From Java 11 to Java 17 (and beyond)

Eder Ignatowicz
Principal Software Engineer
@ederign

Red Hat

O que eu acho mais maneiro do Java moderno :)

JEP 358: Helpful NullPointerExceptions

```
a.i = 99;

Exception in thread "main"
java.lang.NullPointerException
    at Prog.main(Prog.java:5)
```

JDK 14

```
Exception in thread "main"
java.lang.NullPointerException:
        Cannot assign field "i" because "a" is null
    at Prog.main(Prog.java:5)
```

JEP 355: Text Blocks

```
String sql = "SELECT COUNT(*) FROM table; -- Use this to determine rand_low and rand_high\n" +
             "\n" +
             "  SELECT *\n" +
             "    FROM table\n" +
             "   WHERE frozen_rand BETWEEN %(rand_low)s AND %(rand_high)s\n" +
             "ORDER BY RAND() LIMIT 1000";
```

```
var sql = """
                SELECT COUNT(*) FROM table; -- Use this to determine rand_low and rand_high

                  SELECT *
                    FROM table
                   WHERE frozen_rand BETWEEN %(rand_low)s AND %(rand_high)s
                ORDER BY RAND() LIMIT 1000
          """;
```

JEP 286: Local-Variable Type Inference

```
List<Student> students = new ArrayList<>();
students.removeIf(s -> s.getId() == desiredId);

var foo = 1;

var bestStudent = new Student("Dora");

for (var student: students) { /* … */ }

for (var i = 0; i < 10; i++) { /* … */ }
```

```
var x = y.bar();
```

```
Map<Long, Student> idToStudent = studentsRepository.getStudentId();
List<Student> enrolledStudents = studentsRepository.getEnrolledStudents();
Address addressOfBestStudent = studentsRepository.getAddress(bestStudent);

var idToStudent = studentsRepository.getStudentId();
var enrolledStudents = studentsRepository.getEnrolledStudents();
var addressOfTopStudent =  studentsRepository.getAddress(bestStudent);
```

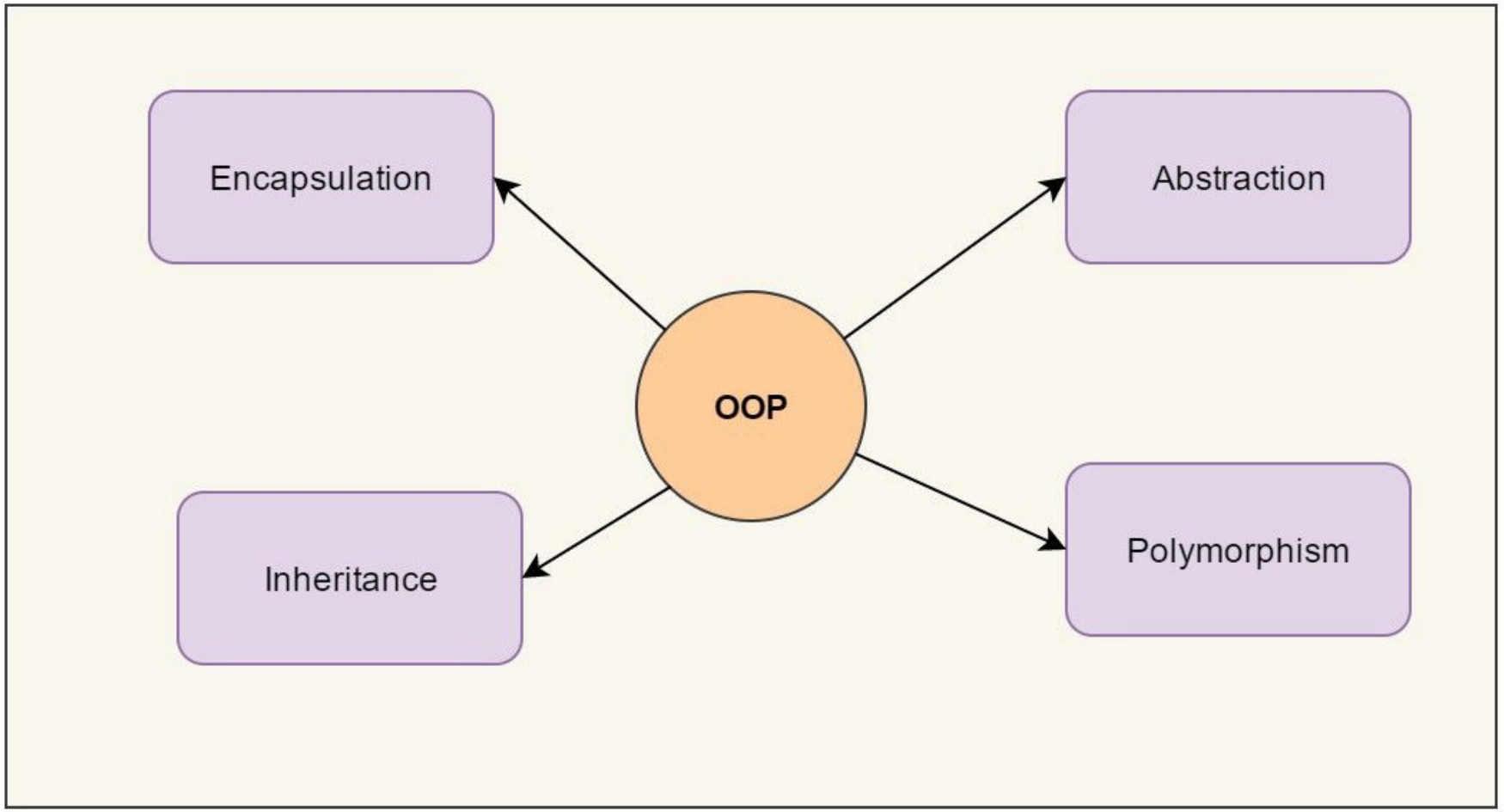
JEP 361: Switch Expressions (Standard)

```
switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        numLetters = 6;
        break;
    case TUESDAY:
        numLetters = 7;
        break;
    case THURSDAY:
    case SATURDAY:
        numLetters = 8;
        break;
    case WEDNESDAY:
        numLetters = 9;
        break;
}
```

```
switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> numLetters = 6;
    case TUESDAY                -> numLetters = 7;
    case THURSDAY, SATURDAY     -> numLetters = 8;
    case WEDNESDAY              -> numLetters = 9;
}
```

```
int numberOfDays = switch (day) {
    case FRIDAY, SUNDAY -> 6;
    case TUESDAY -> 7;
    case THURSDAY, SATURDAY -> 8;
    default -> {
        if (day == Days.WEDNESDAY) {
            yield 9;
        }
        else{
            yield -1;
        }
    }
};
```

JEP 395: Records

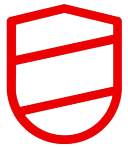


Four Pillars of Object Oriented Programming

Architectures

Evolutionary

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.



Microservices

Architectural style that structures an application as a collection of independent services.

Serverless

Incorporate third-party “Backend as a Service”, and/or that include custom code run as Functions.

Micro Frontends

Design approach in which a front-end app is decomposed into individual, semi-independent “microapps” working loosely together.

```java
package me.ederign;

public class SampleTask {

    private long id;
    private long owner;
    private String fieldA;
    private String fieldB;
    private String fieldC;
    private String fieldD;

...

}
```

```java
package me.ederign;

public class SampleTask {

    private long id;
    private long owner;
    private String fieldA;
    private String fieldB;
    private String fieldC;
    private String fieldD;

    public SampleTask(long id, long owner, String fieldA,
String fieldB, String fieldC, String fieldD) {
        this.id = id;
        this.owner = owner;
        this.fieldA = fieldA;
        this.fieldB = fieldB;
        this.fieldC = fieldC;
        this.fieldD = fieldD;
    }
}
```

```java
package me.ederign;

import java.util.Objects;

public class SampleTask {

    private long id;
    private long owner;
    private String fieldA;
    private String fieldB;
    private String fieldC;
    private String fieldD;

    public SampleTask(long id, long owner, String fieldA, String fieldB, String fieldC, String fieldD) {
        this.id = id;
        this.owner = owner;
        this.fieldA = fieldA;
        this.fieldB = fieldB;
        this.fieldC = fieldC;
        this.fieldD = fieldD;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        SampleTask that = (SampleTask) o;
        return id == that.id &&
                owner == that.owner &&
                Objects.equals(fieldA, that.fieldA) &&
                Objects.equals(fieldB, that.fieldB) &&
                Objects.equals(fieldC, that.fieldC) &&
                Objects.equals(fieldD, that.fieldD);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, owner, fieldA, fieldB, fieldC, fieldD);
    }
}
```

```java
package me.ederign;

import java.util.Objects;

public class SampleTask {

    private long id;
    private long owner;
    private String fieldA;
    private String fieldB;
    private String fieldC;
    private String fieldD;

    public SampleTask(long id, long owner, String fieldA, String fieldB, String fieldC, String fieldD) {
        this.id = id;
        this.owner = owner;
        this.fieldA = fieldA;
        this.fieldB = fieldB;
        this.fieldC = fieldC;
        this.fieldD = fieldD;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        SampleTask that = (SampleTask) o;
        return id == that.id &&
                owner == that.owner &&
                Objects.equals(fieldA, that.fieldA) &&
                Objects.equals(fieldB, that.fieldB) &&
                Objects.equals(fieldC, that.fieldC) &&
                Objects.equals(fieldD, that.fieldD);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, owner, fieldA, fieldB, fieldC, fieldD);
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public long getOwner() {
        return owner;
    }

    public void setOwner(long owner) {
        this.owner = owner;
    }

    public String getFieldA() {
        return fieldA;
    }

    public void setFieldA(String fieldA) {
        this.fieldA = fieldA;
    }

    public String getFieldB() {
        return fieldB;
    }

    public void setFieldB(String fieldB) {
        this.fieldB = fieldB;
    }

    public String getFieldC() {
        return fieldC;
    }

    public void setFieldC(String fieldC) {
        this.fieldC = fieldC;
    }

    public String getFieldD() {
        return fieldD;
    }

    public void setFieldD(String fieldD) {
        this.fieldD = fieldD;
    }
```

88 LINHAS!

Red Hat

```
package me.ederign;

public class SampleTask {

    private long id;
    private long owner;
    private String fieldA;
    private String fieldB;
    private String fieldC;
    private String fieldD;

...

}
```

```
public record SampleTask(long id,
                         long owner,
                         String fieldA,
                         String fieldB,
                         String fieldC,
                         String fieldD) {}
```

```
public record SampleTask(long id,
                         long owner,
                         String fieldA,
                         String fieldB,
                         String fieldC,
                         String fieldD) {}
```

Fields imutáveis

Constructors

equals, hashCode and toString

```
public record SampleTask(long id,
                         long owner,
                         String fieldA,
                         String fieldB,
                         String fieldC,
                         String fieldD) {}
```

"plain data" aggregate

(DTO, wrapper, transfer objects,

etc)

```
public record SampleTask(long id,
                         long owner,
                         String fieldA,
                         String fieldB,
                         String fieldC,
                         String fieldD) {}
```

Desacoplamento total para

data classes entre o estado e a sua

API

```
public record SampleTask(long id,
                         long owner,
                         String fieldA,
                         String fieldB,
                         String fieldC,
                         String fieldD) {}
```

Fit natural para externalização segura em sistemas distribuídos (serialização, marshalling para JSON/XML, mapping)

```java
public record SampleTask(long id,
                         long owner,
                         String fieldA,
                         String fieldB,
                         String fieldC,
                         String fieldD) {}
```

Aceita:

Novos construtores (até o canonico) com lógica adicional

Static fields/methods

Implementa interfaces

Annotations

```
 // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package me.ederign;

public final class SampleTask extends java.lang.Record {
    private final long id;
    private final long owner;
    private final java.lang.String fieldA;
    private final java.lang.String fieldB;
    private final java.lang.String fieldC;
    private final java.lang.String fieldD;

    public SampleTask(long id, long owner, java.lang.String fieldA, java.lang.String fieldB, java.lang.String
fieldC, java.lang.String fieldD) { /* compiled code */ }

    public long id() { /* compiled code */ }

    public long owner() { /* compiled code */ }

    public java.lang.String fieldA() { /* compiled code */ }

    public java.lang.String fieldB() { /* compiled code */ }

    public java.lang.String fieldC() { /* compiled code */ }

    public java.lang.String fieldD() { /* compiled code */ }

    public java.lang.String toString() { /* compiled code */ }

    public final int hashCode() { /* compiled code */ }

    public final boolean equals(java.lang.Object o) { /* compiled code */ }
}
```

```java
public record SampleTask(long id,
                         long owner,
                         String fieldA,
                         String fieldB,
                         String fieldC,
                         String fieldD) {}
```

"plain data" aggregate

Fit perfeito para

Arquiteturas Distribuídas

JEP 360/397/409: Sealed Classes

```
int process(Plant plant) {
    if (plant instanceof Cucumber) {
        return harvestCucumber(plant);
    } else if (plant instanceof Climber) {
        return sowClimber(plant);
    } else if (plant instanceof Herb) {
        return sellHerb(plant);
    } else if (plant instanceof Shrub) {
        return pruneShrub(plant);
    } else {
        System.out.println("Unreachable CODE. Unknown Plant type");
        return 0;
    }
}
```

```
sealed interface Shape
        permits Circle, Rectangle {
}

record Circle(Point center, int radius) implements Shape { }

record Rectangle(Point lowerLeft, Point upperRight) implements Shape { }
```

```
sealed interface Shape
        permits Circle, Rectangle {
}

record Circle(Point center, int radius) implements Shape { }

record Rectangle(Point lowerLeft, Point upperRight) implements Shape { }
```

Vantagens

Designer da API controla melhor as implementações

O compilador pode inferir mais coisas...

Desacopla accessibilidade de extensibilidade

Sealed Classes + Records

```
sealed interface Shape
        permits Circle, Rectangle {
}

record Circle(Point center, int radius) implements Shape { }

record Rectangle(Point lowerLeft, Point upperRight) implements Shape { }
```

Sealed Classes ~= 'Sum Types'

Sum types expressam todas as variações de uma estrutura de Dados

O conjunto de todos os tipos Shape s é igual ao conjunto de todos os Circle c mais todos os Rectable S

```
sealed interface Shape
        permits Circle, Rectangle {
}

record Circle(Point center, int radius) implements Shape { }

record Rectangle(Point lowerLeft, Point upperRight) implements Shape { }
```

Record ~= 'Product Types'

Type-theoretic view de "structs" e "tuples".

Todos os possíveis estados (state space) é um subconjunto do produto cartesianos de todos seus componentes.

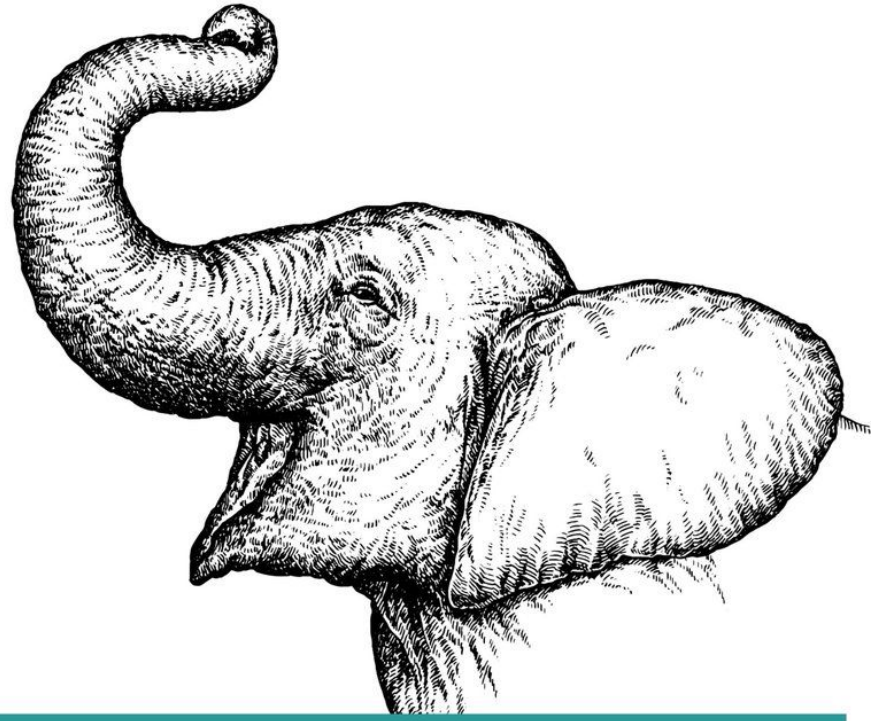
```
sealed interface Shape
        permits Circle, Rectangle {
}

record Circle(Point center, int radius) implements Shape { }

record Rectangle(Point lowerLeft, Point upperRight) implements Shape { }
```

Code smell???

Isto não viola o encapsulamento?

Pq o Java tá fazendo isto?

The answer to every programming question ever conceived



It Depends

The Definitive Guide

O RLY?

@ThePracticalDev

Red Hat

"Sealed classes work together with records and pattern matching to support a more data-centric form of programming."
Brian Goetz

Sealed Classes + Records

JEP 305/JEP 375/394: Pattern Matching for instanceof

```
static int getCenter(Shape shape) {
    if (shape instanceof Rectangle) {
        return ((Rectangle) shape).upperRight().x;
    } else if (shape instanceof Circle) {
        return ((Circle) shape).radius();
    }
    return -1;
}
```

```
static int getCenterJ16(Shape shape) {
        if (shape instanceof Rectangle r) {
            return r.upperRight();
        } else if (shape instanceof  Circle r && r.getRadios()!= null) {
            return r.radius();
        }
        return -1;
    }
```

```
class Example2 {
    Point p;

    void test2(Object o) {
        if (o instanceof Point p) {
            // p refers to the pattern variable
            ...
        } else {
            // p refers to the field
            ...
        }
    }
}
```

JDK 17+

JEP 406: Pattern matching for switch (Preview)

Red Hat

```
static String formatter(Object o) {
    String formatted = "unknown";
    if (o instanceof Integer i) {
        formatted = String.format("int %d", i);
    } else if (o instanceof Long l) {
        formatted = String.format("long %d", l);
    } else if (o instanceof Double d) {
        formatted = String.format("double %f", d);
    } else if (o instanceof String s) {
        formatted = String.format("String %s", s);
    }
    return formatted;
}
```

```
static String formatterPatternSwitch(Object o) {
    return switch (o) {
        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        -> o.toString();
    };
}
```

```
static void testFooBar(String s) {
    if (s == null) {
        System.out.println("oops!");
        return;
    }
    switch (s) {
        case "Foo", "Bar" -> System.out.println("Great");
        default           -> System.out.println("Ok");
    }
}
```

```
static void testFooBar(String s) {
    switch (s) {
        case null          -> System.out.println("Oops");
        case "Foo", "Bar" -> System.out.println("Great");
        default            -> System.out.println("Ok");
    }
}
```

```
static int switchExpressionRegular(A in) {
    return switch (in) {
        case ChildA1 c1 -> c1.getValue();
        case ChildA2 c2 -> c2.getValue();
        case A a -> a.getValue();
    };
}

static int switchExpressionSealed(SealedA in) {
    return switch (in) {
        case ChildSA1 c1 -> c1.getValue();
        case ChildSA2 c2 -> c2.getValue();
        case ChildSA3 c3 -> c3.getValue();
    };
}
```

```
static void test(Object o) {
    if ((o instanceof String s) && s.length() > 3) {
        System.out.println(s);
    } else {
        //no s in scope
        System.out.println("Not a string");
    }
}
```

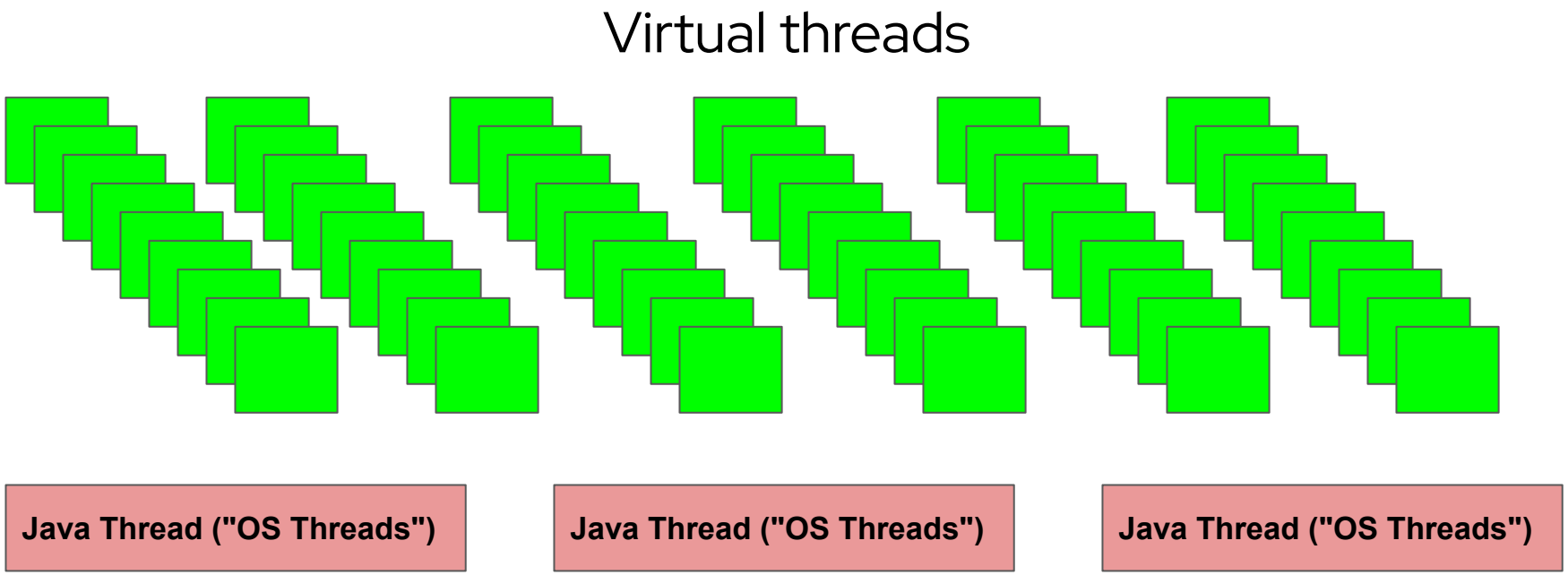
JDK 18+

Project Loom

```
for (int i = 0; i < parameter; i++) {
    Runnable run = () -> {
        //task bem longa e complexa
    };
    Thread th = new Thread(runnable);
    th.start();
}
```

Virtual threads

- Fim do mapeamento 1:1 de "Threads" do Java com Threads do Sistema Operacional
- Extensao da API de Threads
- Mesmo conceito que nós já conhecemos
- São multiplexadas em cima de um thread pool do OS
-



Virtual threads

```
Thread virtualThread1 = Thread.startVirtualThread(() -> {
 //task longa
});

Thread virtualThread2 = Thread.builder().virtual().task(() -> {
 //task longa com blocking I/O
}).build();
virtualThread2.start()
```

```
public void process(Operation op){
    databaseService.process(op);
    auditService.process(op);
    analyticsService.process(op);
    cacheService.process(op);
}
```

Structured Concurrency

- Structured concurrency possibilita desenvolvedores escreverem código concorrente num bloco de código visível
- Código parece síncrono, mas é assíncrono
- Todas as tasks são finalizadas depois de terminar o bloco de código
- Futuro de todas as APIs Java

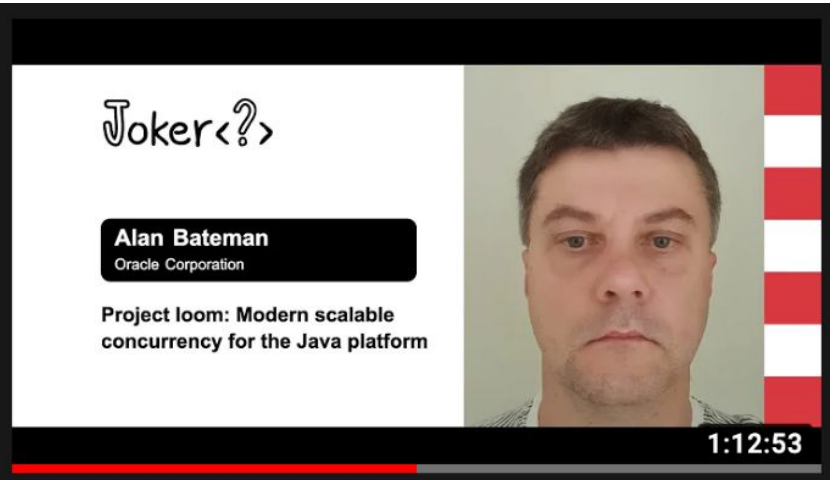
```
try (var executor = Executors.newVirtualThreadExecutor()) {
        executor.submit(() -> databaseService.process(op));
        executor.submit(() -> auditService.process(op));
        executor.submit(() -> analyticsService.process(op));
        // for loop pra criar 'n'
        executor.submit(() -> cacheService.process(op));
}
```

The State of Project Loom with Ron Pressler

2.7K views • 4 months ago

nipafx

Conversation with **Project Loom** lead Ron Pressler about the project's core mission, challenges like interaction with debuggers ...



Alan Bateman — Project loom: Modern scalable concurrency for the Java platform (ENG + RUS SUB)

5.8K views • 9 months ago

JUG .ru

. . Concurrent applications, those serving multiple independent application actions simultaneously, are the bread and butter of ...

4K CC

Red Hat

JDK 18+

JEP 405: Record Patterns & Array Patterns (Preview)

Red Hat

```
// Old code
if (o instanceof String) {
    String s = (String)o;
    ... use s ...
}

// New code
if (o instanceof String s) {
    ... use s ...
}
```

```
record Point(int x, int y) {}

static void printSum(Object o) {
    if (o instanceof Point p) {
        int x = p.x();
        int y = p.y();
        System.out.println(x+y);
    }
}
```

```java
record Point(int x, int y) {}

void printSum(Object o) {
    if (o instanceof Point(int x, int y)) {
        System.out.println(x+y);
    }
}
```

```
record Point(int x, int y) {}
enum Color { RED, GREEN, BLUE }
record ColoredPoint(Point p, Color c) {}
record Rectangle(ColoredPoint upperLeft, ColoredPoint lowerRight) {}

static void printColorOfUpperLeftPoint(Rectangle r) {
    if (r instanceof Rectangle(ColoredPoint(Point p, Color c), ColoredPoint lr)) {
        System.out.println(c);
    }
}
```

```java
static void printFirstTwoStrings(Object o) {
    if (o instanceof String[] sa && sa.length >= 2) {
        String s1 = sa[0];
        String s2 = sa[1];
        System.out.println(s1 + s2);
    }
}

static void printFirstTwoStrings(Object o) {
    if (o instanceof String[] { String s1, String s2, ... }){
        System.out.println(s1 + s2);
    }
}
```

JEP 417: Vector API (Third Incubator)

```
void scalarComputation(float[] a, float[] b, float[] c) {
   for (int i = 0; i < a.length; i++) {
        c[i] = (a[i] * a[i] + b[i] * b[i]) * -1.0f;
   }
}
//using the Vector API:

static final VectorSpecies<Float> SPECIES = FloatVector.SPECIES_PREFERRED;

void vectorComputation(float[] a, float[] b, float[] c) {
    int i = 0;
    int upperBound = SPECIES.loopBound(a.length);
    for (; i < upperBound; i += SPECIES.length()) {
        // FloatVector va, vb, vc;
        var va = FloatVector.fromArray(SPECIES, a, i);
        var vb = FloatVector.fromArray(SPECIES, b, i);
        var vc = va.mul(va)
                   .add(vb.mul(vb))
                   .neg();
        vc.intoArray(c, i);
    }
    for (; i < a.length; i++) {
        c[i] = (a[i] * a[i] + b[i] * b[i]) * -1.0f;
    }
}
```

Vector API: SIMD Programming in Java

3K views • 7 months ago

Java

SIMD **#Java** #OpenJDK The Vector API enables developers to write platform-agnostic, data-parallel programs where single ...

The Vector API in JDK 17

3.9K views • 1 month ago

Java

... Example repo ⇨ https://github.com/PaulSandoz/vector-api-dev-live-10-2021 ◦ Vector API: **SIMD Programming in Java** ...

2:14 ... examples i recommend you watching the replay of the prior **java** innovations talk vector api singly **programming in java** that w...

Red Hat

Thank you

Eder Ignatowicz.

Principal Software Engineer

@ederign


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat