Breaking Java Stereotypes: It's Not Your Dad's Language Anymore

Bazlur Rahman

Staff Software Developer

DNAStack

The Java Stereotypes

- Java often carries a reputation for being overly wordy and lacking flexibility.
- This reputation, while once a bit deserved, is completely outdated.

A Changed Language

- New language features for streamlined syntax
- Emphasis on data handling and expressiveness
- Performance and concurrency optimizations

The flexible main method (JEP 463)

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

    public static void initiateConferenceConjuring() {
            //magic happens here
    }
}
```

The flexible main method (Cont.)

```
public class Main {

    public static void main(String[] args) {
        initiateConferenceConjuring();
    }

    public static void initiateConferenceConjuring() {
           //magic happens here
    }
}
```

The flexible main method (Cont.)

```
public class Main {

    void main() {
        initiateConferenceConjuring();
    }

    public static void initiateConferenceConjuring() {
            //magic happens here
    }
}
```

The flexible main method (Cont.)

```java
void main() {
    initiateConferenceConjuring();
}

void initiateConferenceConjuring() {
    //magic happens here
}
```

Embracing Efficient Data Representation (JEP 395)

- Records – concise and immutable data holders.
- Automatic generation of accessors, equals(), hashCode(), toString().
- Perfect for modelling entities with a clear set of attributes.

Record In Action

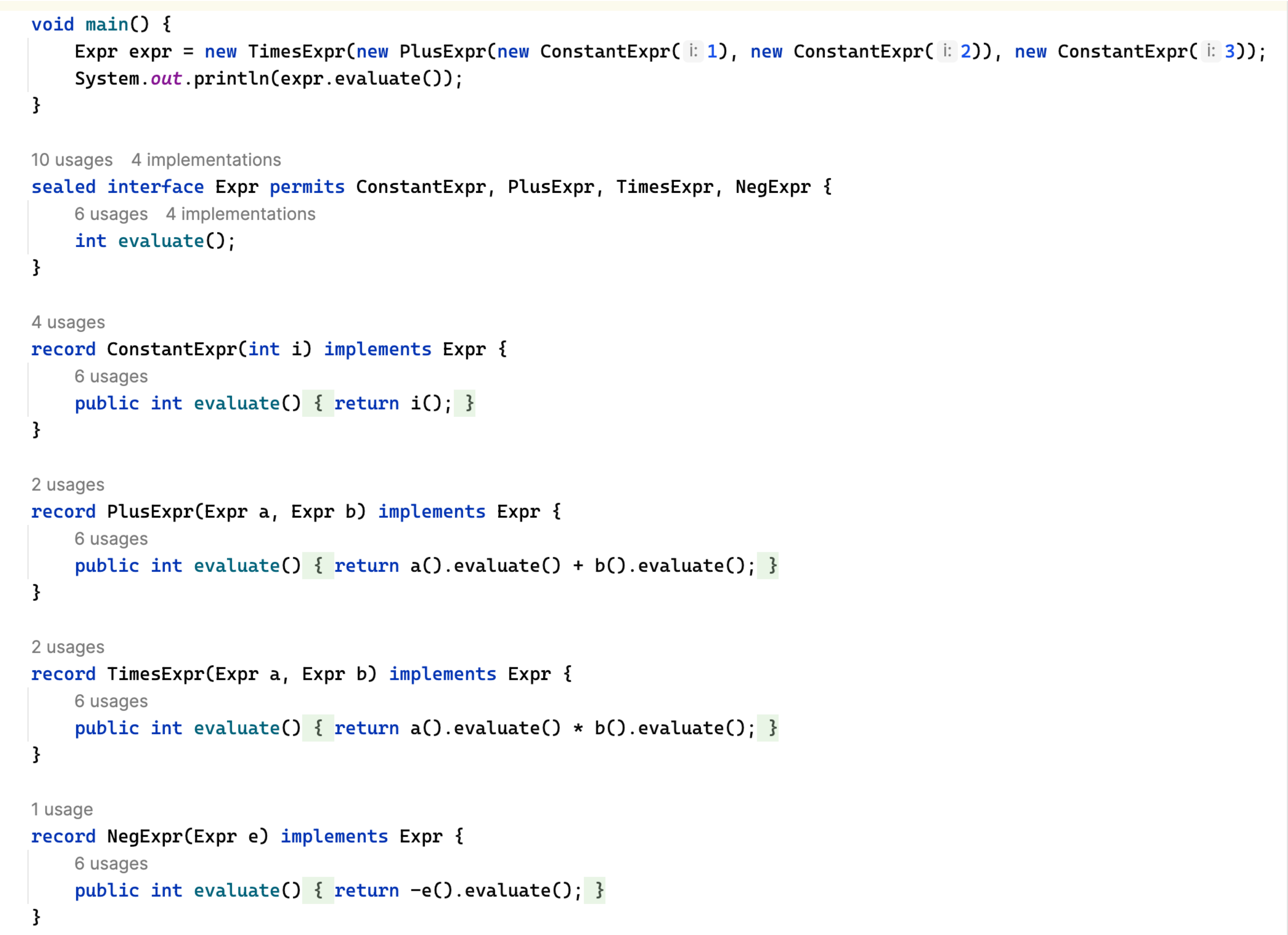
```java
public record Range(int lo, int hi) {
   public Range {
      if (lo > hi)
         throw new IllegalArgumentException(String.format("Invalid range: lower limit (%d) is greater than upper limit (%d).", lo, hi));
   }
}
```

Characteristics of Records

- Records are implicitly final
- Designed to be a transparent carrier for immutable data
- Custom constructors must delegate to the canonical constructor
- Ideal for passing data in a type-safe manner
- Simplifies the creation of DTOs for APIs
- Encourages the use of immutable data structures

Controlled Inheritance with Sealed Classes (JEP 409)

- Sealing restricts which classes can extend a parent class.
- permits clause explicitly lists those allowed subclasses.
- Brings predictability and maintainability to class hierarchies.



```java
void main() {
    Expr expr = new TimesExpr(new PlusExpr(new ConstantExpr( i: 1), new ConstantExpr( i: 2)), new ConstantExpr( i: 3));
    System.out.println(expr.evaluate());
}

10 usages   4 implementations
sealed interface Expr permits ConstantExpr, PlusExpr, TimesExpr, NegExpr {
    6 usages   4 implementations
    int evaluate();
}

4 usages
record ConstantExpr(int i) implements Expr {
    6 usages
    public int evaluate() { return i(); }
}

2 usages
record PlusExpr(Expr a, Expr b) implements Expr {
    6 usages
    public int evaluate() { return a().evaluate() + b().evaluate(); }
}

2 usages
record TimesExpr(Expr a, Expr b) implements Expr {
    6 usages
    public int evaluate() { return a().evaluate() * b().evaluate(); }
}

1 usage
record NegExpr(Expr e) implements Expr {
    6 usages
    public int evaluate() { return -e().evaluate(); }
}
```

Records (Product Types) + Sealed Class (Sum types) = Algebraic data types

Model complex data clearly: They directly represent the structure of your data.

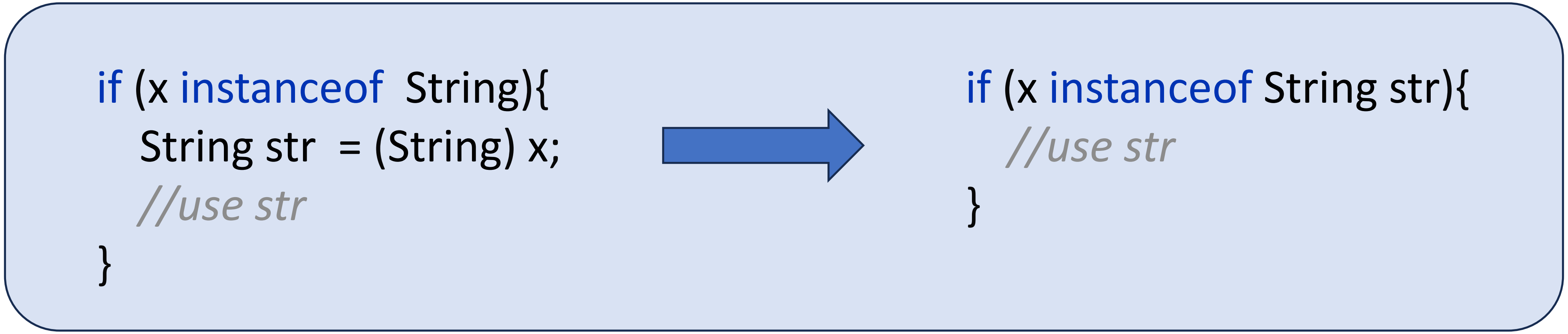
Prevent errors: ADTs can make it impossible to create invalid data states.

Pattern matching: Write code that elegantly handles each case of a sum type.

https://www.infoq.com/articles/data-oriented-programming-java/

Pattern Matching

- Simplify common programming patterns, enhancing code readability and safety.
- Works in conditional contexts such as instanceof and switch



```
if (x instanceof  String){
    String str  = (String) x;
    //use str
}
```

```
if (x instanceof String str){
    //use str
}
```

```
if (x instanceof String str && str.length() > 3){
   //use str
}else {
   // you do something else.
}
```

```
switch (x) {
    case String str when str.length() > 3 -> {
        //use str
    }
    case Integer n when n < 0 -> {
        System.out.println("value is zero or lower");
    }
    default -> {
        // you do something else.
    }
}
```

```java
Object value = 42;

var message = switch (value) {
    case null -> "The value is `null`";
    case String str -> STR."Is String: \{str}";
    case Integer n -> STR."is an integer: \{n}";
    case Number n -> STR."Is a Number: \{n}";
    case int[] intArray -> STR."Is an array of number: \{intArray}";
    case List list -> STR."Is a list of some type: \{list}";
    case Wrapper(var v) -> STR."Wrapped value: \{v}";
    default -> STR."Is untested type =(: \{value.toString()}";
};

record Wrapper<T>(T t) {}
```

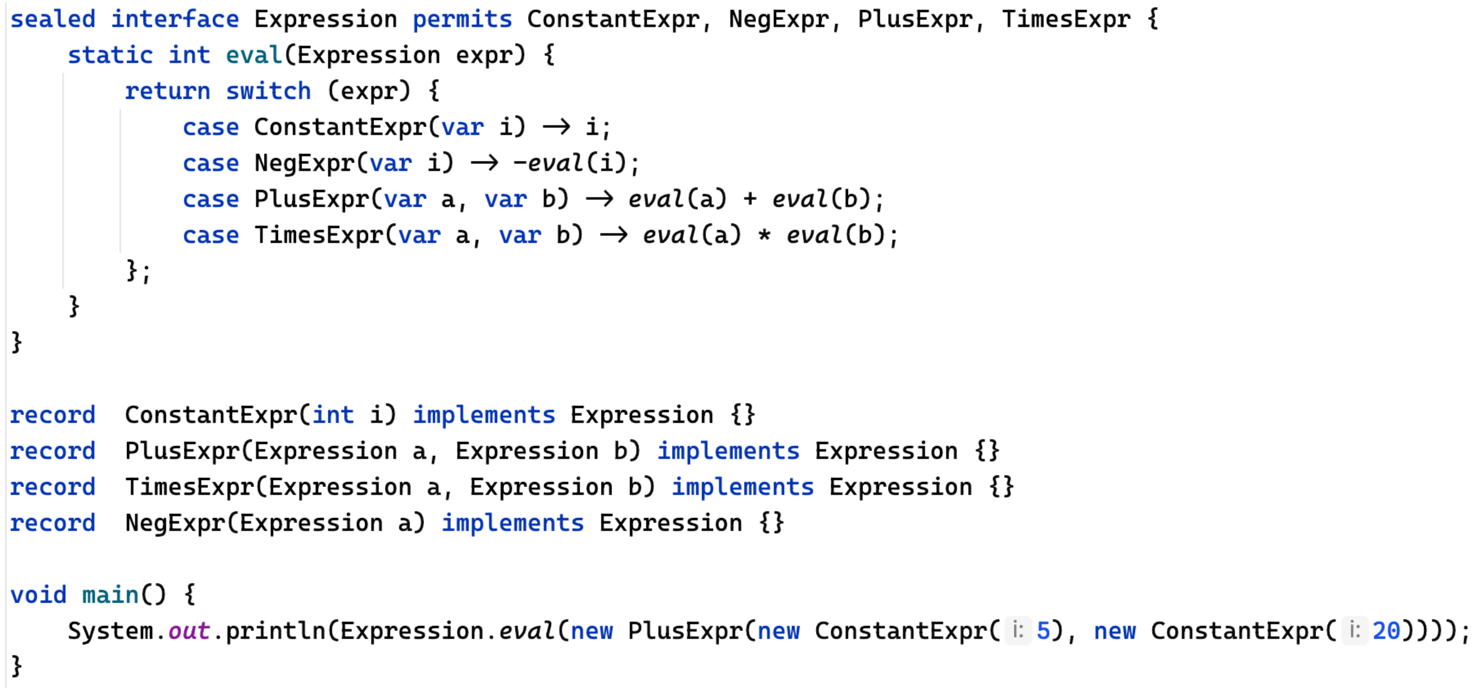
Let’s combine (sealed class + pattern matching)

```
sealed interface Option<T> permits Some, None{ }
record Some<T>(T value) implements Option<T> {}
record None<T>() implements Option<T> {}
```

```
String getOptionValue(Option<String> str) {
    return switch (str) {
        case None<String> _ -> "";
        case Some<String>(var value) -> "the value is %s".formatted(value);
    };
}
```

Traversing algebraic data types

Records, sealed types, and pattern matching are designed to work together.



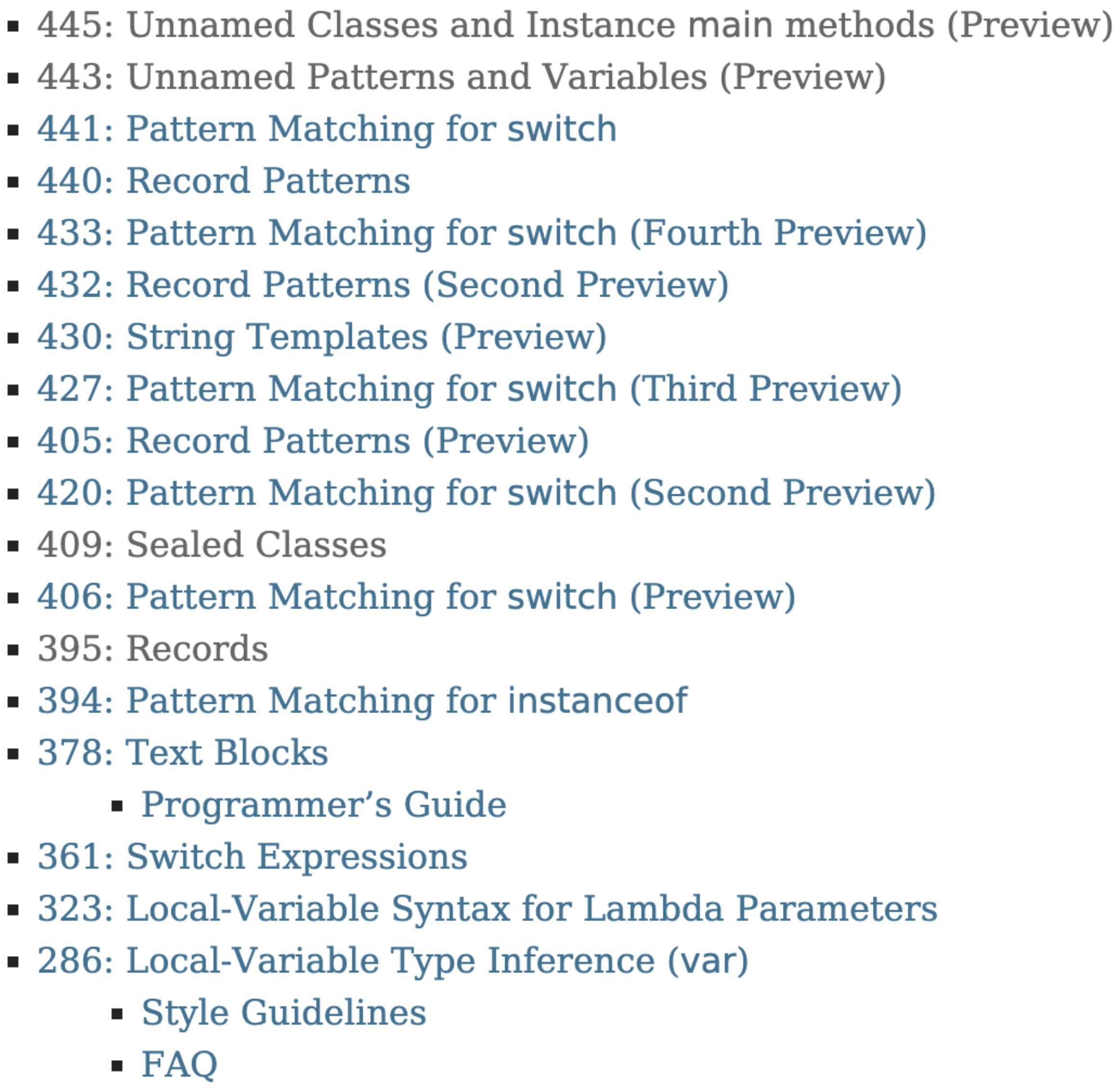
```
sealed interface Expression permits ConstantExpr, NegExpr, PlusExpr, TimesExpr {
    static int eval(Expression expr) {
        return switch (expr) {
            case ConstantExpr(var i) -> i;
            case NegExpr(var i) -> -eval(i);
            case PlusExpr(var a, var b) -> eval(a) + eval(b);
            case TimesExpr(var a, var b) -> eval(a) * eval(b);
        };
    }
}

record  ConstantExpr(int i) implements Expression {}
record  PlusExpr(Expression a, Expression b) implements Expression {}
record  TimesExpr(Expression a, Expression b) implements Expression {}
record  NegExpr(Expression a) implements Expression {}

void main() {
    System.out.println(Expression.eval(new PlusExpr(new ConstantExpr( i: 5), new ConstantExpr( i: 20))));
}
```

It took 19 lines of code, which would have taken 60 lines using traditional Java.

https://www.infoq.com/articles/data-oriented-programming-java/

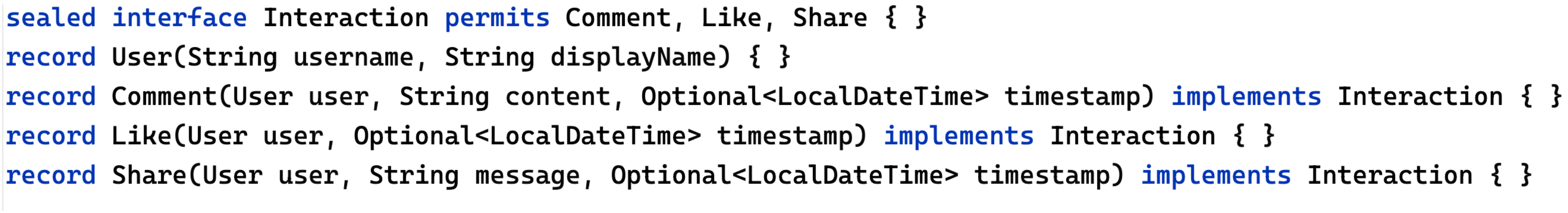
Project Amber

https://openjdk.org/projects/amber/

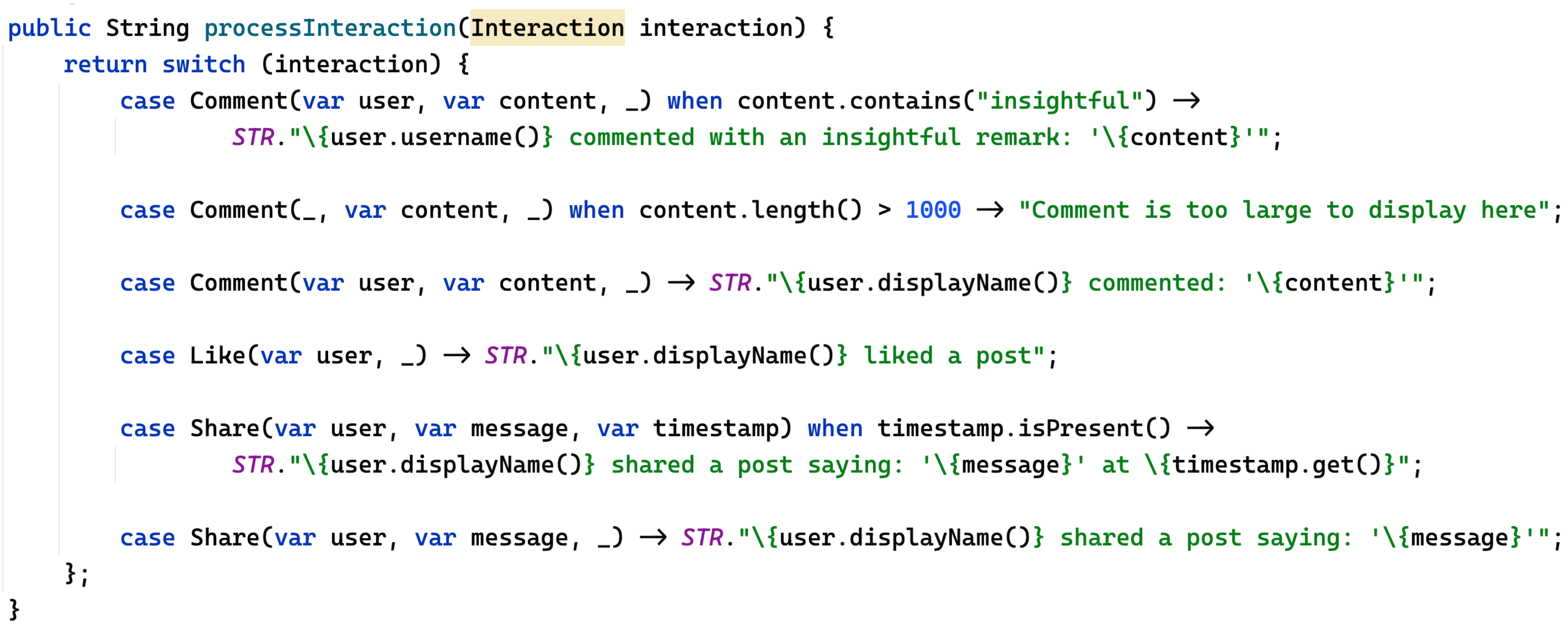


- 445: Unnamed Classes and Instance main methods (Preview)
- 443: Unnamed Patterns and Variables (Preview)
- 441: Pattern Matching for switch
- 440: Record Patterns
- 433: Pattern Matching for switch (Fourth Preview)
- 432: Record Patterns (Second Preview)
- 430: String Templates (Preview)
- 427: Pattern Matching for switch (Third Preview)
- 405: Record Patterns (Preview)
- 420: Pattern Matching for switch (Second Preview)
- 409: Sealed Classes
- 406: Pattern Matching for switch (Preview)
- 395: Records
- 394: Pattern Matching for instanceof
- 378: Text Blocks
  - Programmer’s Guide
- 361: Switch Expressions
- 323: Local-Variable Syntax for Lambda Parameters
- 286: Local-Variable Type Inference (var)
  - Style Guidelines
  - FAQ

Unnamed Variable and Patterns



```java
sealed interface Interaction permits Comment, Like, Share { }
record User(String username, String displayName) { }
record Comment(User user, String content, Optional<LocalDateTime> timestamp) implements Interaction { }
record Like(User user, Optional<LocalDateTime> timestamp) implements Interaction { }
record Share(User user, String message, Optional<LocalDateTime> timestamp) implements Interaction { }
```



```
public String processInteraction(Interaction interaction) {
    return switch (interaction) {
        case Comment(var user, var content, _) when content.contains("insightful") ->
                STR."\{user.username()} commented with an insightful remark: '\{content}'";

        case Comment(_, var content, _) when content.length() > 1000 -> "Comment is too large to display here";

        case Comment(var user, var content, _) -> STR."\{user.displayName()} commented: '\{content}'";

        case Like(var user, _) -> STR."\{user.displayName()} liked a post";

        case Share(var user, var message, var timestamp) when timestamp.isPresent() ->
                STR."\{user.displayName()} shared a post saying: '\{message}' at \{timestamp.get()}";

        case Share(var user, var message, _) -> STR."\{user.displayName()} shared a post saying: '\{message}'";
    };
}
```

StringTemplate (JEP 459)

- The most common request in Java since the beginning
- New kind of expression specifically for manipulating text & structured data.
- Require a template processor ('interpreter') to generate a result.
- Not limited to just strings - they can generate various output types.

The STR Processor

- Template expressions handle interpolation, data sanitization, and code streamlining.
- Expressions enclosed in familiar curly braces {} enhance readability and reduce errors.
- No need for manual string concatenations and repeated type conversions.

```
var name = "Bazlur Rahman";
var info = STR."My name is \{name}";
System.out.println(info);
```

STR: Security Built-In

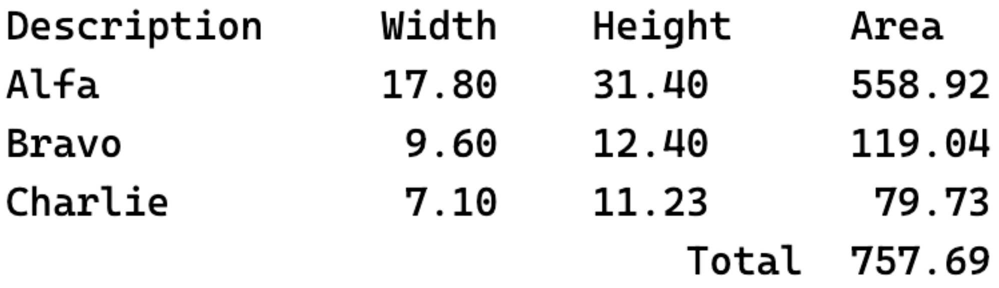
- **Reduces injection risks:** Focus on SQL injection as the most common use case, but mention STR's help in other areas (XSS, HTML issues, etc.).
- **Automatic escaping/validation:** Emphasize that users aren't expected to implement these manually - STR handles the heavy security work.

FMT Processor – Structured Formatting

- Leverages the core string interpolation and safety features of the STR template processor.
- Employs familiar formatting patterns from **`java.util.Formatter`** (e.g., %7.2f, %-12s) for precise control over numerical formatting and alignment.
- Excels in scenarios where tabular data representation, well-aligned reports, or formatted logging messages are required.

```
record Rectangle(String name, double width, double height) {
    double area() {
        return width * height;
    }
}
```

```
String table = FMT."""
    Description     Width    Height     Area
    %-12s\{zone[0].name}  %7.2f\{zone[0].width}  %7.2f\{zone[0].height}     %7.2f\{zone[0].area()}
    %-12s\{zone[1].name}  %7.2f\{zone[1].width}  %7.2f\{zone[1].height}     %7.2f\{zone[1].area()}
    %-12s\{zone[2].name}  %7.2f\{zone[2].width}  %7.2f\{zone[2].height}     %7.2f\{zone[2].area()}
    \{" ".repeat(28)} Total %7.2f\{zone[0].area() + zone[1].area() + zone[2].area()}
    """;
```



```
Description     Width    Height     Area
Alfa            17.80    31.40      558.92
Bravo            9.60    12.40      119.04
Charlie          7.10    11.23       79.73
                             Total  757.69
```

Beyond String

```
var JSON = StringTemplate.Processor.of((StringTemplate st) -> {
    var json = new JSONObject();
    var valueIterator = st.values().iterator();
    for (String string : st.fragments()) {
        String key = string.trim();
        if (!key.isEmpty() && valueIterator.hasNext()) {
            Object value = valueIterator.next();
            json.put(key, value);
        }
    }
    return json;
});

String language = "Java";
int version = 21;

JSONObject jsonObject = JSON."language: \{language}, version: \{version}";
System.out.println(jsonObject);

//{"language:":"Java",", version:":21}
```

JEP 447: Refining Java Constructors for Enhanced Flexibility

```
class PositiveBigInteger extends BigInteger {
    public PositiveBigInteger(long value) {
        super(String.valueOf(value));  // Potentially unnecessary work
        if (value <= 0)
            throw new IllegalArgumentException("non-positive value");
    }
}
```

```
class PositiveBigInteger extends BigInteger {
    public PositiveBigInteger(long value) {
        if (value <= 0)
            throw new IllegalArgumentException("non-positive value");
        super(String.valueOf(value));
    }
}
```

Vector API: Optimizing Java for Modern Hardware

- Introduced in Java 16 as an incubator API, the Vector API enables reliable, cross-platform vector computations that leverage hardware-specific Single Instruction/Multiple Data (SIMD) capabilities.
- The Vector API continuously evolves to be more performant, adaptable, and expressive.
- Future Vector API iterations will work in tandem with Project Valhalla's value classes, leading to increased performance and reduced memory overhead.
- The API provides capabilities for lane-wise (element by element) and cross-lane (whole vector at once) operations, including arithmetic, logic, and bitwise manipulation.

```
public void scalarAddition(int[] a, int[] b, int[] result) {
    for (int i = 0; i < a.length; i++) {
        result[i] = a[i] + b[i];
    }
}
```

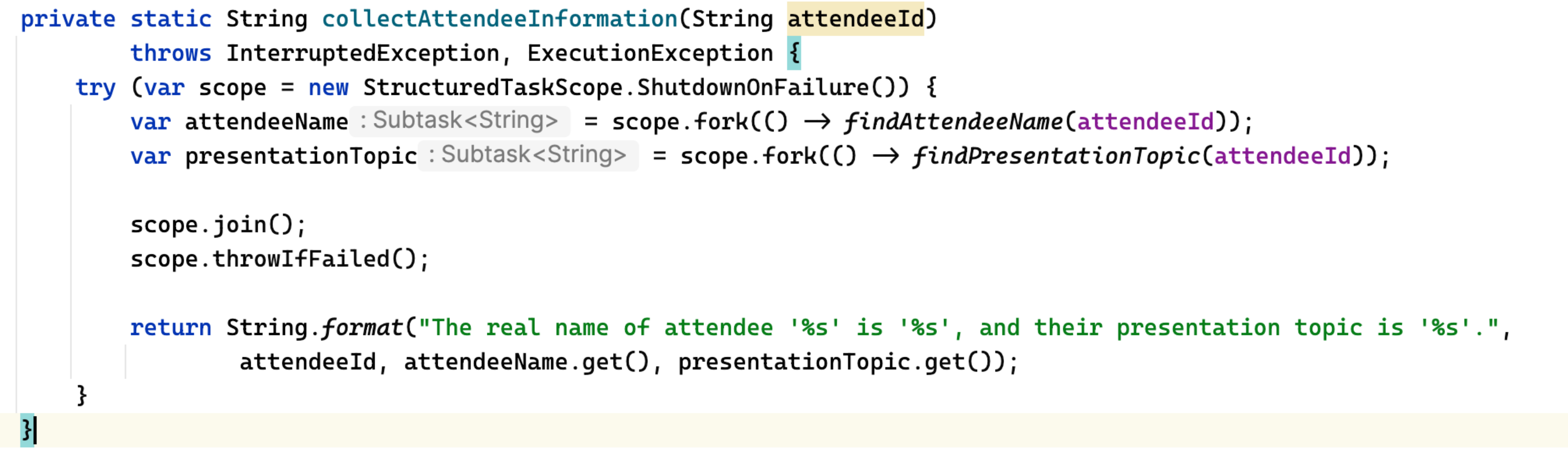
```
public void vectorAddition(int[] a, int[] b, int[] result) {
    final VectorSpecies<Integer> species = IntVector.SPECIES_PREFERRED;
    int length = species.loopBound(a.length);
    for (int i = 0; i < length; i += species.length()) {
        IntVector va = IntVector.fromArray(species, a, i);
        IntVector vb = IntVector.fromArray(species, b, i);
        IntVector vc = va.add(vb);
        vc.intoArray(result, i);
    }
    // Handle remaining elements
    for (int i = length; i < a.length; i++) {
        result[i] = a[i] + b[i];
    }
}
```

Virtual Threads

```
void main() throws InterruptedException {
    Thread vThread = Thread.ofVirtual().start(() -> {
        System.out.println("Hello ConFoo!!!");
        System.out.println(STR."Running inside a virtual
thread\{Thread.currentThread()}");
    });

    vThread.join();
}
```

Structured Concurrency

- Simplify how Java developers manage groups of related tasks concurrently.
- Encourages treating the whole group of tasks as a single unit for error handling, cancellation, and observability.



```java
private static String collectAttendeeInformation(String attendeeId)
        throws InterruptedException, ExecutionException {
    try (var scope = new StructuredTaskScope.ShutdownOnFailure()) {
        var attendeeName : Subtask<String> = scope.fork(() -> findAttendeeName(attendeeId));
        var presentationTopic : Subtask<String> = scope.fork(() -> findPresentationTopic(attendeeId));

        scope.join();
        scope.throwIfFailed();

        return String.format("The real name of attendee '%s' is '%s', and their presentation topic is '%s'.",
                attendeeId, attendeeName.get(), presentationTopic.get());
    }
}
```

Foreign Function Interface (FFI)

- **Improved Native Interoperability:** The FFM API simplifies how Java interacts with code and data outside the JVM (e.g., native libraries written in C, C++).
- **Replaces JNI:** Offers a more developer-friendly and safer alternative to the cumbersome and error-prone Java Native Interface (JNI).
- **Efficiency & Safety:** Promotes a Java-idiomatic style, enhancing performance and security when working with native code.

```java
import java.lang.foreign.*;
import java.lang.invoke.MethodHandle;
import java.util.Arrays;

public class RadixSortExample {
    public static void main(String[] args) {
        RadixSortExample radixSorter = new RadixSortExample();
        String[] javaStrings = {"mouse", "cat", "dog", "car"};

        System.out.println(STR."radixsort input: \{Arrays.toString(javaStrings)}");

        // Perform radix sort on input array of strings
        javaStrings = radixSorter.sort(javaStrings);

        System.out.println(STR."radixsort output: \{Arrays.toString(javaStrings)}");
    }

    private String[] sort(String[] strings) {
        // Find foreign function on the C library path
        Linker linker = Linker.nativeLinker();
        SymbolLookup stdlib = linker.defaultLookup();
        MemorySegment radixSort = stdlib.find("radixsort").orElseThrow();
        MethodHandle methodHandle = linker.downcallHandle(radixSort, FunctionDescriptor.ofVoid(
                ValueLayout.ADDRESS, ValueLayout.JAVA_INT, ValueLayout.ADDRESS, ValueLayout.JAVA_CHAR
        ));

        // Use try-with-resources to manage the lifetime of off-heap memory
        try (Arena arena = Arena.ofConfined()) {
            // Allocate a region of off-heap memory to store pointers
            MemorySegment pointers = arena.allocate(ValueLayout.ADDRESS, strings.length);

            // Copy the strings from on-heap to off-heap
            for (int i = 0; i < strings.length; i++) {
                MemorySegment cString = arena.allocateFrom(strings[i]);
                pointers.setAtIndex(ValueLayout.ADDRESS, i, cString);
            }

            // Sort the off-heap data by calling the foreign function
            methodHandle.invoke(pointers, strings.length, MemorySegment.NULL, '\0');

            // Copy the (reordered) strings from off-heap to on-heap
            for (int i = 0; i < strings.length; i++) {
                MemorySegment cString = pointers.getAtIndex(ValueLayout.ADDRESS, i);
                cString = cString.reinterpret(Long.MAX_VALUE);
                strings[i] = cString.getString(0);
            }
        } catch (Throwable e) {
            throw new RuntimeException(e);
        }

        return strings;
    }
}
```

Tools: Launch Multi-File Source-Code Programs

- Java streamlines development by supporting direct execution of multi-file source code programs.
- Small and early-stage projects benefit from simplified setup and faster iteration.
- Focus on coding without the immediate need to configure compilers or build systems.

```
import org.json.JSONObject;

public class Hello {
    void main() {
//        System.out.println("ConFoo!" +
//                " or should we say JConFoo");
//        System.out.println(Greeting.say() + " JConFoo!");

        System.out.println(STR."\{Greeting.say()} ConFoo!\{new JSONObject().put("hello", "world").toString()}");
    }
}
```

```
java --enable-preview --source 23 -cp libs/* src/Hello.java
```

About Me

DNASTACK

-Staff Software Developer

-Java Champion

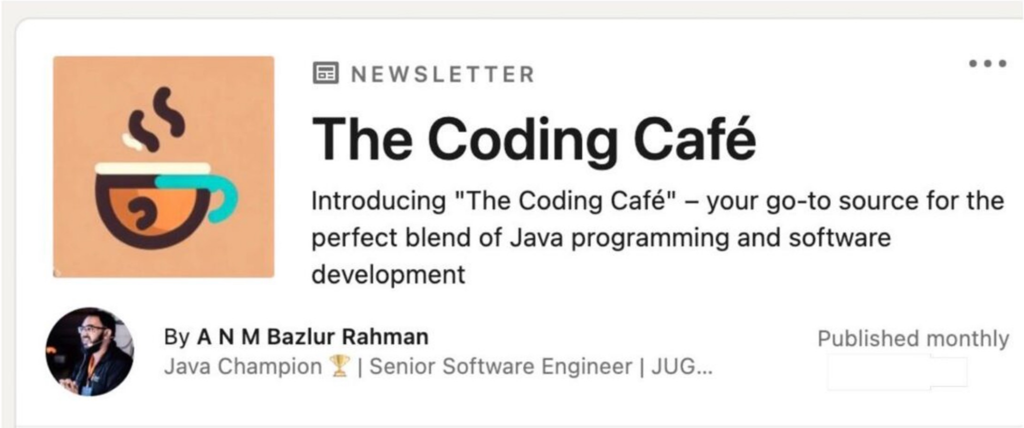
-Jakarta EE Ambassador

-JUG Leader

-Published Author

-InfoQ Editor of Java Queue

-Editor of Foojay.io



NEWSLETTER

The Coding Café

Introducing "The Coding Café" – your go-to source for the perfect blend of Java programming and software development

By A N M Bazlur Rahman
Java Champion 🏆 | Senior Software Engineer | JUG...

Published monthly

https://twitter.com/bazlur_rahman
https://www.linkedin.com/in/bazlur/
https://foojay.io/today/author/bazlur-rahman/
https://bazlur.ca

Thank you