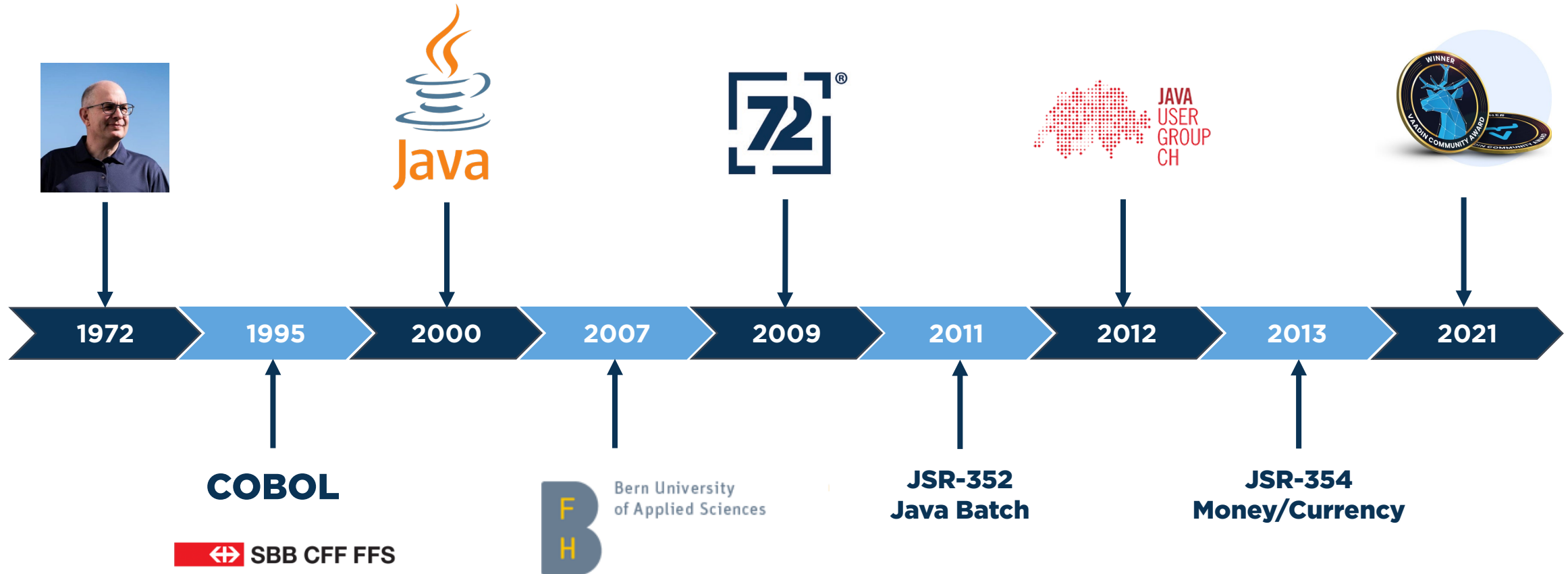




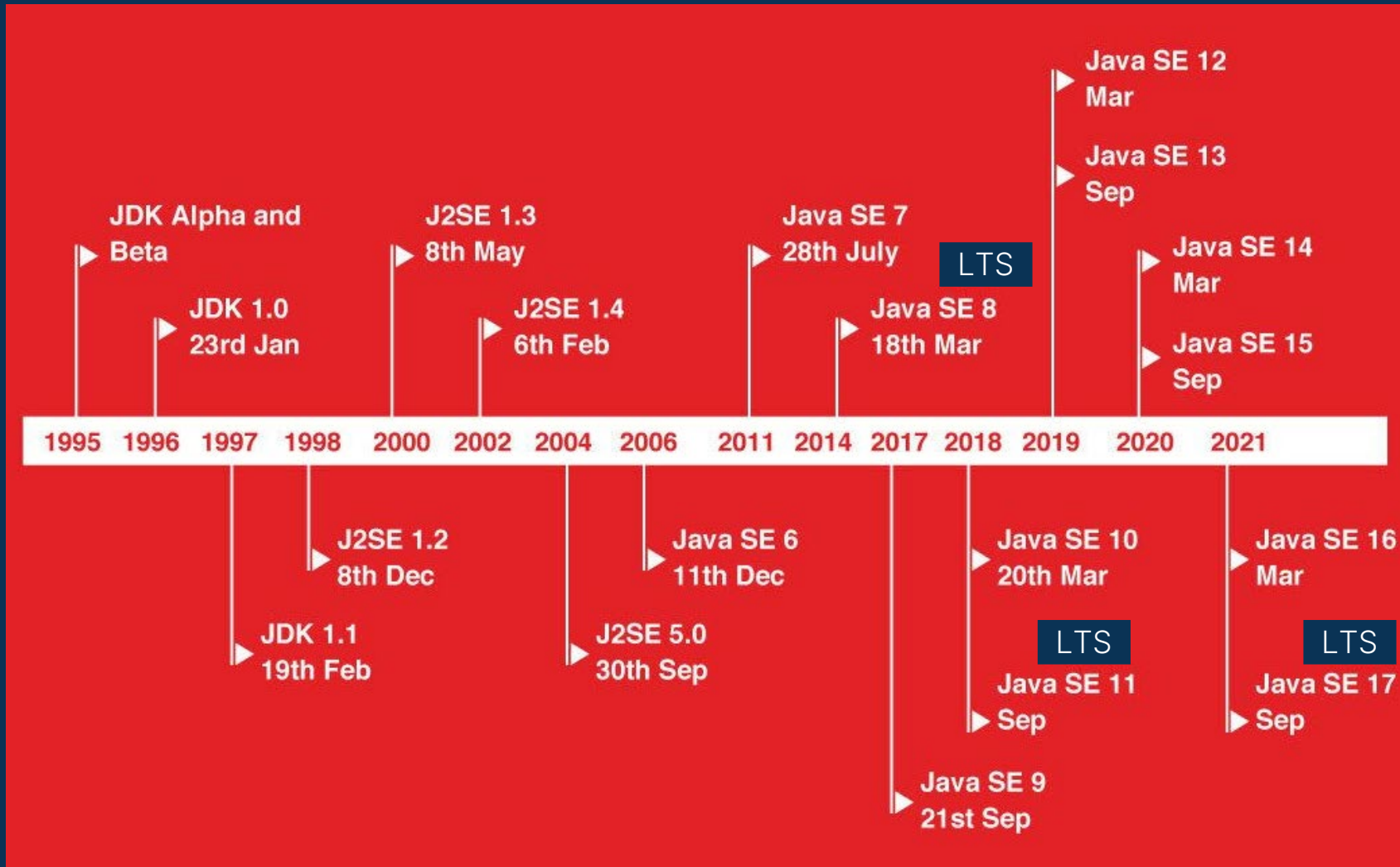
Simon Martinelli, 72 Services LLC



# About @simas\_ch



# Evolution of Java



# API Changes

- **The Java Version Almanac**  
<https://javaalmanac.io/>

# Maven

```
<properties>
  <!-- >= JDK 9 -->
  <maven.compiler.release>17</maven.compiler.release>
  <!-- < JDK 9 -->
  <maven.compiler.source>17</maven.compiler.source>
  <maven.compiler.target>17</maven.compiler.target>
</properties>
```

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <!-- overrides properties -->
    <source>17</source>
    <target>17</target>
    <release>17</release>

    <!-- To test preview features -->
    <compilerArgs>--enable-preview</compilerArgs>
  </configuration>
</plugin>
```

# Gradle

- Gradle  $\geq 7.3$  must be used for Java 17

```
compileJava {  
    options.release = 7  
}
```

# Text Blocks (Basics) (Preview 13, Standard 15)

```
// Text blocks start with a "" followed by optional whitespaces and a newline.  
// The simplest example looks like this:  
String sql1 = ""  
    select * from employee"";  
  
// Trailing whitespaces are removed  
String sql2 = ""  
    select *  
    from employee"";  
  
// \s (escape sequence) can be used to keep the blank after *  
String sql3 = ""  
    select * \s  
    from employee"";  
  
// Use \ to continue next line without line break  
String sql3 = ""  
    select * \  
    from employee"";
```

# Text Block (Indentation)

*// A text block differentiates **incidental** white space from **essential** white space*

```
String html1 = ""  
    <html>  
        <body>  
            <p>Hello World.</p>  
        </body>  
    </html>  
    "";
```

```
String html2 = ""  
    <html>  
        <body>  
            <p>Hello World.</p>  
        </body>  
    </html>  
    "";
```



# String Methods (Standard 15)

| Method          | From beginning | From end   | Per line |
|-----------------|----------------|------------|----------|
| strip()         | Leading        | Trailing   | No       |
| stripIndent()   | Incidental     | Incidental | Yes      |
| stripLeading()  | Leading        | n/a        | No       |
| stripTrailing() | n/a            | Trailing   | No       |

# String Methods (Standard 12)

```
// This method is equivalent to String.format(this, args). The advantage is that, as an instance method,  
// it can be chained off the end of a text block:
```

```
String formatted(Object... args)
```

```
String output = ""
```

```
    Name: %s
```

```
    Phone: %s
```

```
    Address: %s
```

```
    Salary: $%.2f
```

```
    """.formatted(name, phone, address, salary);
```

```
// Allows to transform a String with a Function
```

```
public <R> R transform(Function<? super String,? extends R> f)
```

```
String lower = "ABC".transform(s -> s.toLowerCase());
```

# Switch Expression (Preview 13, Standard 14)

```
// Switch Statement

int numLetters;

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;
    default:
        throw new IllegalStateException(
            "Invalid day: " + day);
}
```

```
// Switch Expression

int numLetters = switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> 6;
    case TUESDAY -> 7;
    case THURSDAY, SATURDAY -> 8;
    case WEDNESDAY -> 9;
    default -> throw new IllegalStateException(
        "Invalid day: " + day);
};
```

# Switch Expression yield

```
Day day = Day.WEDNESDAY;
int numLetters = switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> {
        System.out.println(6);
        yield 6;
    }
    case TUESDAY -> {
        System.out.println(7);
        yield 7;
    }
    case THURSDAY, SATURDAY -> {
        System.out.println(8);
        yield 8;
    }
    case WEDNESDAY -> {
        System.out.println(9);
        yield 9;
    }
    default -> throw new IllegalStateException("Invalid day: " + day);
};
```

# Switch Statement yield

```
Day day = Day.WEDNESDAY;
int numLetters = switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        System.out.println(6);
        yield 6;
    case TUESDAY:
        System.out.println(7);
        yield 7;
    case THURSDAY:
    case SATURDAY:
        System.out.println(8);
        yield 8;
    case WEDNESDAY:
        System.out.println(9);
        yield 9;
    default:
        throw new IllegalStateException("Invalid day: " + day);
};
```

# Records (Preview 14)

```
record Rectangle(double length, double width) {  
}
```

```
public final class Rectangle {  
    private final double length;  
    private final double width;  
  
    public Rectangle(double length, double width) {  
        this.length = length;  
        this.width = width;  
    }  
  
    double length() { return this.length; }  
    double width() { return this.width; }  
  
    public boolean equals...  
    public int hashCode...  
  
    public String toString() {...}  
}
```

# Records with Constructor

```
record Rectangle(double length, double width) {  
  
    public Rectangle {  
        if (length <= 0 || width <= 0) {  
            throw new java.lang.IllegalArgumentException(  
                String.format("Invalid dimensions: %f, %f", length, width));  
        }  
    }  
}
```

# Records Mutability

```
public record Book (String title, int numPages, List<String> chapters) {  
}
```

```
// chapters is a mutable collection!  
  
Book book = new Book("Breaking and entering", 289, chapters);  
  
System.out.println(book.title());  
System.out.println(book.toString());  
  
chapters.add("2");  
book.chapters().add("3");
```



# Records Immutable Collection

```
public record Book (String title, int numPages, List<String> chapters) {  
  
    public Book {  
        chapters = List.copyOf(chapters);  
    }  
}
```

# Sealed Classes (Preview 15, Standard 17)

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

public final class Circle extends Shape { ... }

public sealed class Rectangle extends Shape
    permits TransparentRectangle, FilledRectangle { ... }
public final class TransparentRectangle extends Rectangle { ... }
public final class FilledRectangle extends Rectangle { ... }

public final class Square extends Shape { ... }

public non-sealed class WeirdShape extends Shape { ... }
```

# Pattern Matching for instanceof

(Preview 14, Standard 16)

*// Without Pattern Matching*

```
if (shape instanceof Rectangle) {  
    Rectangle r = (Rectangle) shape;  
    return 2 * r.length() + 2 * r.width();  
} else if (shape instanceof Circle) {  
    Circle c = (Circle) shape;  
    return 2 * c.radius() * Math.PI;  
}
```

*// With Pattern Matching*

```
if (shape instanceof Rectangle r) {  
    return 2 * r.length() + 2 * r.width();  
} else if (shape instanceof Circle c) {  
    return 2 * c.radius() * Math.PI;  
}
```

# Pattern Matching Flow Scope

```
// Compiles  
if (num instanceof Double d1 && d1.intValue() % 2 == 0) {  
    System.out.println(d1.intValue());  
}  
  
// Does not compile! d2 might not be Double  
if (num instanceof Double d2 || d2.intValue() % 2 == 0) {  
    System.out.println(d2.intValue());  
}
```

# Pattern Matching for switch (Preview 17)

```
static double getPerimeter(Shape shape) throws IllegalArgumentException {  
    return switch (shape) {  
        case Rectangle r -> 2 * r.length() + 2 * r.width();  
        case Circle c    -> 2 * c.radius() * Math.PI;  
        default          -> throw new IllegalArgumentException("Unrecognized shape");  
    };  
}
```

```
// Selector Expression Type  
static void typeTester(Object obj) {  
    switch (obj) {  
        case null      -> System.out.println("null");  
        case String s  -> System.out.println("String");  
        case Color c   -> System.out.println("Color with " + c.values().length + " values");  
        case Point p   -> System.out.println("Record class: " + p.toString());  
        case int[] ia  -> System.out.println("Array of int values of length" + ia.length);  
        default        -> System.out.println("Something else");  
    }  
}
```

# Helpful NullPointerExceptions (14)

```
static void foo(String name) {  
    if (name.equals("Simon")) { // Line 52  
        // ...  
    }  
}
```

```
Exception in thread "main" java.lang.NullPointerException:  
    Cannot invoke "String.equals(Object)" because "name" is null  
        at io.seventytwo.edu.Examples.foo(Examples.java:52)  
        at io.seventytwo.edu.Examples.main(Examples.java:16)
```

# Compact Number Formatting

```
NumberFormat fmt = NumberFormat.getCompactNumberInstance(Locale.ENGLISH, NumberFormat.Style.SHORT);  
  
System.out.println(fmt.format(1000));  
System.out.println(fmt.format(100000));  
System.out.println(fmt.format(1000000));
```

```
// Output  
1K  
100K  
1M
```

# Day Period Support Added

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("B");

System.out.println(dtf.format(LocalTime.of(8, 0)));
System.out.println(dtf.format(LocalTime.of(13, 0)));
System.out.println(dtf.format(LocalTime.of(20, 0)));
System.out.println(dtf.format(LocalTime.of(23, 0)));
System.out.println(dtf.format(LocalTime.of(0, 0)));
```

```
// Output
in the morning
in the afternoon
in the evening
at night
midnight
```



# Stream.toList()

```
// Old  
List<String> stringList = Stream.of("a", "b", "c").collect(Collectors.toList());  
  
// New  
List<String> stringList = Stream.of("a", "b", "c").toList();
```

# Local Records, Enums, Interfaces

```
List<Merchant> findTopMerchants(List<Merchant> merchants, int month) {  
  
    // Local record  
    record MerchantSales(Merchant merchant, double sales) {}  
  
    return merchants.stream()  
        .map(merchant -> new MerchantSales(merchant, computeSales(merchant, month)))  
        .sorted((m1, m2) -> Double.compare(m2.sales(), m1.sales()))  
        .map(MerchantSales::merchant)  
        .collect(toList());  
}
```

# 72 Services GmbH

Tulpenweg 1  
2575 Täuffelen, Switzerland

+41 32 544 88 88  
info@72.services  
<https://72.services>

