

# Refactoring to Java 17 and beyond

Jeanne Boyarsky  
August 8, 2022  
KCDC

[speakerdeck.com/boyarsky](https://speakerdeck.com/boyarsky)



## Jeanne Boyarsky

### Java 17 Cert Book Author



Jeanne Boyarsky is a Java Champion from New York City and has been a Java developer for more than 20 years. She has co-authored Wiley's Oracle Java 8/11/17 certification books. Jeanne also serves as her team's Scrum Master. She volunteers at CodeRanch and mentors the programmers on a high school robotics team in her free time. Jeanne has spoken at numerous conferences including Dev Nexus, KCDC, QCon, and Oracle Code One.

- **Refactoring Lab: To Java 17 and beyond**
- **Refactoring to Java 17 and beyond**

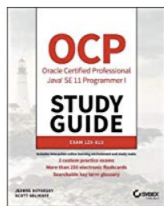
# Pause for a Commercial

## Amazon Best Sellers

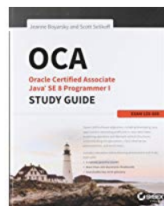
Our most popular products based on sales. Updated hourly.

### Best Sellers in Oracle Certification

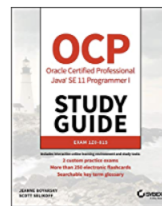
#1



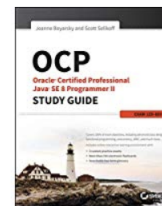
#2



#3



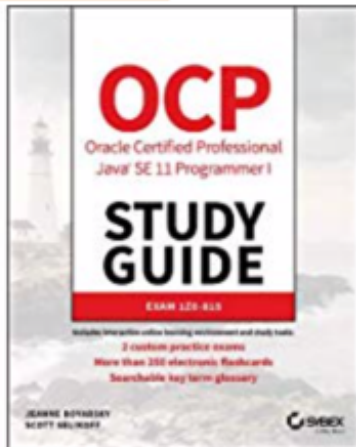
#4



#5



#1 New Release



## Java certs: 8/11/17

## Book giveaway at end!

@jeanneboyarsky



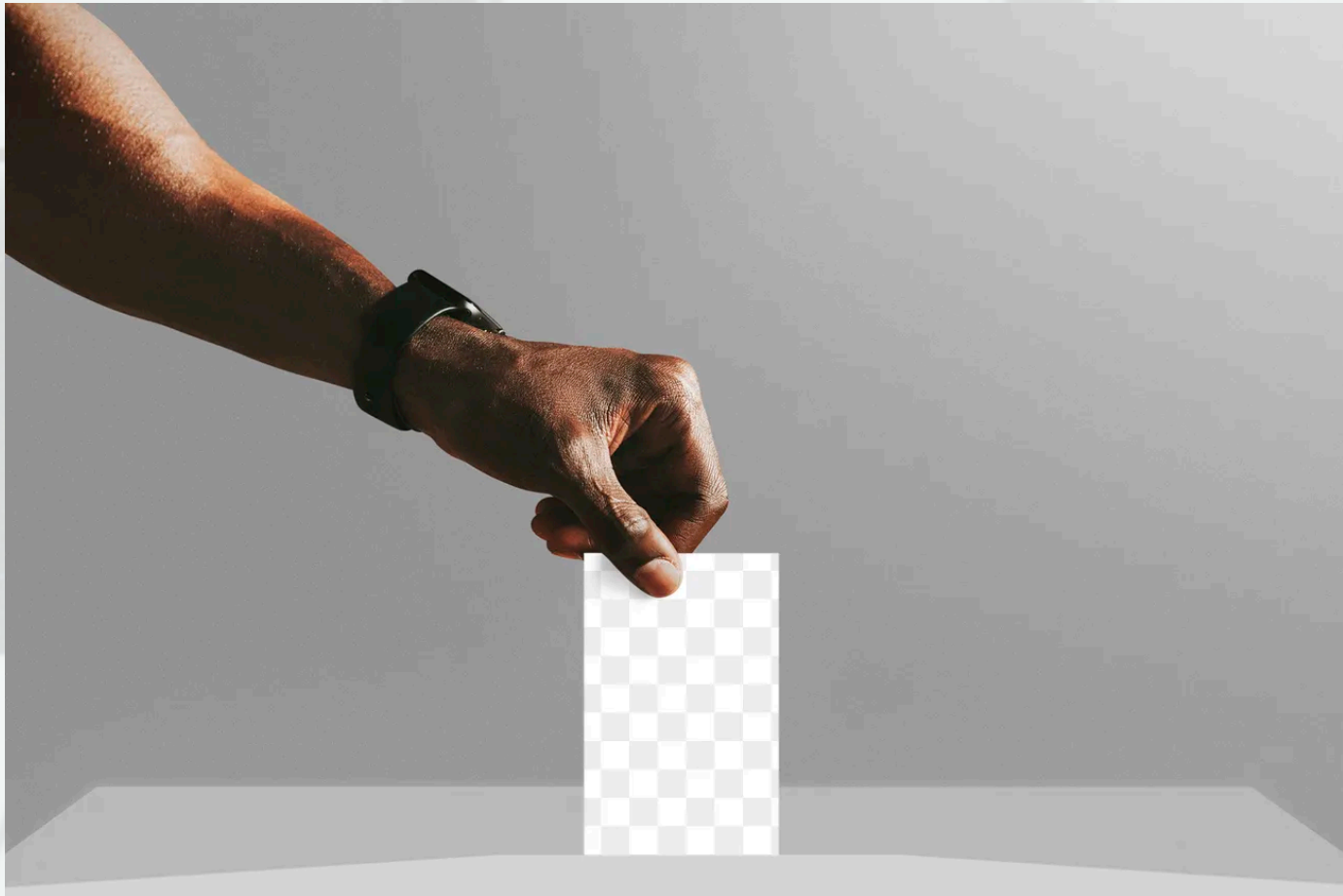
# At end of session

<https://speakerdeck.com/boyarsky>

# Disclaimer

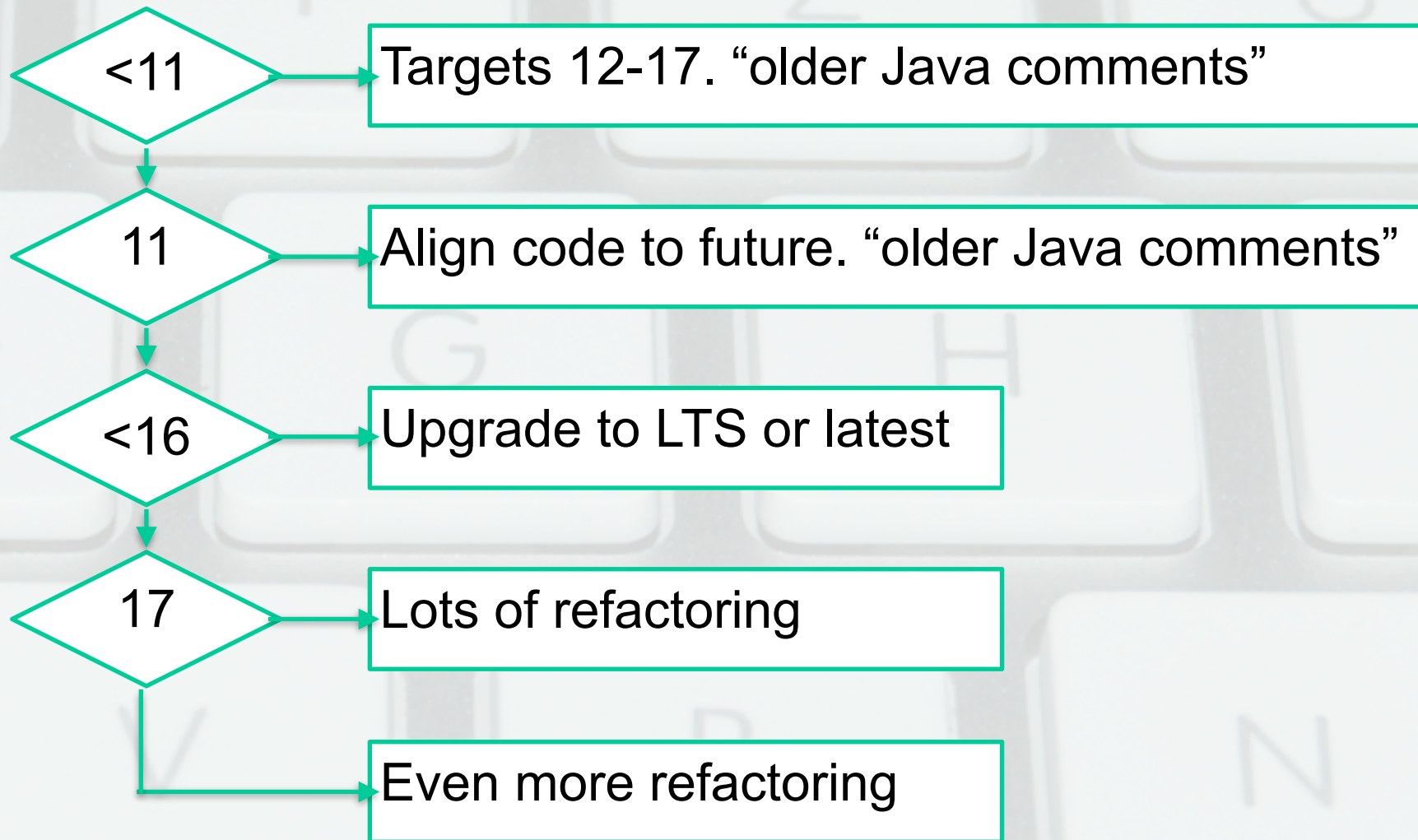
- A bit of the material is from my books.

# Version of Java?





# Version of Java?



# For each Topic

- Example
- About the feature
- Opportunities
- IDE Support
- What to do if on older Java
- What will be explored in more detail in the lab version  
Wednesday....



# Refactoring

- We are writing legacy code now!
- Refactor for future compatibility

# Text blocks and Strings

# Example: REST API Params

```
public String getJson(String search) {  
    String json = "{" +  
        "    \"query\": \"%s\" +  
        "    \"start\": \"1\", +  
        "    \"end\": \"10\" +  
        "}";  
    return String.format(json, search);  
}
```

This is hard to read



# Take Two

```
public String getJson(String search) {  
    Path path = Path.of(  
        "src/main/resources/query.json");  
    String json = null;  
    try {  
        json = Files.readString(path);  
    } catch (IOException e) {  
        throw new UncheckedIOException(e);  
    }  
    return String.format(json, search);  
}
```

Now the String is far away

# Text Block

15

```
public String getJson(String search) {  
    String json = ""  
        {  
            "query": "%s"  
            "start": "1"  
            "end": "10"  
        }"";  
    return String.format(json, search);  
}
```

It's a string literal!

Adds line breaks, but still works

# Text Block Syntax

15

```
String textBlock = """  
    kcdc,Kansas City,"session,workshop"  
    meetup,Various,lecture  
    """;
```

start block

incidental  
whitespace

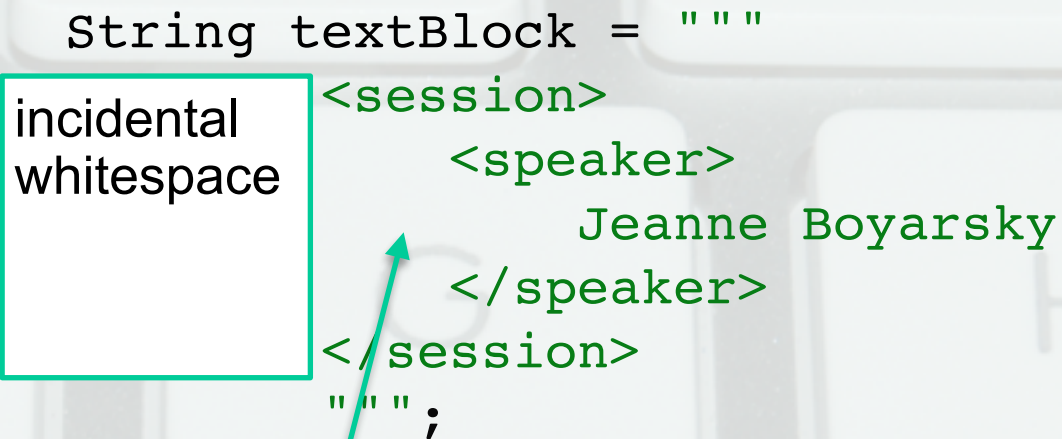
end block



# Essential Whitespace

15

```
String textBlock = ""  
<session>  
    <speaker>  
        Jeanne Boyarsky  
    </speaker>  
</session>  
"";
```



The diagram illustrates the difference between incidental and essential whitespace in XML. A green box labeled "incidental whitespace" points to the spaces between the XML tags and the text "Jeanne Boyarsky". A green arrow labeled "essential whitespace" points to the spaces between the opening and closing tags of the <speaker> element.

essential whitespace

# Ending lines

15

```
String textBlock = """
```

```
<session>
```

```
<speaker>
```

```
    Jeanne Boyarsky
```

```
</speaker>
```

```
<title>
```

```
    Becoming one of the first Java 17 \
```

```
    certified programmers \
```

```
    (and learning new features)
```

```
</title>
```

```
</session>
```

```
""";
```

new escape character  
keeps trailing whitespace

\s

tab

continue on next line  
without a line break

# New lines

15

```
String textBlock = ""
```

```
<session>\n
```

```
<speaker>
```

```
  Jeanne\nBoyarsky
```

```
</speaker>
```

```
</session>"";
```

Two new lines  
(explicit and implicit)

One new line (explicit)

no line break at end



# Escaping Three Quotes

15

```
String textBlock = """  
    better \"""  
    but can do \"\\\"\\\"  
    """;
```

# Opportunities

15

- Externalized data
- Expected values in JUnit
- Formats - CSV, GraphQL, JSON, SQL, Text, XML, YAML, etc
- Others?

# IDE Support



```
// TODO convert to text block when on Java 17
```

```
String json = "{" +
```

```
    "  \"query\": \"%s\" +
```

```
    "  \"st
```

```
    "  \"en
```

```
    \"}\";
```

```
return String.fo
```

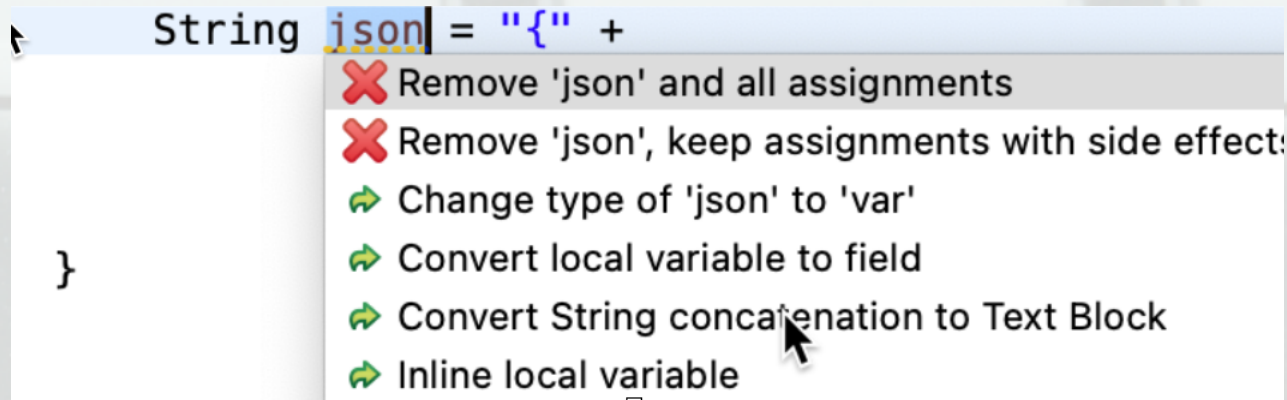
- ✎ Compute constant value of '"{" + " \"query\":'
- ✎ Copy string concatenation text to the clipboard
- ✎ Replace '+' with 'StringBuilder.append()'
- ✎ Replace '+' with 'formatted()'
- ✎ Replace with text block**
- ✎ Split into declaration and assignment

```
String json = """  
    {  "query": "%s"    "start": "1",    "end": "10"}""";
```

Literal refactoring - no \n



# IDE Support



```
String json = ""  
    {\br/>        "query": "%s"\br/>        "start": "1",\br/>        "end": "10"\br/>    }"";
```

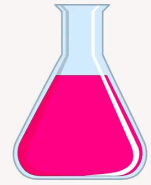
Preserve lines but still no \n

# On older Java?

```
public String getJson(String search) {  
    //TODO convert to text block when on Java 17  
    String json = "{" +  
        "    \"query\": \"%s\" +  
        "    \"start\": \"1\", +  
        "    \"end\": \"10\" +  
        "    }";  
    return String.format(json, search);  
}
```

Hard to read but positions for future

# Wed Lab Version



- Practice identifying valid/invalid text blocks
- Related String APIs
- Hands on practice





Instanceof

# Casting

```
if (num instanceof Integer) {  
    Integer numAsInt = (Integer) num;  
    System.out.println(numAsInt);  
}  
if (num instanceof Double) {  
    Double numAsDouble = (Double) num;  
    System.out.println(numAsDouble.intValue());  
}
```

# Casting

16

```
if (num instanceof Integer numAsInt) {  
    System.out.println(numAsInt);  
}  
if (num instanceof Double numAsDouble) {  
    System.out.println(numAsDouble.intValue());  
}
```

Pattern  
variable





# Flow Scope

16

```
if (num instanceof Double d1  
    && d1.intValue() % 2 == 0) {  
    System.out.println(d1.intValue());  
}
```

Compiles

```
if (num instanceof Double d2  
    || d2.intValue() % 2 == 0) {  
    System.out.println(d2.intValue());  
}
```

Does not  
compile  
because  
d2 might  
not be  
double

# Does this compile?

16

```
if (num instanceof Double n)  
    System.out.println(n.intValue());
```

```
if (num instanceof Integer n)  
    System.out.println(n);
```

Yes. Only in scope for if statement

# Does this compile?

16

```
if (num instanceof Double n)  
    System.out.println(n.intValue());  
  
System.out.println(n.intValue());
```

No. If statement is over



# Does this compile?

16

```
if (!(num instanceof Double n)) {  
    return;  
}  
System.out.println(n.intValue());
```

Yes. Returns early so rest is like an else

# Does this compile?

16

```
if (!(num instanceof Double n)) {  
    return;  
}  
System.out.println(n.intValue());
```

```
if (num instanceof Double n)  
    System.out.println(n.intValue());
```

No. n is still in scope

# Opportunities

16

- Library code
- Equals methods

```
public boolean equals(Object anObject) {  
    if (this == anObject) {  
        return true;  
    }  
    return (anObject instanceof String aString)  
        && (!COMPACT_STRINGS || this.coder == aString.coder)  
        && StringLatin1.equals(value, aString.value);  
}
```

- Others?



# IDE Support



```
if (num instanceof Integer) {  
    Integer numAsInt = (Integer) num;  
    System.out.print(  
        "Replace 'numAsInt' with pattern variable  >
```

```
if (num instanceof Integer numAsInt) {  
    System.out.println(numAsInt);  
}
```

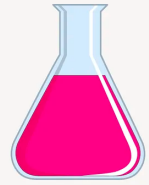
# On older Java?

*//TODO convert to pattern var when on Java 17*

```
if (num instanceof Double) {  
    Double numAsDouble = (Double) num;  
    System.out.println(numAsDouble.intValue());  
}
```

Positions for future

# Wed Lab Version



- Explore edge cases
- Sealed classes
- Hands on practice



# Switch expressions

# Originally

```
public String getLocation(String store) {  
    String result = "";  
    switch (store) {  
        case "Hallmark":  
            result = "KC";  
            break;  
        case "Crayola":  
            result = "PA";  
            break;  
        default:  
            result = "anywhere";  
    }  
    return result;  
}
```

You remembered the breaks, right?

# Switch Expressions

14

```
public String getLocation(String store) {  
    return switch (store) {  
        case "Hallmark" -> "KC";  
        case "Crayola" -> "PA";  
        default -> "anywhere";  
    };  
}
```

Arrow labels

No break keyword



# Missing value

14

```
enum Position { TOP, BOTTOM };
```

```
Position pos = Position.TOP;
```

```
int stmt = switch(pos) {  
  case TOP: yield 1;  
};
```

```
int expr = switch(pos) {  
  case BOTTOM -> 0;  
};
```

Does not compile  
because assigning  
value

(poly expression)

# Pattern matching for switch

19  
preview

```
public int toInt(Object obj) {  
    return switch (obj) {  
        case Integer i -> i;  
        case Double d -> d.intValue();  
        case String s -> Integer.parseInt(s);  
  
        default -> throw new  
            IllegalArgumentException("unknown type");  
    };  
}
```

Reminder: Syntax can change

# But wait, there's more

19  
preview

```
static void printOddOrEven(Object obj) {  
    switch (obj) {  
  
        case Integer i when i % 2 == 1 ->  
            System.out.println("odd");  
  
        case Integer i when i % 2 == 0 ->  
            System.out.println("even");  
  
        default -> System.out.println("not an int");  
    };  
}
```

Reminder: Feature can still change

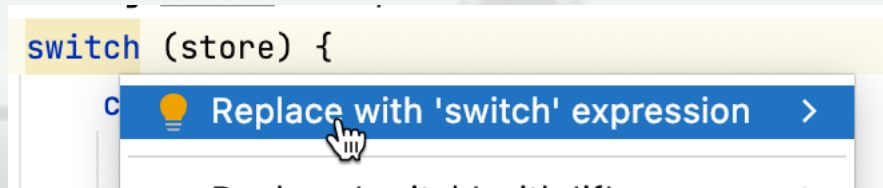


# Opportunities

17

- Many if/else chains!
- Switch statements with many breaks
- Sets the stage for advanced matching
- Others?

# IDE Support



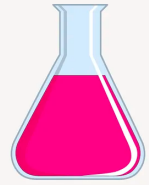
```
String result = switch (store) {  
    case "Hallmark" -> "KC";  
    case "Crayola" -> "PA";  
    default -> "anywhere";  
};  
return result;
```

# On older Java?

```
public String getLocation(String store) {  
    //TODO convert to switch expression on Java 17  
    String result = "";  
    switch (store) {  
        case "Hallmark":  
            result = "KC";  
            break;  
        case "Crayola":  
            result = "PA";  
            break;  
        default:  
            result = "anywhere";  
    }  
    return result;  
}
```



# Wed Lab Version



- Blocks and yield
- Switch with records
- More edge cases
- Hands on practice

# Records

# Originally

```
public class Book {  
  
    2 usages  
    private String title;  
  
    2 usages  
    private int numPages;  
  
    public Book(String title, int numPages) {  
        this.title = title;  
        this.numPages = numPages;  
    }  
  
    public String getTitle() {  
        return title;  
    }  
  
    public int getNumPages() {  
        return numPages;  
    }  
}
```

Ran out of room  
on screen!



# Record

16

```
public record Book (String title, int numPages) {  
}
```

New type

Automatically get

- \* final record
- \* private final instance variables
- \* public accessors
- \* constructor taking both fields
- \* equals
- \* hashCode

# Using the Record

16

```
Book book = new Book("Breaking and entering", 289);
```

```
System.out.println(book.title());  
System.out.println(book.toString());
```

← No "get"

Outputs:

Breaking and entering

Book[title=Breaking and entering, numPages=289]

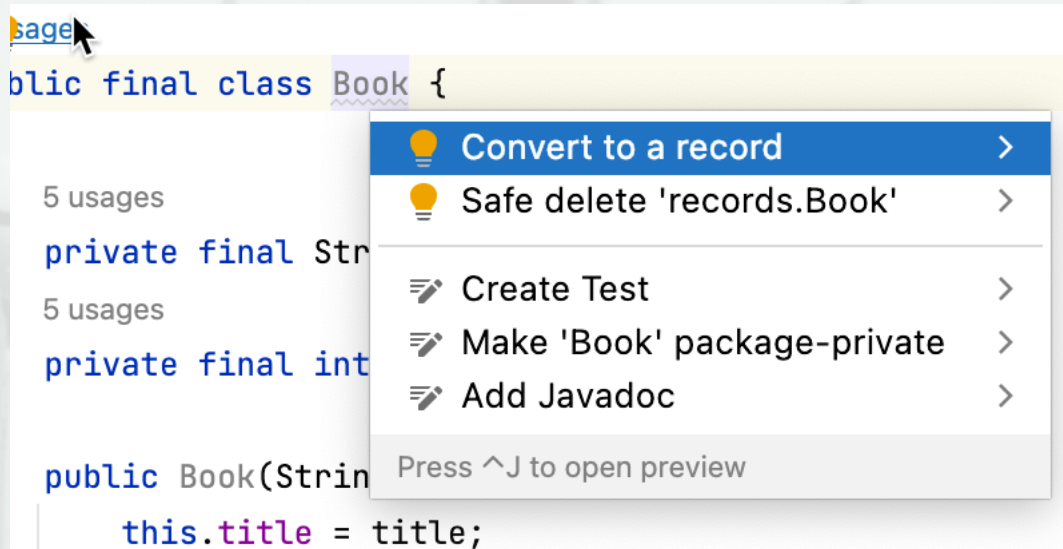
# Opportunities

17

- Immutable POJOs
- Don't have to write equals/hashCode
- Vs reflection - EqualsBuilder
- Make code coverage tool happy
- Others?



# IDE Support



```
public record Book(String title, int numPages) {  
}
```

Had to make instance variables final. Also didn't remove my equals() even though generated by IntelliJ

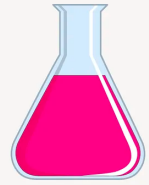
# On older Java?

```
//TODO convert to record when on Java 17
public final class Book {
    private String title;
    private int numPages;

    public Book(String title, int numPages) {
        this.title = title;
        this.numPages = numPages;
    }
    public String title() {
        return title;
    }
    public int numPages() {
        return numPages;
    }
    // hash code, equals
```

Be sure to use all  
fields for equals/  
hashCode

# Wed Lab Version



- Compact constructors
- Custom methods
- More edge cases
- Hands on practice



# APIs

# toList()

16

```
public List<String> listLonger(  
    Stream<String> stream) {  
  
    return stream.collect(Collectors.toList());  
}  
  
public List<String> listShorter(  
    Stream<String> stream) {  
  
    return stream.toList();  
}
```

# Teeing Collector

12

```
record Separations(String spaceSeparated,  
    String commaSeparated) {}
```

```
var list = List.of("x", "y", "z");  
Separations result = list.stream()  
    .collect(Collectors.teeing(  
        Collectors.joining(" "),  
        Collectors.joining(","),  
        (s, c) -> new Separations(s, c)));
```

```
System.out.println(result);
```



# Formatting a String

12

```
String firstName = "Jeanne";  
String lastName = "Boyarsky";  
String str = String.format(  
    "Hi %s %s!", firstName, lastName);  
System.out.println(str);  
  
System.out.println("Hi %s %s!".formatted(  
    firstName, lastName));
```

Outputs:  
Hi Jeanne Boyarsky!  
Hi Jeanne Boyarsky!

# Common Conversions

Conversion	What it does
%s	Formattable as String
%d	Decimal integer (no dot)
%c	Char
%f	Float (decimal)
%n	New line

Many more out of scope. Examples:

- %e - scientific notation
- %t - time
- %S - converts to all uppercase

# Conversion Examples

12

Code	Output
<code>"%d%%".formatted(1.2)</code>	exception
<code>"%d%%".formatted(1)</code>	1%
<code>"%s%%".formatted(1)</code>	1%
<code>"%s%%".formatted(1.2)</code>	1.2%
<code>"%f%%".formatted(1.2)</code>	1.200000f



# Formatting a Number

Char	What it does
-	Left justified
+	Always include +/-
space	Leading space if positive

Char	What it does
0	Zero padded
,	Group numbers
(	Negative # in parens

# Flag Examples

12

Code	Output
<code>"%,d".formatted(1234)</code>	1,234
<code>"%+d".formatted(1234)</code>	1234
<code>"% d".formatted(1234)</code>	1234
<code>"%, (d".formatted(-1234)</code>	(1,234)
<code>"%,f".formatted(1.23456789)</code>	1.234568

# Compact Number

12

```
NumberFormat defaultFormat =  
NumberFormat.getCompactNumberInstance();  
NumberFormat shortFormat = NumberFormat  
    .getCompactNumberInstance(  
        Locale.US, NumberFormat.Style.SHORT);  
NumberFormat longFormat = NumberFormat  
    .getCompactNumberInstance(  
        Locale.US, NumberFormat.Style.LONG);  
  
System.out.println(defaultFormat.format(1_000_000));  
System.out.println(shortFormat.format(1_000_000));  
System.out.println(longFormat.format(1_000_000));
```

1M  
1M  
1 million



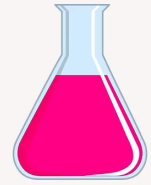
# New Files.mismatch()

12

```
Path kcdc = Path.of("files/kcdc.txt");  
Path kc = Path.of("files/kc.txt");  
  
System.out.println(Files.mismatch(kcdc, kc));  
System.out.println(Files.mismatch(kcdc, kcdc));
```

11 (index of first character different)  
-1 (same file contents regardless of whether exists)

# Wed Lab Version



- Hands on practice

# Book Giveaway

