
sogeti
Part of Capgemini

Java 22 & beyond

SoJava Groningen, 06-06-2024

Wie ben ik?

Marcel Schutte

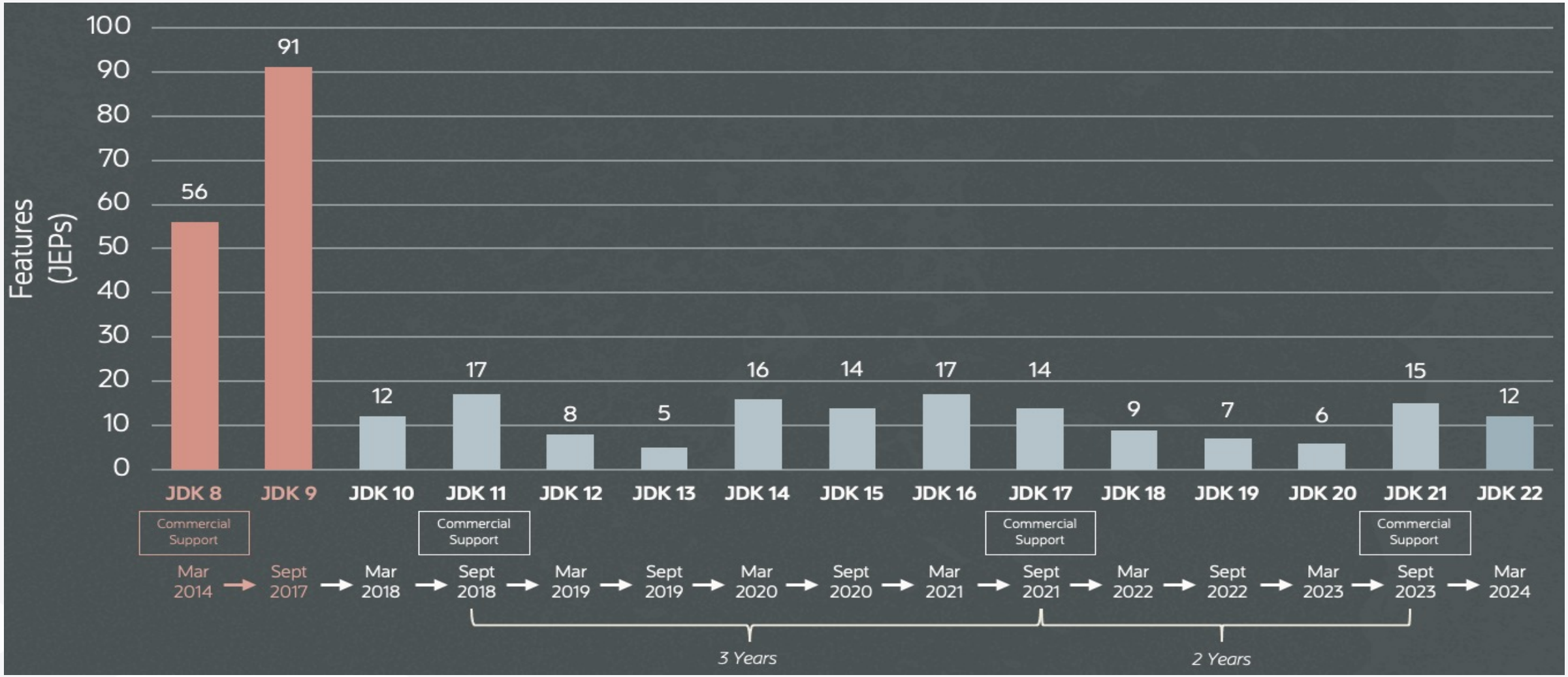
- Lead Solution Architect bij Sogeti
- Rijksoverheid en Finance
- Toepasbaarheid van AI
- Architectuurmodellering
- Spreker

marcel.schutte@sogeti.com

linkedin.com/in/marceljschutte

@PragArchitect

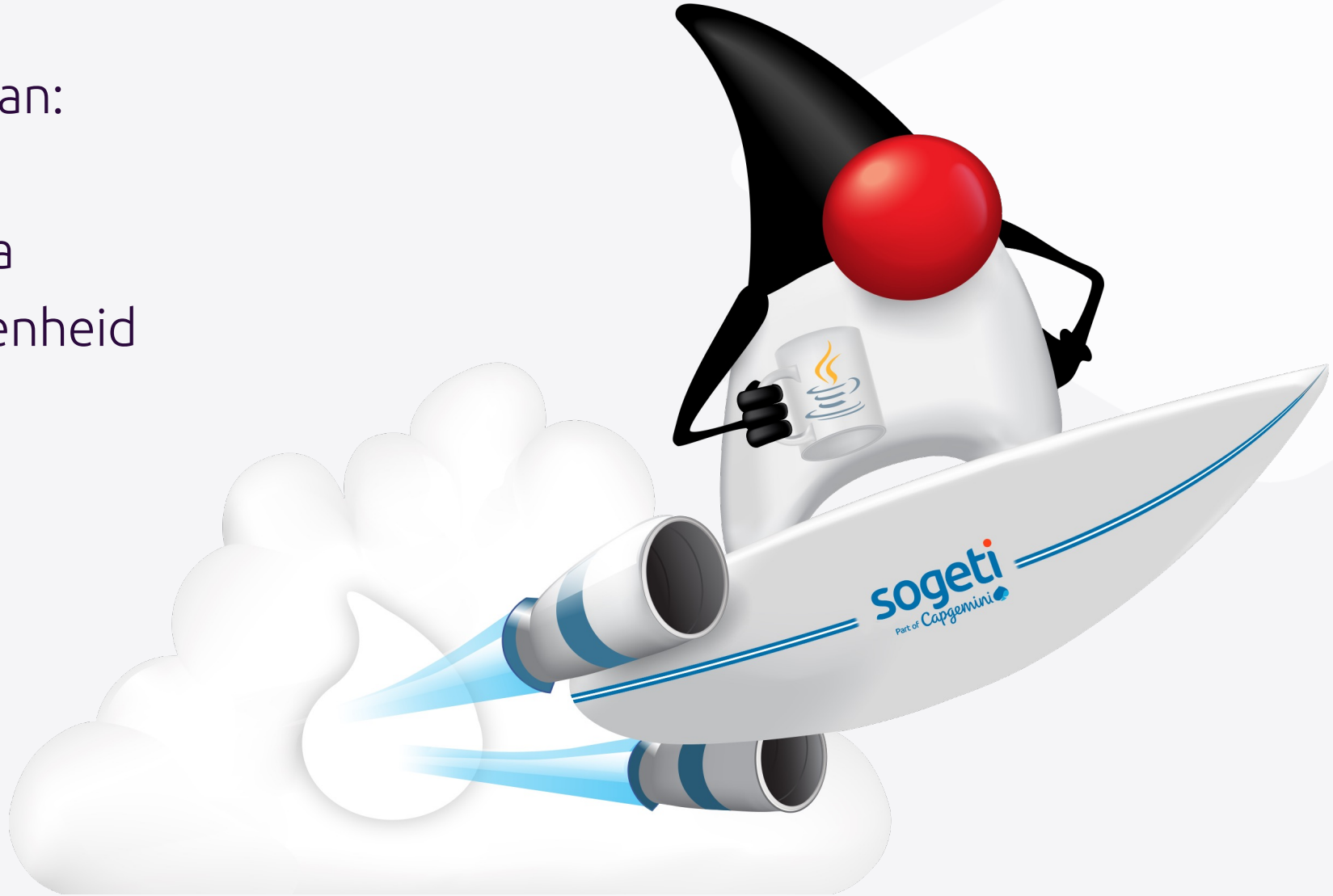
Release Cadence



bron: https://blogs.oracle.com

Release Cadence

- Elke release draagt bij aan:
  - Ontwikkeling van Java
  - Ontwikkelaarstevredenheid
  - Robuustheid van Java
  - Populariteit van Java



JDK Enhancement Proposals

Definitieve JEP’s

JEP 423: Region Pinning for G1

JEP 454: Foreign Function & Memory API

JEP 456: Unnamed Variables & Patterns

JEP 458: Launch Multi-File Source-code Programs

JEP’s in Preview

JEP 447: Statements before super()

JEP 457: Class-file API

JEP 459: String Templates

JEP 460: Vector API

JEP 461: Stream Gatherers

JEP 462: Structured Concurrency

JEP 463: Implicitly Declared Classes and Instance Main Methods

Definitieve JEP’s

JEP 423: Region Pinning for G1

Region pinning zet geheugenregio's met cruciale objecten vast tijdens GC.

Voordelen:

- Verminderde latency
- Betere GC pauzes
- Minimale regressie bij JNI critical regions
- Robuuste oplossing voor minor en major GC

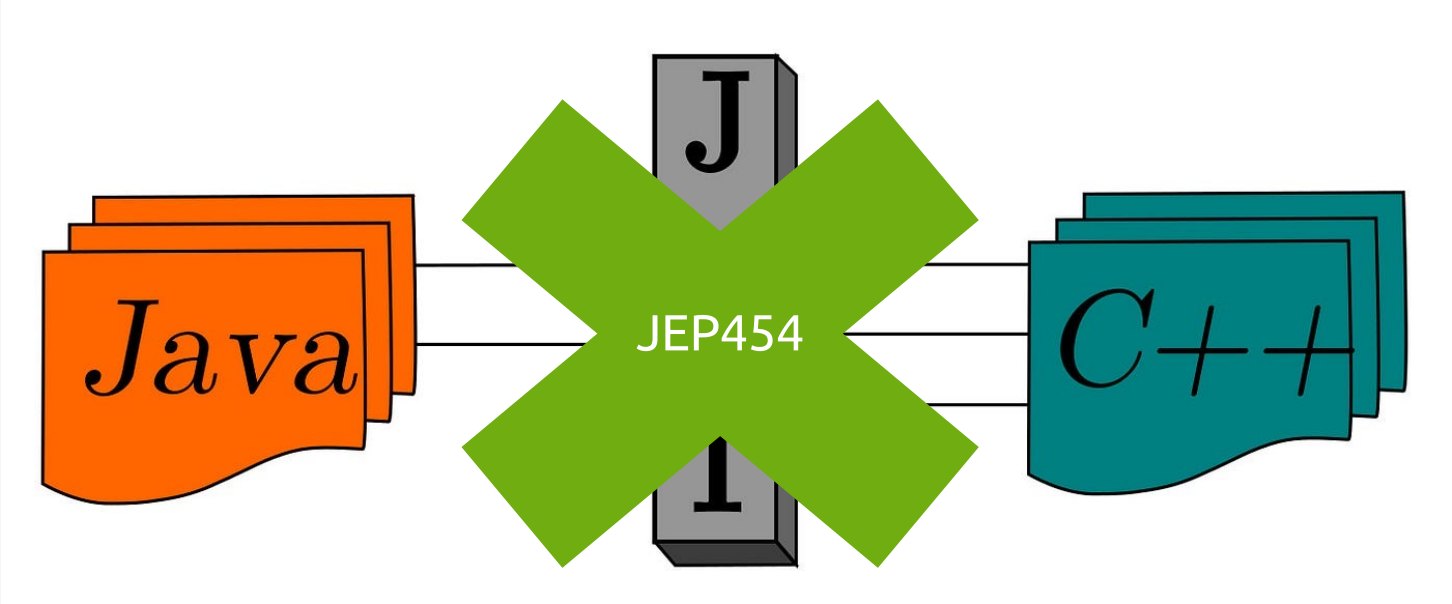
JEP 454: Foreign Function & Memory API

Vereenvoudigde native code-integratie:

Probleem: Complex JNI (Java Native Interface) bemoeilijkt native code-integratie.

Oplossing JEP 454:

- Prestatieverbetering
- Verhoogde flexibiliteit



JEP 454: Foreign Function & Memory API

Veiligheid en interoperabiliteit

Verbeterde veiligheid

- Gecontroleerde geheugentoegang
- Versterkte typeveiligheid

Verbeterde interoperabiliteit:

- Vereenvoudigde native code-integratie
- Samenwerking met andere talen

JEP456: Unnamed Variables & Patterns


```
// SIDE-EFFECT ONLY
static int count(Iterable<Order> orders) {
int total = 0;
for (Order unused : orders) {
total++;
}
return total;
}

// TRY-WITH-RESOURCES
try (var unused = ScopedContext.acquire()) {
// ...
}

// TRY-CATCH
try {
// ...
}
catch (Exception unused) {
// ...
}

// LAMBDA PARAMETERS
...stream.collect(Collectors.toMap(String::toUpperCase,
unused -> "NODATA"));

// ... and more
```

JEP456: Unnamed Variables & Patterns

Declaraties waar een _ is toegestaan:

- A local variable in a block (JLS §14.4.2)
- try-with-resources (JLS §14.20.3)
- for loop (JLS §14.14.1)
- Enhanced for loop (JLS §14.14.2)
- Exception for a catch block (JLS §14.20)
- Lambda parameter (JLS §15.27.1)

```
// SIDE-EFFECT ONLY
static int count(Iterable<Order> orders) {
int total = 0;
for (Order _ : orders) {
total++;
}
return total;
}

// TRY-WITH-RESOURCES
try (var _ = ScopedContext.acquire()) {
// ...
}

// TRY-CATCH
try {
// ...
}
catch (Exception _) {
// ...
}

// LAMBDA PARAMETERS
...stream.collect(Collectors.toMap(String::toUpperCase,
_ -> "NODATA"));
```

JEP 458: Launch Multi-File Source-code Programs

Komt voort uit JEP330: Launch Single-File Source-Code Programs:

```
class Prog {
    public static void main(String[] args) {
        Helper.run();
    }
}

class Helper {
    static void run() {
        System.out.println("Hello!");
    }
}
```

JEP 458: Launch Multi-File Source-code Programs

```java
// Prog.java
class Prog {
    public static void main(String[] args) {
        Helper.run();
    }
}

// Helper.java
class Helper {
    static void run() {
        System.out.println("Hello!");
    }
}
```

JEP's in Preview

Preview Features in Java

Preview features zijn:

- Nieuwe functies voor taal, virtuele machine of API.
- Volledig gespecificeerd en geïmplementeerd.
- Nog niet definitief.

Doel:

- Ontwikkelaars feedback laten geven in praktijk.
- Mogelijk permanente toevoeging aan Java SE Platform in de toekomst.

Preview Features in Java

Voordelen

- Duidelijkheid over toekomstige features
- Korte feedback loop

Nadelen

- Code met preview features uit oudere versie werkt mogelijk niet in nieuwere versie.

JDK Enhancement Proposals

Definitieve JEP’s

JEP 423: Region Pinning for G1

JEP 454: Foreign Function & Memory API

JEP 456: Unnamed Variables & Patterns

JEP 458: Launch Multi-File Source-code Programs

JEP’s in Preview

JEP 447: Statements before super()

JEP 457: Class-file API

JEP 459: String Templates

JEP 460: Vector API

JEP 461: Stream Gatherers

JEP 462: Structured Concurrency

JEP 463: Implicitly Declared Classes and Instance Main Methods

JEP 447: Statements before super()

```
public class PositiveBigInteger extends BigInteger {

    public PositiveBigInteger(long value) {
        super(value); // Potentially unnecessary work
        if (value <= 0)
            throw new IllegalArgumentException("non-positive value");
    }

}
```

JEP 447: Statements before super()

```
public class PositiveBigInteger extends BigInteger {

    public PositiveBigInteger(long value) {
        super(verifyPositive(value));
    }

    private static long verifyPositive(long value) {
        if (value <= 0)
            throw new IllegalArgumentException("non-positive value");
        return value;
    }

}
```

JEP 447: Statements before super()

```
public class PositiveBigInteger extends BigInteger {

    public PositiveBigInteger(long value) {
        if (value <= 0)
            throw new IllegalArgumentException("non-positive value");
        super(value);
    }

}
```

JEP 447: Statements before super()

```
public class Sub extends Super {

    public Sub(Certificate certificate) {
        super(prepareByteArray(certificate));
    }

    // Auxiliary method
    private static byte[] prepareByteArray(Certificate certificate) {
        var publicKey = certificate.getPublicKey();
        if (publicKey == null)
            throw new IllegalArgumentException("null certificate");
        return switch (publicKey) {
            case RSAKey rsaKey - > ...
            case DSAPublicKey dsaKey - > ...
            ...
            default - > ...
        };
    }

}
```

JEP 447: Statements before super()

```
public Sub(Certificate certificate) {
    var publicKey = certificate.getPublicKey();
    if (publicKey == null) {
        throw new IllegalArgumentException("null certificate");
    }
    final byte[] byteArray =
        switch (publicKey) {
            case RSAKey rsaKey - > ...
            case DSAPublicKey dsaKey - > ...
            ...
            default - > ...
        };

    super(byteArray);
}
```

JEP 457: Class-file API

Class files zijn essentieel voor het Java ecosysteem.

Tools en libraries kunnen class files lezen, genereren en aanpassen om programma's uit te breiden.

Frameworks gebruiken "on-the-fly" bytecode transformaties om functionaliteit toe te voegen.

JEP 457: Class-file API

Huidige situatie

Verschillende class-file libraries bestaan met eigen voor- en nadelen.

Frameworks bundelen vaak libraries zoals ASM, BCEL of Javassist.

Class-file format verandert sneller door de snelle release cyclus van de JDK (6 maanden).

JEP 457: Class-file API

Gevolgen

Frameworks stuiten vaker op nieuwere class files dan hun gebundelde library aankan.

Dit leidt tot fouten voor ontwikkelaars.

Framework ontwikkelaars moeten gokken of toekomstige class files werken met hun code.

JEP 457: Class-file API

Probleem:

- JDK gebruikt ASM voor class-file processing
- Update van ASM werkt vertragend

Voorstel:

- Standaard API voor class-files
- Java maakt standaard gebruik van deze API
- Externe tools/frameworks ondersteunen automatisch nieuwere class files
- Nieuwe taal en VM features worden makkelijker geadopteerd

JEP 459: String Templates

```
String s = x + " plus " + y + " equals " + (x + y);

String s = new StringBuilder()
    .append(x)
    .append(" plus ")
    .append(y)
    .append(" equals ")
    .append(x + y)
    .toString();

String s = String.format("%2$d plus %1$d equals %3$d", x, y, x + y);
String t = "%2$d plus %1$d equals %3$d".formatted(x, y, x + y);

MessageFormat mf = new MessageFormat("{0} plus {1} equals {2}");
String s = mf.format(x, y, x + y);
```

JEP 459: String Templates

```
String name = "Joan";
String info = STR.
"My name is \{name}";
assert info.equals("My name is Joan"); // true

// 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 and access fields
String s = STR.
"You have a \{getOfferType()} waiting for you!"; |
"You have a gift waiting for you!"
String t = STR.
"Access at \{req.date} \{req.time} from \{req.ipAddress}"; |
"Access at 2022-03-25 15:34 from 8.8.8.8"
```

JEP 460: Vector API

Mogelijkheden:

- efficiënte uitvoering van wiskundige vectoroperaties.
- Snelle berekeningen op ondersteunde hardware.
- Platformonafhankelijk en geen specialisatie van code nodig.

Voordelen:

- **Low-level vectoroperaties:** eenvoudige API.
- **Prestaties:** naadloze integratie van optimalisaties in bestaande Java-code.

JEP 461: Stream Gatherers

Gatherer interface

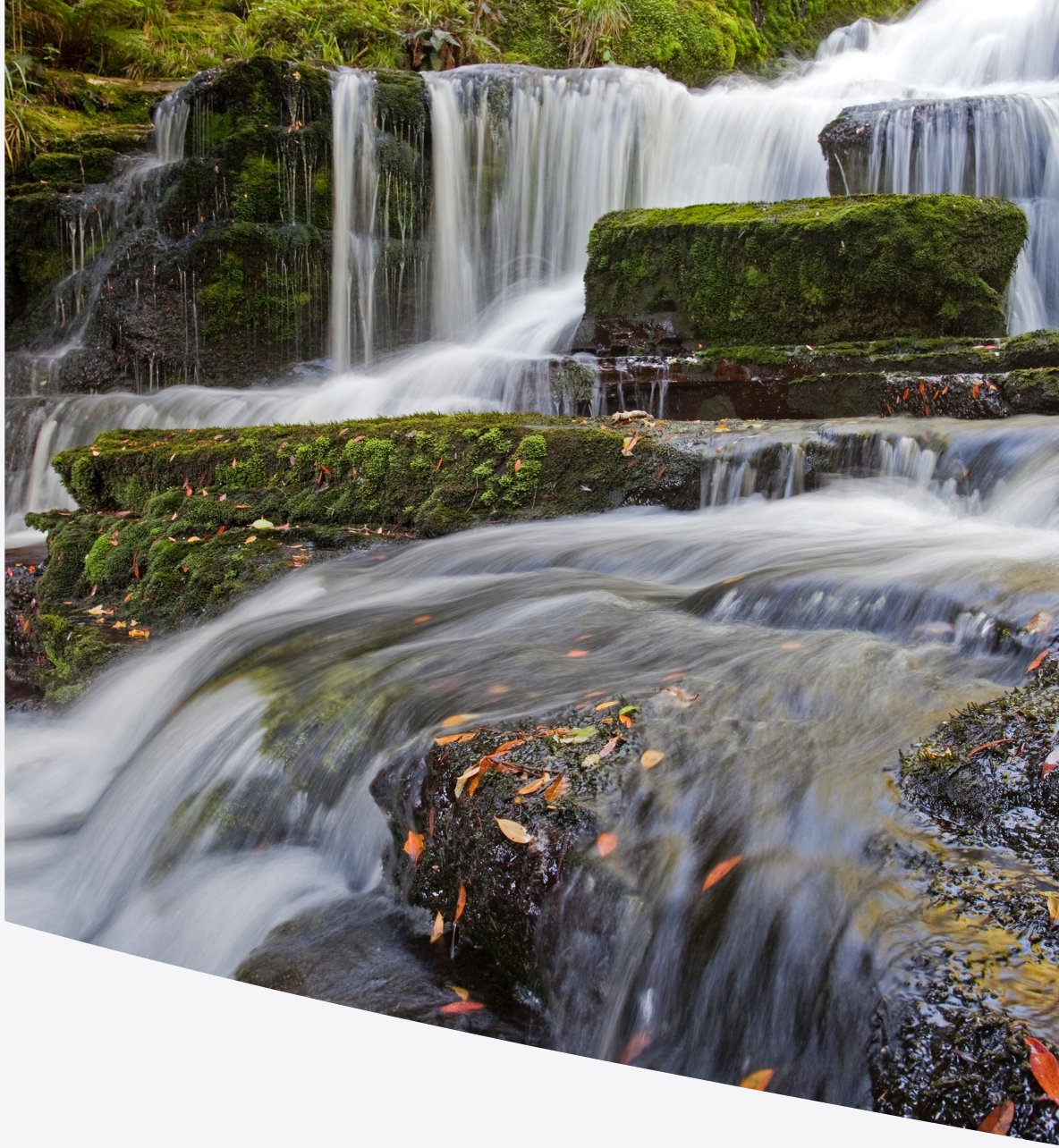
Deze interface definieert methoden om elementen te verzamelen, te combineren en het resultaat te finaliseren.

Mogelijkheden

Flexibele dataverwerking

Hergebruik van logica

Parallelle verwerking



JEP 462: Structured Concurrency

Concurrent tasks

- Gegroepeerde taken
- Voltooiing van scope
- Gecontroleerde annulering

Voordelen

- Betere stabiliteit
- Efficiëntere code

JEP 463: Implicitly Declared Classes and Instance Main Methods

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

JEP 463: Implicitly Declared Classes and Instance Main Methods

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

JDK Enhancement Proposals – Java 23, September 2024

JEP's in Preview

JEP 455: Primitive Types in Patterns, instanceof, and switch (Preview)

JEP 466: Class-File API (Second Preview)

JEP 469: Vector API (Eighth Incubator)

JEP 473: Stream Gatherers (Second Preview)

JEP 476: Module Import Declarations (Preview)

JEP 477: Implicitly Declared Classes and Instance Main Methods (Third Preview)

JEP 480: Structured Concurrency (Third Preview)

JEP 481: Scoped Values (Third Preview)

JEP 482: Flexible Constructor Bodies (Second Preview)

JDK Enhancement Proposals – Java 23, September 2024

Definitieve JEP’s

JEP 467: Markdown Documentation Comments

JEP 471: Deprecate the Memory-Access Methods in sun.misc.Unsafe for Removal

JEP 474: ZGC: Generational Mode by Default

Thank you for your attention!

marcel.schutte@sogeti.com

linkedin.com/in/marceljschutte

@PragArchitect

https://github.com/marceljschutte/slidedecks

About Sogeti

Part of the Capgemini Group, Sogeti makes business value through technology for organizations that need to implement innovation at speed and want a local partner with global scale. With a hands-on culture and close proximity to its clients, Sogeti implements solutions that will help organizations work faster, better, and smarter. By combining its agility and speed of implementation through a DevOps approach, Sogeti delivers innovative solutions in quality engineering, cloud and application development, all driven by AI, data and automation.

Capgemini is a global leader in partnering with companies to transform and manage their business by harnessing the power of technology. The Group is guided everyday by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of 360,000 team members in more than 50 countries. With its strong 55-year heritage and deep industry expertise, Capgemini is trusted by its clients to address the entire breadth of their business needs, from strategy and design to operations, fueled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering, and platforms. The Group reported in 2022 global revenues of €22 billion. Get the Future You Want.

Visit us at
www.sogeti.com