```

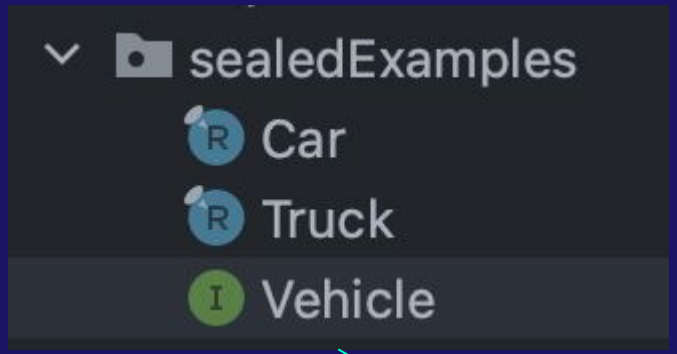
Invalid permits clause: 'Truck' must directly implement 'Vehicle'

Implement 'com.krillinator.demo.Enterprise.sealedExamples.Vehicle' ⌥⇧⏎ More actions... ⌥⏎

com.krillinator.demo.Enterprise.sealedExamples

```
public record Truck
extends Record
```

Testing



Problem?

Efter att vi tillämpat nyckelordet ‘sealed’ och ‘permits’ där vi tillåter Två klasser

- Car
- Truck

Så får vi ett error… Detta är normalt!

```
3  public sealed interface Vehicle permits Car, Truck {
4  }
5
6
7
```

Invalid permits clause: 'Truck' must directly implement 'Vehicle'

Implement 'com.krillinator.demo.Enterprise.sealedExamples.Vehicle' ⌥⇧⏎ More actions... ⌥⏎

com.krillinator.demo.Enterprise.sealedExamples

```
public record Truck
extends Record
```

Testing

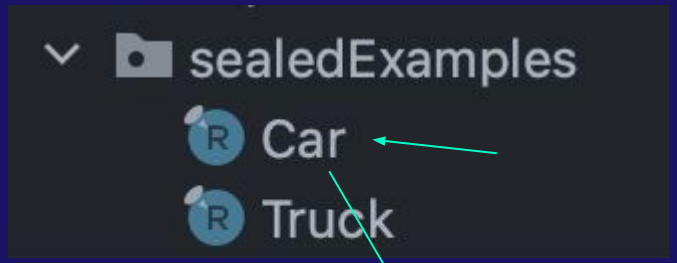
Problem?

Tryck på implement!

Problem?

Gör samma sak för Truck!

```
3  public sealed interface Vehicle permits Car, Truck {
4  }
5
```



Car?

Låt oss kika vad som förändrats!

```
3    public record Car() implements Vehicle {
4    }
```

Restriktion

Nu har vi en begränsning på vad som får överskridas eller inte!

Nyckelord

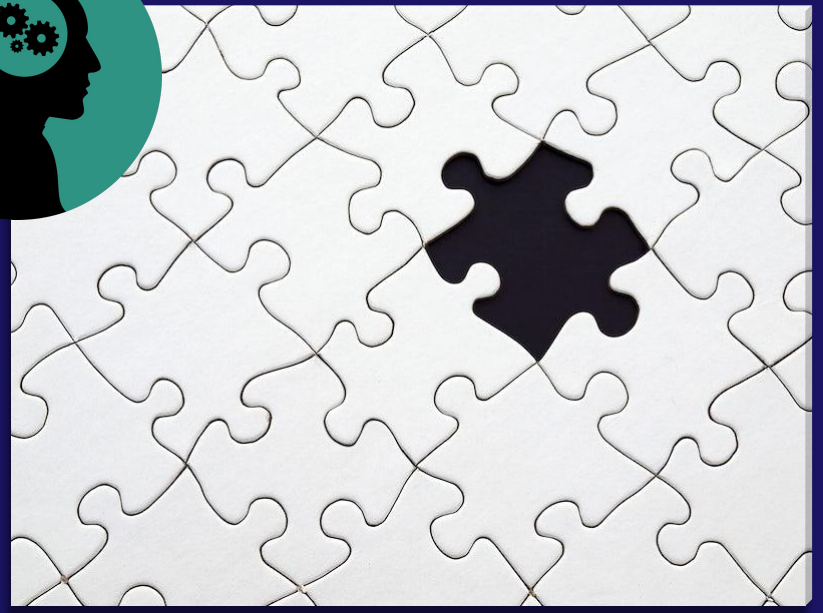
Non-sealed?

Sealed	Begränsar arv till nämnda klasser
Non-sealed	Vem som helst får ärva denna klass
Permits	Tillåter sagda klasser att få lov att ärva

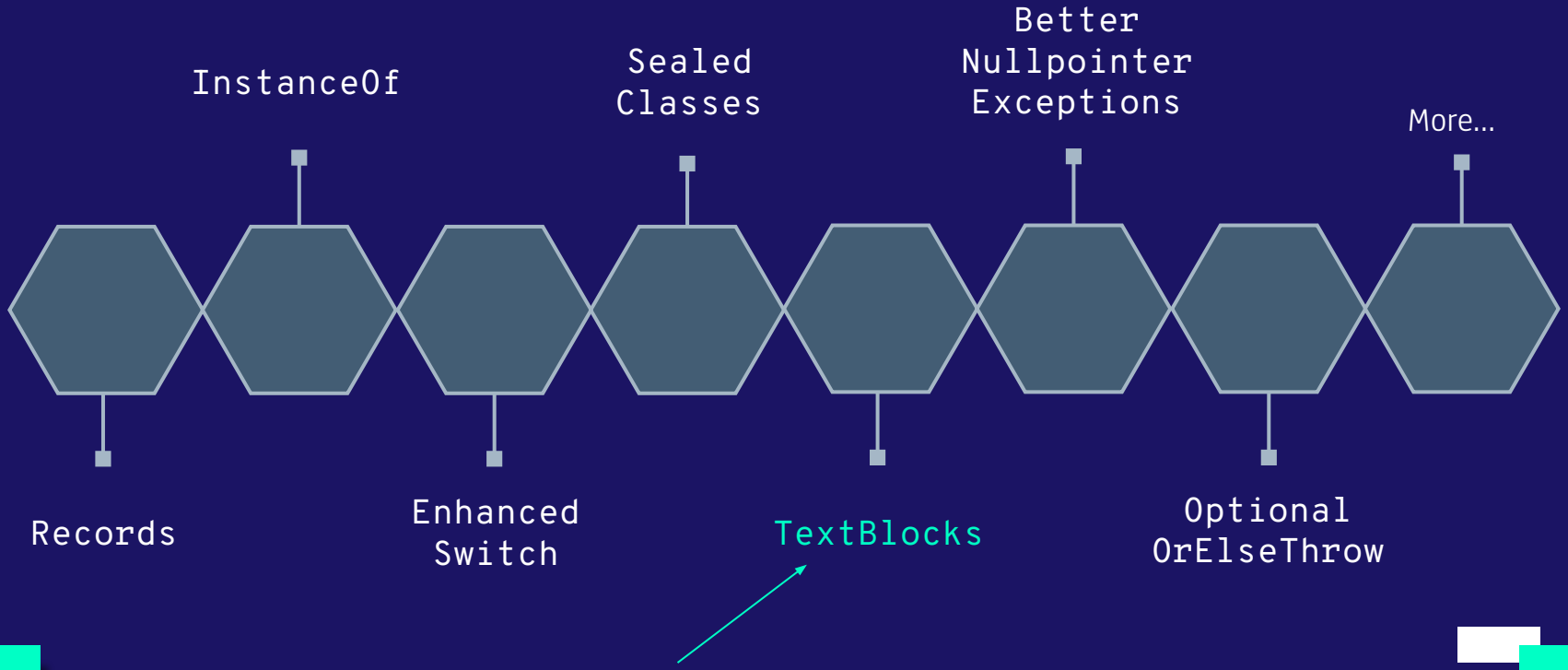
Instance of combination

```
// Composition with a class that uses Shape
public class ShapeHolder {
    private final Shape shape;

    public ShapeHolder(Shape shape) {
        this.shape = shape;
    }

    public double getArea() {
        if (shape instanceof Circle) {
            return ((Circle) shape).area();
        } else if (shape instanceof Rectangle) {
            return ((Rectangle) shape).area();
        }
        return 0;
    }
}
```

Frågor?

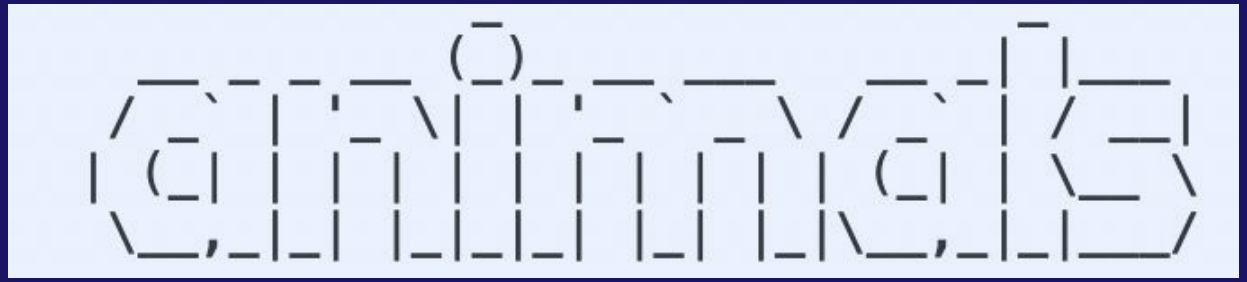



Textblocks


InstanceOf
Sealed Classes
Better Nullpointer Exceptions
More...
Records
Enhanced Switch
TextBlocks
Optional OrElseThrow

Textblocks

Tänk er att vi hade velat skriva ut en SOUT som ser ut som såhär:



Textblocks

Tänk er att vi hade velat skriva ut en SOUT som ser ut som såhär:

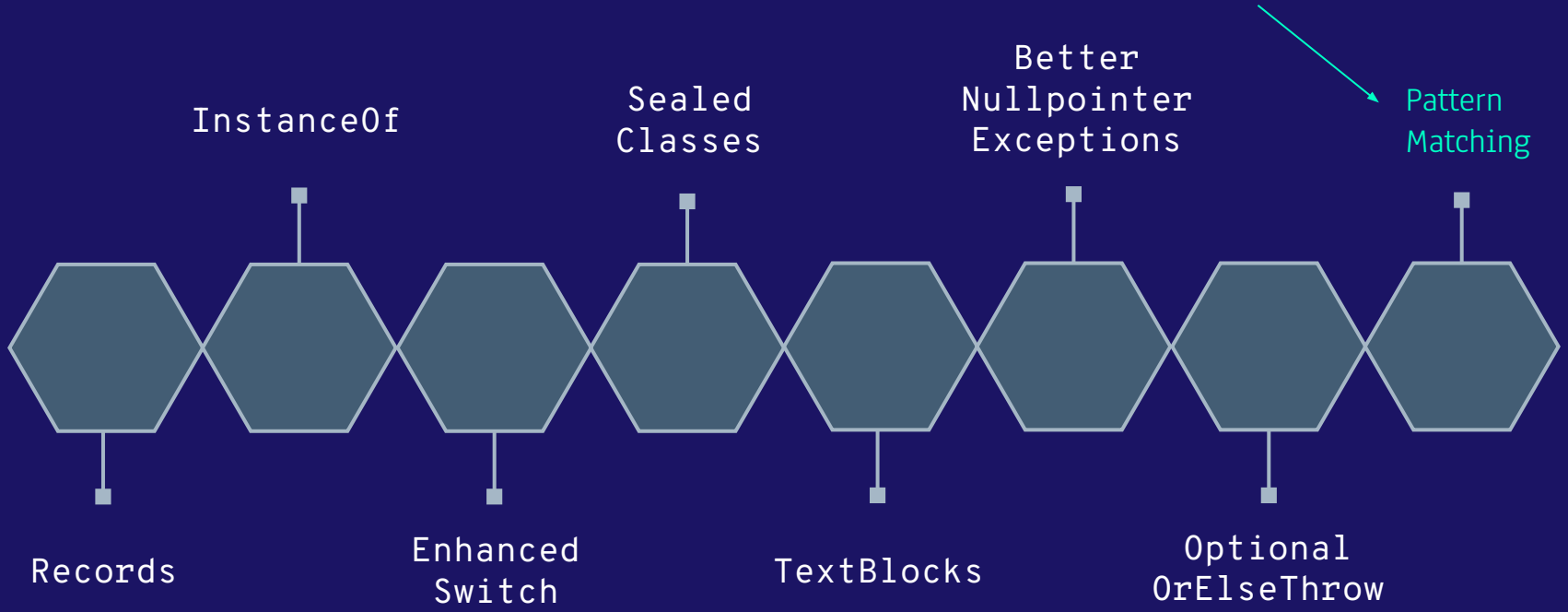
```
public void test() {
    System.out.println("" +
            "                _                    _     \n" +
            "   __ _ _ __ (_)_ __ ___   __ _| |___ \n" +
            "  / _` | '_ \\| | '_ ` _ \\ / _` | / __|\n" +
            " | (_| | | | | | | | | | | (_| | \\__ \\\n" +
            "  \\__,_|_| |_|_|_| |_| |_|\\__,_|_|___/\n" +
            "");
}
```

Textblocks

Inga ‘+’ tecken!

```
public void test() {
    System.out.println("""
                                _                 _~
                 __ _ _ __ (_)_ __ ___   __ _| |___
                / _` | '_ \\| | '_ ` _ \\ / _` | / __|
               | (_| | | | | | | | | | | (_| | \\__ \\
                \\__,_|_| |_|_|_| |_| |_|\\__,_|_|___/
            """);
}
```


Pattern Matching


InstanceOf
Sealed Classes
Better Nullpointer Exceptions
Pattern Matching
Records
Enhanced Switch
TextBlocks
Optional OrElseThrow

```java
Object obj = "Hello Java";

if (obj instanceof String) {
    String s = (String) obj; // must cast manually
    System.out.println(s.toUpperCase());
}
```

Java 8

Manuell ‘casting’ behövs.

Den är inte **smart** nog för att göra detta åt oss

```java
Object obj = "Hello Java";

// instanceof with pattern binding
if (obj instanceof String s) {
    System.out.println(s.toUpperCase()); // no cast needed
}

// switch with pattern matching
static String format(Object o) {
    return switch (o) {
        case Integer i -> "int " + i;
        case Long l    -> "long " + l;
        case String s  -> "String " + s.toLowerCase();
        default        -> "unknown";
    };
}
```

```java
Object obj = "Hello Java";

// instanceof with pattern binding
if (obj instanceof String s) {
    System.out.println(s.toUpperCase()); // no cast needed
}

// switch with pattern matching
static String format(Object o) {
    return switch (o) {
        case Integer i -> "int " + i;
        case Long l    -> "long " + l;
        case String s  -> "String " + s.toLowerCase();
        default        -> "unknown";
    };
}
```

Instanceof pattern matching was added:
Preview: Java 14 (JEP 305)
Second preview: Java 15
Final (standard): Java 16

Switch pattern matching was added:
Preview: Java 17 (JEP 406)
Second preview: Java 18
Third preview: Java 19
Fourth preview: Java 20
Final (standard): Java 21 (JEP 441)

06

Uppgifter & Eget Arbete

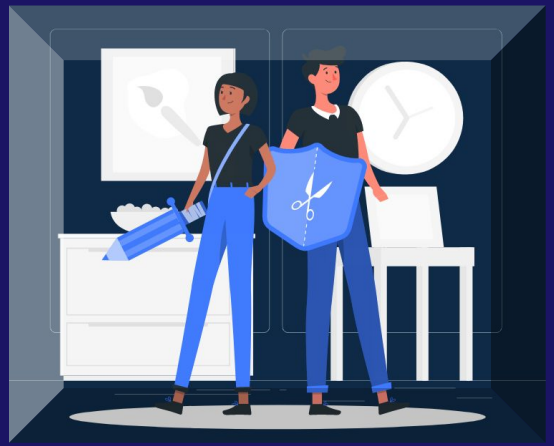
Uppgifter

Välkommen till första uppgiften!

Uppgifterna är till för att testa dina färdigheter och kunskaper för att både öva och repetera på det vi har arbetat med under föreläsningarna.

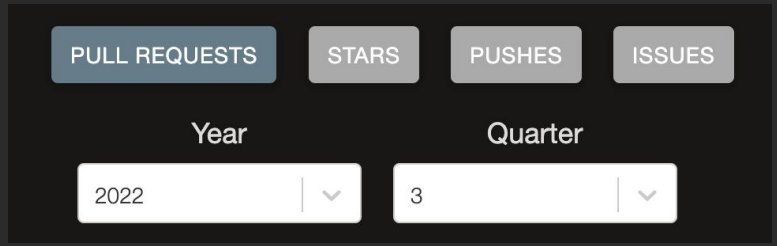
Dessa är **INTE** obligatoriska.

Men är starkt rekommenderat att arbeta med.



MINNS DU?

```
// Har Oracle övergivit Java?
// Vad händer med Java från och med nu?

/* Hur ser antalet Pull Requests ut för
    De olika programmeringsspråk?
    Glöm inte filtrera efter år!
    */
// https://madnight.github.io/githut/#/pull_requests/2022/3
```



MINNS DU?

```
/* Inom Java Web Services tittade vi
   på något som heter 'Records'
*/

// Vad är ett 'Record'?

// Vilka problem löser 'Records'?

// Kan en 'Entity' vara ett 'Record'?
// Varför / Varför inte?
```

FACIT

```
// Vad är ett 'Record'?
// En specialklass (Java 16+) för att skapa
'immutable' dataobjekt med mindre kod.

// Vilka problem löser 'Records'?
// Minskar boilerplate (getters, equals, hashCode,
toString), gör dataobjekt tydligare.

// Kan en 'Entity' vara ett 'Record'?
// Nej, normalt inte… en JPA-Entity kräver 'mutable'
fält och en tom konstruktor,
// vilket strider mot Records 'immutability'.
```

MINNS DU?

```
/*
    Varför är det viktigt för ett språk
som Java att vara open source, vilka
tekniska fördelar ger det utvecklare
och ekosystemet jämfört med ett stängt
språk?

*/
```

Facit

/*

Open source gör att **Java** kan utvecklas öppet via **JEP:ar**(Jdk, enhancement, proposal), vilket ger **transparens** och **möjlighet** för **communityn** att **påverka**.

Det möjliggör **snabba bugfixar** och **säkerhetspatchar**, fler implementationer av JDK (t.ex. Adoptium, Corretto, Zulu) och ett **robustare ekosystem**.

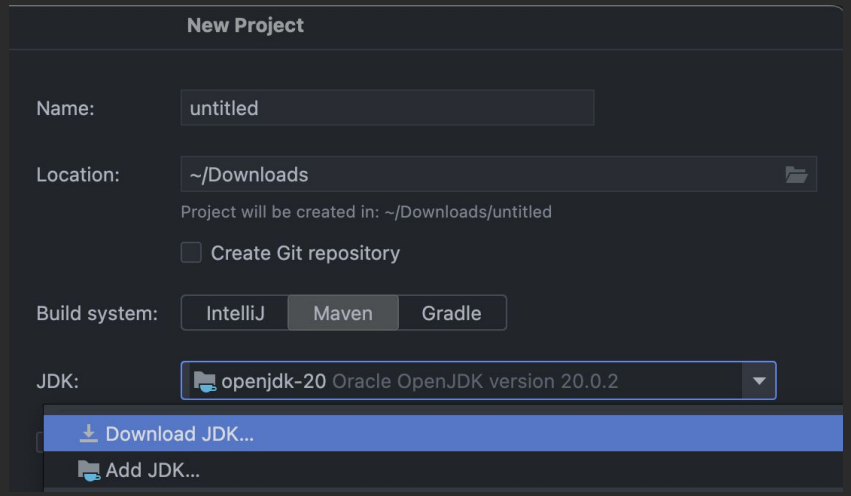
För **utvecklare** innebär det **fri tillgång**, **långsiktig stabilitet** och **större innovation** jämfört med ett stängt språk.

*/

```
1   // –Uppgift #1– //
2
3   /* INSTRUCTIONS
4
5       Skapa repository: EE_Uppgifter
6       Skapa projektet: EE_Uppgifter
7
8       Viktigt:
9         • För att detta ska fungera så bra som
10          möjligt är det viktigt att inte välja
11          några ‘dependencies’.
12          Välj därför dependencies enbart inom den
13          nya grenen som skapats.
14
15            ○ Exempelvis: om vi väljer Spring Web
16              (Imperative) så kommer vi inte kunna
17              arbeta inom Flux (Reactive).
18
19            ○ Låt Main vara fri från dependencies!
20
21
22  */
23
```

```
1   // -Uppgift #2- //
2
3   /* INSTRUCTIONS
4
5       Skapa en ny ‘branch‘
6           $ git branch lecture/X/Feature-Y
7           $ git branch -a (listar alla grenar)
8           $ git checkout ‘branchName’(navigerar)
9
10      Gör en ‘commit & push’!
11
12  */
13
14
15
16
17
18
19
20
21
22
23
```

```
1   // -Uppgift #1- //
2
3   /* INSTRUCTIONS
4
5       Öppna upp IntelliJ
6
7       Generera ett nytt Maven/Gradle projekt
8
9       Se till att Ladda ner JDK 8
10
11  */
12
13
14
15
16
17
18
19
20
21
22
23
```



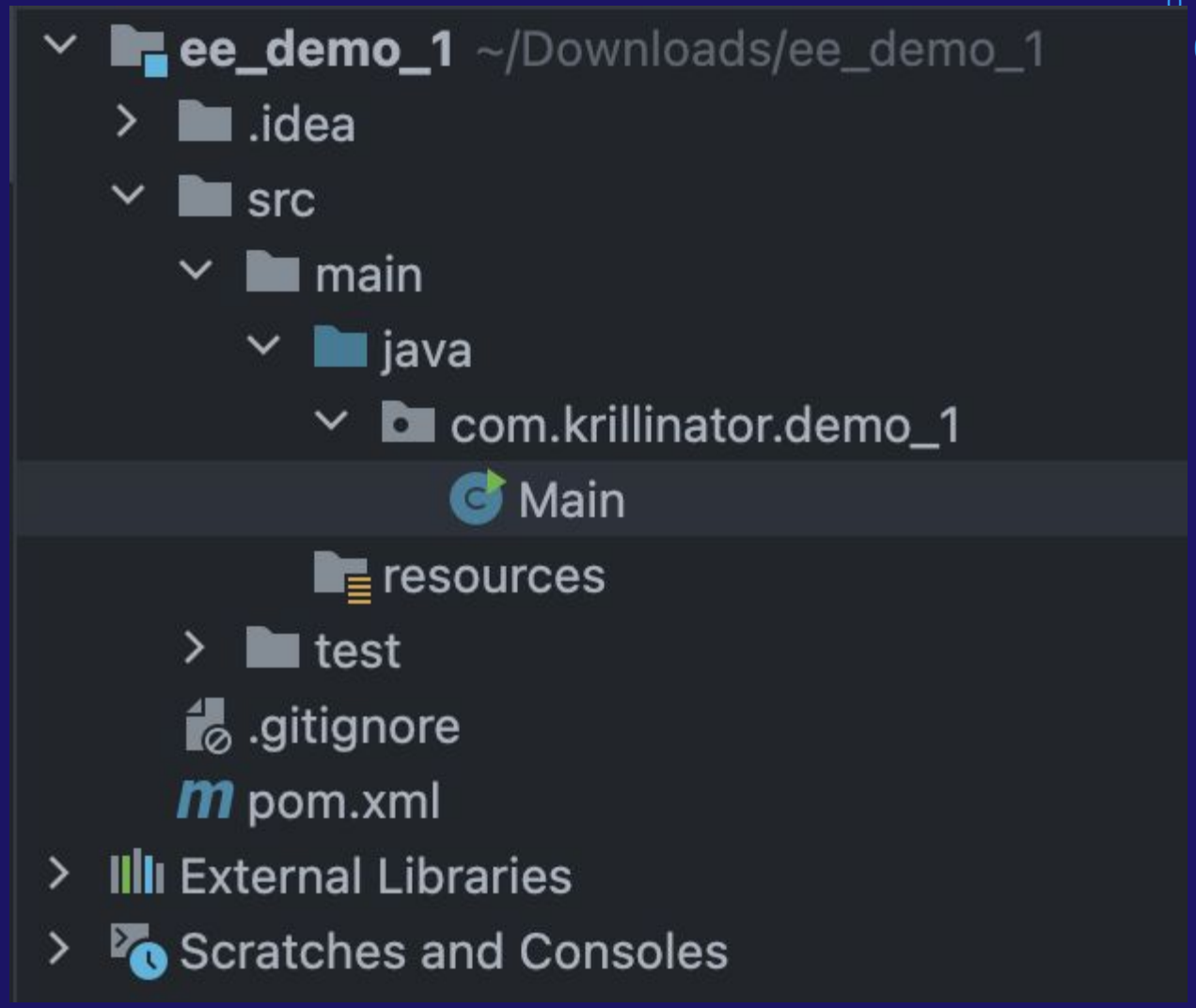
Uppgift #1

Get started with #1

```
1   // -Uppgift #2- //
2
3   /* INSTRUCTIONS
4
5       Välj version 1.8 = Java 8
6
7       Tryck sedan på Select
8
9       Generera sedan projektet
10  */
11
12
13
14
15
16
17
18
19
20
21
22
23
```

Uppgift #2

Om projektet inte har JDK 1.8
Så misslyckas uppgifterna direkt



Uppgift #3

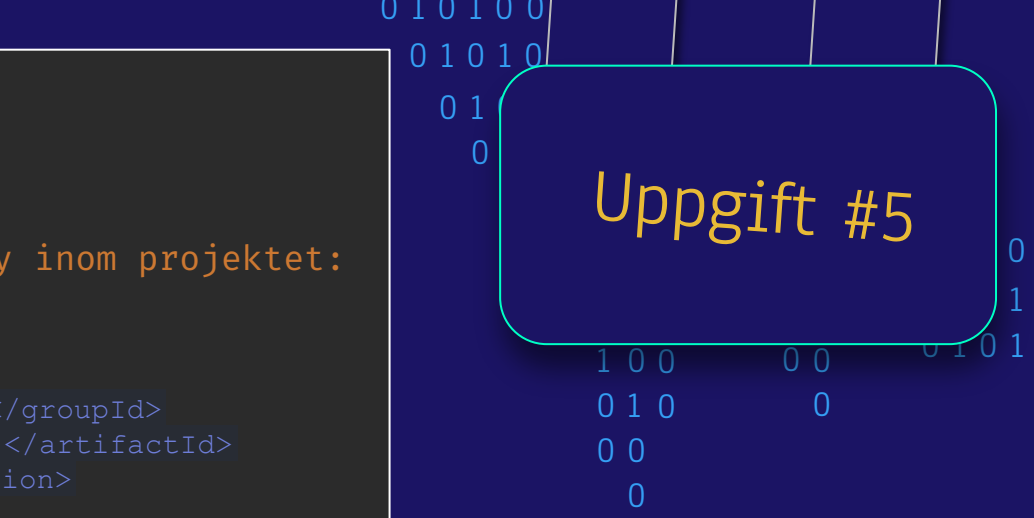
Följ projektets struktur

```
1   // -Uppgift #4- //
2
3   /* INSTRUCTIONS
4
5       Vad händer om vi försöker skapa ett
6       'Record'?
7
8       Prova följande kod:
9       List<String> names = List.of("Alice", "Bob", "Charlie");
10
11      Enhanced Switch case:
12      int day = 3;
13
14      String result = switch (day) {
15          case 1 -> "Monday";
16          case 2 -> "Tuesday";
17          case 3 -> "Wednesday";
18          default -> "Unknown";
19      };
20
21      System.out.println(result);
22
23  */
```

Uppgift #4

Vad händer här?

```
1   // -Uppgift #5- //
2                 Analys
3   /* INSTRUCTIONS
4
5       Lägg till följande Dependency inom projektet:
6
7       <dependencies>
8          <dependency>
9              <groupId>javax.validation</groupId>
10             <artifactId>validation-api</artifactId>
11             <version>2.0.1.Final</version>
12         </dependency>
13      </dependencies>
14
15      Använd @NotBlank annotation och importera…
16      Notera att importen ser lite annorlunda ut!
17
18      Analysera skillnaden
19  */
20
21
22
23
```



Att namnbyte har förändrats har gjort att en del bibliotek behövt anpassa sig och refaktorera sin kod.

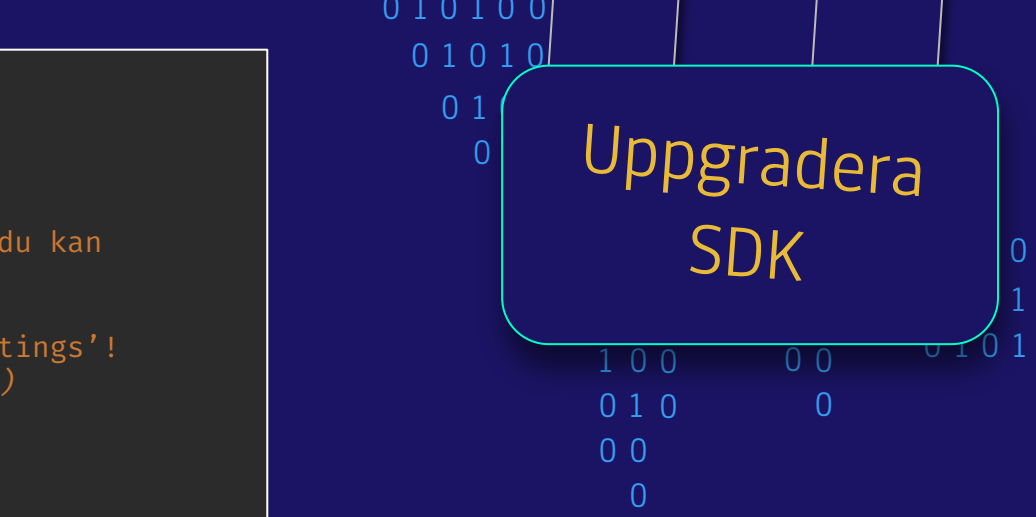
MINNS DU?

```
// Vilka problem löser 'enhanced
// switch statement'?

// Varför har man övergått till en
// Nyare variant?

Mer om Switch:
https://www.vojtechruzicka.com/java-enhanced-switch/
```

MINNS DU?

```
// Vad är ett 'Record' i jämförelse mot vanliga klasser?

Mer om records:
https://www.baeldung.com/java-record-keyword
```

```
1
2
3   /* INSTRUCTIONS
4
5        Välj nu att uppgradera SDK så att du kan
6        implementera records!
7
8        Glöm inte ändra Projekt SDK i ‘settings’!
9        (kolla också inom pom.xml / gradle)
10
11  */
12
13
14
15
16
17
18
19
20
21
22
23
```



```
1                   // –Uppgift #6– //
2
3   /* INSTRUCTIONS
4
5         Skapa en ny ‘Record’ klass
6
7         Lägg till en variabel av typen ‘String’
8
9         Instansiera den senare inuti Main.
10
11        Efter insansiering, lägg till en punkt
12        för att gå in i objektet och hämta ut
13        metoder. Ta nu reda på:
14        Om klassen har: ‘setters’ & ‘getters’?
15
16  */
17
18  // HINT & Examples
19  hint(" Försök att skriva ut ditt värde när du väl
20  instantiserat ditt objekt! ")
21
22
23
```

Uppgift #6

The basics of ‘Records’

Här är det viktigt att förstå sig på vad som redan finns tillgängligt inuti ett ‘record’!

```
1   // –Uppgift #7– //
2
3   /* INSTRUCTIONS
4
5       Skapa en ‘switch’ med breaks.
6       Ge den två ‘cases’
7
8       Märk av att er ‘switch’ kommer börja
9       lysa och kommer hänvisa er till att
10      Byta ut er ‘switch’ till en
11      ‘Enhanced switch’
12
13      Prova detta!
14
15  */
16
17
18
19
20
21
22
23
```

Back to basics with Switch!

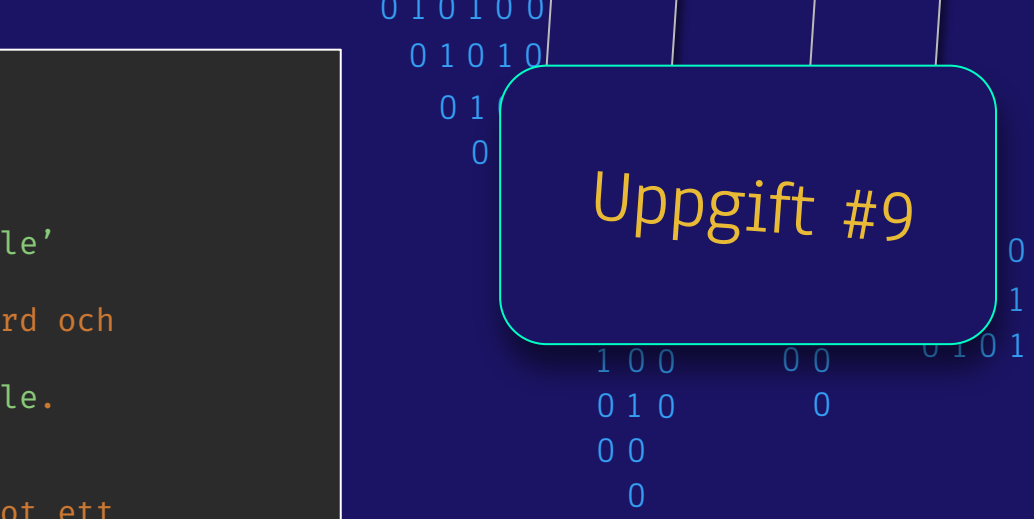
Här är det viktigt att följa med grundläggande strukturering av kod.
Switch är utmärkt som ersättning mot innästlade if/else satser!

```
1   // –Uppgift #8– //
2
3   /* INSTRUCTIONS
4
5       Skapa ett ‘interface’
6
7       Ge den ett ‘sealed’ nyckelord med
8       ‘permits’ följt av en klass.
9
10      Gör gärna ett ‘Animal interface’
11      Med två vanliga klasser ‘Dog’ & ‘Cat’
12      som ärver från ‘Animal’!
13
14  */
15
16  // HINT & Examples
17  hint(" Sealed är till för att begränsa åtkomst.
18  Permits är vad som tillåts. Non-sealed tillåter
19  oss att öppna upp igen för bättre kontrollering!
20  ")
21
22
23
```

Här är det viktigt att förstå sig på ‘arv’ AKA ‘inheritance’.

Konceptet kan annars kännas luddigt och otydligt.

```
1   // -Uppgift #9- //
2
3   /* INSTRUCTIONS
4
5       Skapa 'IAnimal, IMammal, IReptile'
6
7       Ge IAnimal ett 'sealed' nyckelord och
8       'permits' följt av
9       två interface: IMammal & IReptile.
10
11      Skapa nu en AnimalUtil klass
12      Innanför denna, så ska vi ta emot ett
13      Interface IAnimal.
14
15      Skapa två if-satser som kollar med
16      'instanceOf' om djur är från
17      IMammal eller IReptile
18  */
19
20  // HINT & Examples
21  hint(" Med hjälp av polymorfism kan vi begränsa och
22  typchecka mot objekt och dess arv")
23
```



Vi leker vidare med interfaces och 'sealed' med 'permits'

THANKS !

Do you have any questions?
kristoffer.johansson@sti.se

sti.learning.nu/

You can also contact me VIA Teams (quicker response)
Du kan också kontakta mig VIA Teams (Snabbare svar)