

Records and Sealed Types

Coming Soon to a JVM Near You!

Ben Evans (He / Him)

Safe Harbor

This presentation and the information herein (including any information that may be incorporated by reference) is provided for informational purposes only and should not be construed as an offer, commitment, promise or obligation on behalf of New Relic, Inc. ("New Relic") to sell securities or deliver any product, material, code, functionality, or other feature. Any information provided hereby is proprietary to New Relic and may not be replicated or disclosed without New Relic's express written permission.

Such information may contain forward-looking statements within the meaning of federal securities laws. Any statement that is not a historical fact or refers to expectations, projections, future plans, objectives, estimates, goals, or other characterizations of future events is a forward-looking statement. These forward-looking statements can often be identified as such because the context of the statement will include words such as "believes," "anticipates," "expects" or words of similar import.

Actual results may differ materially from those expressed in these forward-looking statements, which speak only as of the date hereof, and are subject to change at any time without notice. Existing and prospective investors, customers and other third parties transacting business with New Relic are cautioned not to place undue reliance on this forward-looking information. The achievement or success of the matters covered by such forward-looking statements are based on New Relic's current assumptions, expectations, and beliefs and are subject to substantial risks, uncertainties, assumptions, and changes in circumstances that may cause the actual results, performance, or achievements to differ materially from those expressed or implied in any forward-looking statement. Further information on factors that could affect such forward-looking statements is included in the filings New Relic makes with the SEC from time to time. Copies of these documents may be obtained by visiting New Relic's Investor Relations website at ir.newrelic.com or the SEC's website at www.sec.gov.

New Relic assumes no obligation and does not intend to update these forward-looking statements, except as required by law. New Relic makes no warranties, expressed or implied, in this presentation or otherwise, with respect to the information provided.

About Me – Career

- New Relic, Principal Engineer
- jClarity, Co-founder
 - Sold to Microsoft
- Deutsche Bank
 - Chief Architect (Listed Derivatives)
- Morgan Stanley
 - Google IPO
- Sporting Bet
 - Chief Architect



jClarity



About Me – Community

- Java Champion
- JavaOne Rock Star Speaker
- Java Community Process Executive Committee
- London Java Community
 - Organising Team
 - Co-founder, AdoptOpenJDK



Enums

Enums

Enums

Enums

Enums

Enums

Why Enums?

- General Principle
 - Patterns in one language become features in the next

Why Enums?

- General Principle
 - Patterns in one language become features in the next
 - Vtables in C ==> `virtual` in C++
 - Iterator pattern in C++ ==> `Iterator` in Java
 - Enum in C++ ==> `Typesafe enum` in Java

Why Enums?

- General Principle
 - Patterns in one language become features in the next
 - Vtables in C ==> `virtual` in C++
 - Iterator pattern in C++ ==> `Iterator` in Java
 - Enum in C++ ==> Typesafe enum in Java
- Specific Case
 - Java enum is a restricted form of `class`
 - With semantics defined by a pattern (“Finitely Many Instances”)
 - With compact syntax that stems from the pattern

DEMO

Project Amber

“... explore and incubate smaller, productivity-oriented Java language features ...”

– Goal of Project Amber

Why Records?

- First-class support for modeling data-only aggregates
 - “The state, the whole state and nothing but the state”
- Close a possible gap in Java's type system
- Language-level syntax for a common pattern
- Reduce class boilerplate

Wadler's Law

“Emotional intensity of debate on a language feature increases as one moves down the following scale: Semantics, Syntax, Lexical syntax, Comments.”

– Philip Wadler

Boilerplate

- `toString()`
- `hashCode()` and `equals()`
- Getter methods
- Public constructor

A Java Cashflow Class

```
public final class Cashflow {  
    private final double amount;  
    private final String currency;  
    private final LocalDateTime due;  
  
    public Cashflow(String currency, double amount, LocalDateTime due) {  
        this.amount = amount;  
        this.currency = currency;  
        this.due = due;  
    }  
  
    public double amount() {  
        return amount;  
    }  
  
    public String currency() {  
        return currency;  
    }  
  
    public LocalDateTime due() {  
        return due;  
    }  
}
```



A Java 14 Cashflow Record

```
public final class Cashflow {  
    record Cashflow(double amount, String currency, LocalDateTime due) { }  
    private final double amount;  
    private final String currency;  
    private final LocalDateTime due;  
  
    public Cashflow(String currency, double amount, LocalDateTime due) {  
        this.amount = amount;  
        this.currency = currency;  
        this.due = due;  
    }  
  
    public double amount() {  
        return amount;  
    }  
  
    public String currency() {  
        return currency;  
    }  
  
    public LocalDateTime due() {  
        return due;  
    }  
}
```


What Could Records Be?

- Boilerplate reduction of POJOs
- Java Beans 2.0
- Named Tuples
- Product Types (a form of *Algebraic Data Type*)

Records Are Named Tuples

- Nominal Typing
 - Not Structural Typing
- Emphasize the semantics above everything else
- Records have benefits over Plain Old Tuples
 - Compact constructors

DEMO

Equals Invariant

- If a record `R` has components `c1`, `c2`, ... `cn`
 - Is copied as follows:
- `R r2 = new R(r.c1(), r.c2(), .., r.cn());`
 - Then `r.equals(r2)` must be true
- Usual `equals()` - `hashCode()` contract also applies

Record Serialization

“Serialization constitutes an invisible but public constructor, and an invisible but public set of accessors for your internal state.”

– Brian Goetz

Interlude: Java Switch Expressions

```
import java.time.DayOfWeek;

public class Days {
    public static boolean isWorkDay(DayOfWeek day) {
        var today = switch(day) {
            case SATURDAY, SUNDAY -> false;
            case MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY -> true;
        };

        // Do any further processing for e.g. public holidays...

        return today;
    }
}
```

Why Sealed Types?

- Recall enums...
 - Enums are instances of a class
 - Enums are exhaustive
- How can we say: “A Pet IS-A Cat or Dog”?
- How can we model many different Cats and Dogs?

Option 1 - The State Field

- Single implementing Pet class
 - Enum field that holds the “real” type
- Ugly and messy
 - Programmer must keep track of “type bits”
 - No type-specific functionality - “Cat can’t `purr ( )`”

Option 2 - Abstract Base

- Use an abstract base with package-private ctor
 - Concrete subclasses
- Leaky abstraction
 - Outside of JDK packages no protection of packages
 - Maybe modules help?
 - But what about API packages?
 - And reflection?

Solution - New OO construct

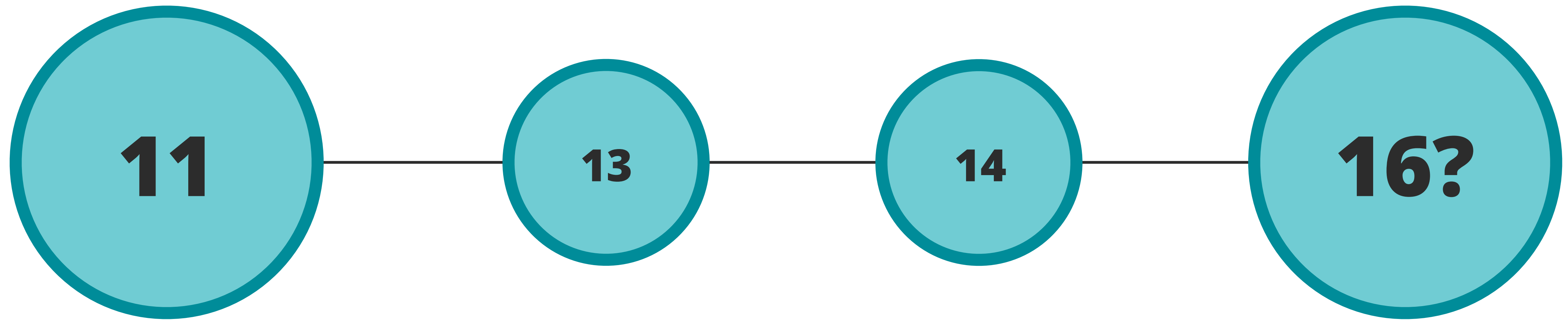
- Sealed types
 - Can be extended by a known list of types
 - But no others
 - “almost final” classes
- New restricted keywords
 - `sealed`
 - `permits`
- Known as “union types” in some languages

Java 11 Nestmates



DEMO

The Path So Far...



Nestmates

Inner classes done right

Switch Expressions

Second preview

Records

Preview of Records

Sealed Types

At some point
in the future

Conclusion

- Records are about semantics
 - Restricted classes that implement a pattern
 - Not a general replacement, just very useful
- Sealed Types are about a new OO construct
- They are part of a journey towards pattern matching
 - Made up of smaller language parts - incremental delivery
 - Algebraic Data Types
- Hopefully everything will be final by Java 17

Further Reading

- <https://www.infoq.com/articles/java-14-feature-spotlight/>
- <https://blogs.oracle.com/javamagazine/records-come-to-java>
- <https://blogs.oracle.com/javamagazine/inside-the-language-sealed-types>

What Can You Do To Help?

- Try out the Java 14 / 15 betas (and the new features)
- Keep an eye out for
 - Sealed Types betas
 - Deconstruction patterns
- Write code using the new features
 - Even just as research / innovation time
- Give feedback!
 - Sooner is better

THANK YOU & QUESTIONS?



bevans@newrelic.com