# Java™ 17 PREVIEW

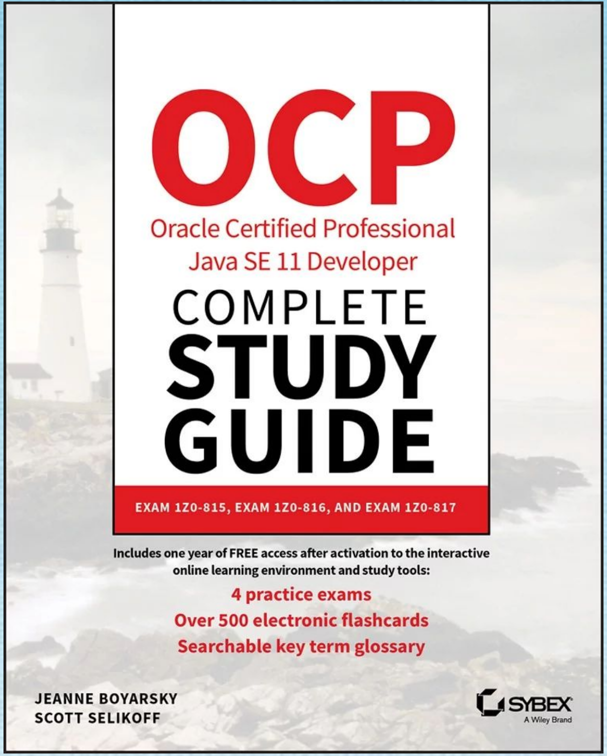
Michel ten Voorde
Software Engineer Java

eonics

## $ whoami

michel.tenvoorde@eonics.nl

# 1Z0-819 Certification @ Eonics


OCP
Oracle Certified Professional
Java SE 11 Developer
COMPLETE
STUDY
GUIDE
EXAM 1Z0-815, EXAM 1Z0-816, AND EXAM 1Z0-817
Includes one year of FREE access after activation to the interactive online learning environment and study tools:
4 practice exams
Over 500 electronic flashcards
Searchable key term glossary
JEANNE BOYARSKY
SCOTT SELIKOFF
SYBEX
A Wiley Brand

# New in Java 12 - 15

- Java 17 (expected Sep 2021) will be a new LTS version.
- Not everything new can be covered…
- Since Java 11, a total of 43 JEPs have been added.
- The focus is developer-centered.

# Files::mismatch

- Useful for determining if 2 files have the same content.
- By design, very similar to `Arrays::mismatch`.

**Example**:

```
long differAtPosition = Files.mismatch(path1, path2);
```

# Compact Number Formatting

- Provides support for locale-sensitive compact number formatting
- 1000 becomes `1K` (short style) or `1 thousand` (long style)

**Example**:

```
NumberFormat compactFormatter = NumberFormat.getCompactNumberInstance(
    Locale.US, NumberFormat.Style.LONG);
String result = compactFormatter.format(9000);
System.out.println("It's over " + result + "!!!");
```

Outputs: `It’s over nine thousand!!!`

# Switch Expressions

- Extends `switch` so it can be used both as a statement and an expression.
- The statement syntax has been revamped, though the old one is still available.

**Example**:

```
switch (LocalDate.now().getDayOfWeek()) {
    case MONDAY               -> System.out.println("Sigh...");
    case TUESDAY, WEDNESDAY -> System.out.println("Hang in there...");
    case THURSDAY             -> System.out.println("Almost there now..!");
    case FRIDAY               -> System.out.println("Party time!");
}
```

# Switch Expressions

- The expression syntax has the additional restriction that all input paths *must* be covered. Most of the time*, this implies using the `default` clause.

**Example**:

```
System.out.println(
    switch (interestingInt) {
        case 1     -> "One";
        case 2     -> "Two";
        default    -> "Many";
    }
);
```

## Pattern Matching for instanceof

- Adds *pattern matching* to the `instanceof` operator.
- Shortens the test - cast - declaration boilerplating.

**Old**:

```
if (obj instanceof Point) {
    Point p = (Point) obj;
    return x == p.x && y == p.y;
} else {
    return false;
}
```

**New**:

```
if (obj instanceof Point p) {
    return x == p.x && y == p.y;
} else {
    return false;
}
```

# Helpful NullPointerExceptions

- Improves the usability of NullPointerExceptions by describing which variable actually was null.

```
// c == null
System.out.println(a.b.c.d);
```

**Old**:

```
Exception in thread "main" java.lang.NullPointerException
```

**New**:

```
Exception in thread "main" java.lang.NullPointerException: Cannot read
field "d" because "a.b.c" is null
```

# Text Blocks

- Adds a new multiline string literal to Java.
- Possibilities to customize whitespace removal (see exercices).

**Old**:

```
@Query("select c from Customer c " +
       "where c.country = :country " +
       "and c.active = 1 ")
)
```

**New**:

```
@Query("""
        select c from Customer c
        where c.country = :country
        and c.active = 1
        """)
```

# Text Blocks

- New `String::formatted` method to simplify variable substitution.
- Basically, it's the reverse of `String::format`.

**Example**:

```
String welcome = """
    Well
    hello
    there
    %s""".formatted(name);
```

# Sealed Classes

- Sealed classes and interfaces restrict which other classes or interfaces may extend or implement them.
- This makes it possible for a superclass to be *accessible* without giving the ability to *extend* it.

**Example:**

```
public abstract sealed class Shape
    permits Triangle, Square, Circle {...}
```

# Sealed Classes

- Permitted subclasses are restrained in that they:
  - have to directly extend the sealed class;
  - have to be in the same module or package;
  - have to declare how to continue the sealing initiated by its superclass.

**Example:**

```
public final class Triangle extends Shape {...}
public sealed class Square extends Shape permits Rectangle {...}
public non-sealed class Circle extends Shape {...}
```

# Records

- Records are immutable data classes that are defined only by the type and name of its fields.

**Example:**

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

- Everything else is generated by the compiler:
  - Corresponding `private final` fields
  - getters (without the *get*-prefix)
  - a constructor for all fields (a so-called *canonical* constructor)
  - `equals`, `hashCode` and `toString`

# Demo source code

```
git clone https://github.com/MichelTenVoorde/java17-preview.git
```

Learn by playing around, breaking things, and having fun. :)