Java 12 to 15

Rahman Usta
rahmanusta@kodedu.com

/ustarahman

/rahmanusta

Rahman Usta

- Senior Developer and Software Trainer in Luxoft Poland
- Based in Istanbul
- Book author and blogger
- Java champion
- Open source contributor
  - https://github.com/rahmanusta

Java 12 to 15 Features

- JEP 358 - Helpful NullPointerExceptions^{14}
- JEP 361 - Switch Expressions (Standard^{14})
- JEP 375 - Pattern Matching for instanceof (Second Preview^{15})
- JEP 384 - Records Component (Second Preview^{15})
- JEP 360 - Sealed Classes (Preview^{15})
- JEP 378 - Multiline Text Blocks (Standard^{15})

JDK 12: https://openjdk.java.net/projects/jdk/12/
JDK 13: https://openjdk.java.net/projects/jdk/13/
JDK 14: https://openjdk.java.net/projects/jdk/14/
JDK 15: https://openjdk.java.net/projects/jdk/15/

New release every 6 months

Java 10, Java 11 (LTS) => 2018 (March & September)

Java 12, Java 13 => 2019 (March & September)

Java 14, Java 15 => 2020 (March & September)

- LTS -> Long Term Support
- New LTS every 3 years

Preview Features

- Not a standard yet
- Ready to try, review and feedback
- May be changed, even removed
- Can be staged, first preview, second preview etc.

Enable preview features, disabled by default

`--enable-preview`

JEP 358: Helpful NullPointerExceptions

Pre Helpful NullPointerExceptions

```
Person person = new Person();
person.address = new Address();

String toUpperCase = person.address.street.toUpperCase();
System.out.println(toUpperCase);
```

```
Exception in thread "main" java.lang.NullPointerException
	at java14.edu/com.kodedu.NullPointerException.main(NullPointerException.java:10)
```

Helpful NullPointerExceptions

```
Person person = new Person();
person.address = new Address();

String toUpperCase = person.address.street.toUpperCase();
System.out.println(toUpperCase);
```

New flag! Enabled by default from Java 15+
-XX:+ShowCodeDetailsInExceptionMessages

```
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.toUpperCase()"
because "person.address.street" is null
	at java14.edu/com.kodedu.NullPointerException.main(NullPointerException.java:10)
```

JEP 361: Switch Expressions (Standard^{14})

Pre Switch Expressions

```java
int speedLimit=-1;
switch (vehicleType) {
   case BIKE:
   case SCOOTER:
       speedLimit = 40;
       break;
   case MOTORBIKE:
   case AUTOMOBILE:
       speedLimit = 140;
       break;
   case TRUCK:
       speedLimit = 80;
       break;
}

System.out.println("Speed limit: " + speedLimit);
```

Switch Expressions

```
VehicleType vehicleType = VehicleType.AUTOMOBILE;

int speedLimit = switch (vehicleType) {
   case BIKE, SCOOTER -> 40;
   case MOTORBIKE, AUTOMOBILE -> 140;
   case TRUCK -> 80;
};

System.out.println("Speed limit: " + speedLimit);
```

All enum cases have to be covered in switch block!

Switch Expressions : return a switch

```java
int speedLimit = getSpeedLimit(VehicleType.AUTOMOBILE);
System.out.println("Speed limit: " + speedLimit);

private static int getSpeedLimit(VehicleType vehicleType) {
   return switch (vehicleType) {
       case BIKE, SCOOTER -> 40;
       case MOTORBIKE, AUTOMOBILE -> 140;
       case TRUCK -> 80;
   };
}
```

Switch Expressions : block and yield

```
VehicleType vehicleType = VehicleType.TRUCK;

int speedLimit = switch (vehicleType) {
   case BIKE, SCOOTER -> 40;
   case MOTORBIKE, AUTOMOBILE -> 140;
   case TRUCK -> {
       int randomSpeed = ThreadLocalRandom.current().nextInt(70, 80);
       yield randomSpeed;
   }
};

System.out.println("Speed limit: " + speedLimit);
```

JEP 375: Pattern Matching for instanceof (Second Preview[15])

Pre Pattern Matching

```
Object obj = "Hello world!";

if (obj instanceof String) {
   String s = (String) obj;
   System.out.println("String: " + s);
}
```

Pattern Matching

```
if (obj instanceof String s) {
   System.out.println("String: " + s);
}
```

```
// cannot resolve symbol 's'
if (obj instanceof String s || !s.isBlank()) {
   System.out.println("String: " + s);
}
```

```
// legal usage
if (obj instanceof String s && !s.isBlank()) {
   System.out.println("String: " + s);
}
```

JEP 384: Records (Second Preview[15])

Pre Records

```
public class Point {
    private int x;
    private int y;

    // constructor
    // setters & getters
    // equals & hashcode
    // toString

}
```

```
public Point(int x, int y) {
    this.x = x;
    this.y = y;
}
```

```
public int getX() {
    return x;
}

public void setX(int x) {
    this.x = x;
}

public int getY() {
    return y;
}

public void setY(int y) {
    this.y = y;
}
```

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Point point = (Point) o;
    return x == point.x &&
            y == point.y;
}

@Override
public int hashCode() {
    return Objects.hash(x, y);
}
```

```
@Override
public String toString() {
    return "Point{" +
            "x=" + x +
            ", y=" + y +
            '}';
}
```

Records

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

- 1 canonical constructor
- Final record and fields
- No setter but getters -> point.x() , point.y()
  - Records are immutable!
- Default implementation of hashCode and equals
- A standard toString implementation “Point[x=1, y=2]”
- Default characteristic can be overridden

JEP 360: Sealed Classes (Preview[15])

Sealed Classes

too **restrictive**

A **final** class

cannot have any subclass(es)

restrictive as API developer's desire

A **sealed** class

defines what are the sum of subtypes.

too **permissive**

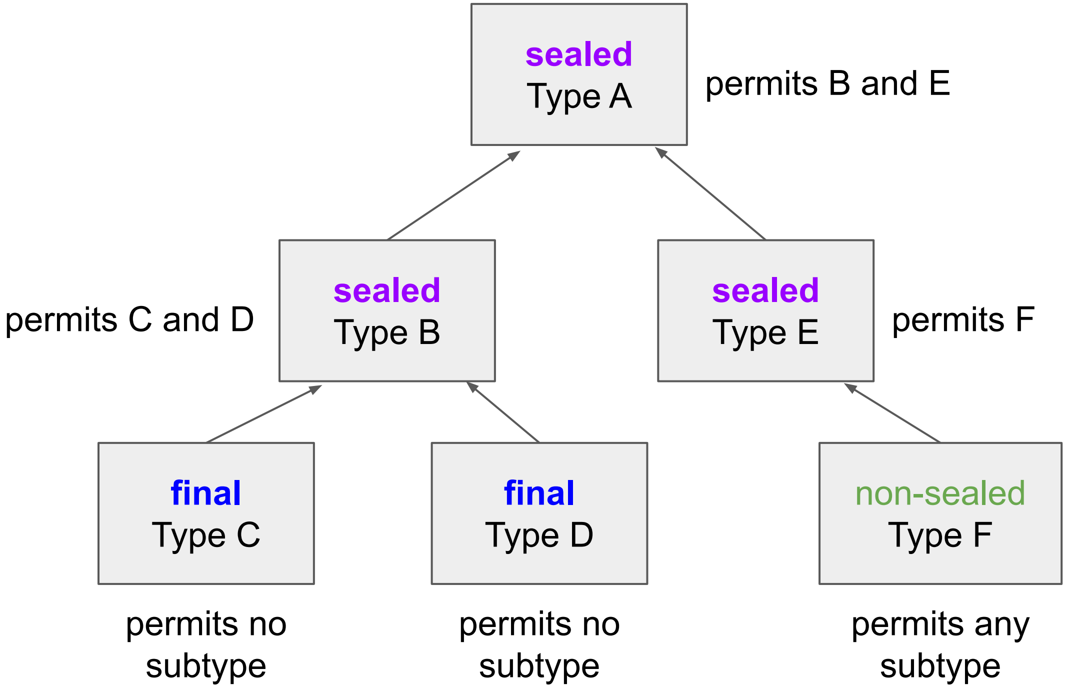
A **non-final** class

may have subclass(es)

Sealed Classes

```
public sealed class Shape permits Square, Circle {

}

public final class Square extends Shape {

}

public final class Circle extends Shape {

}
```

Sealed Classes : Exhaustive



Algebraic types: Records + Sealed Classes

```
sealed interface Expr permits  ConstantExpr, NegExpr, PlusExpr, TimesExpr   { }

record ConstantExpr(int i) implements Expr { }
record PlusExpr(Expr a, Expr b) implements Expr { }
record TimesExpr(Expr a, Expr b) implements Expr { }
record NegExpr(Expr e) implements Expr { }
```

- Records
  - Defines product types
- Sealed classes
  - Defines sum of types

```
int calculate(Expr e) {
        return switch (e) {
                case ConstantExpr(var i) -> i;
                case PlusExpr(var a, var b) -> calculate(a) + calculate(b);
                case TimesExpr(var a, var b) -> calculate(a) * calculate(b);
                case NegExpr(var e) -> -calculate(e);
                // no default needed, Expr is sealed
   }
}
```

JEP 378: Text Blocks (Standard^{15})

Pre Text Blocks

```
String html = "<html>\n" +
              "    <body>\n" +
              "        <p>Hello, world</p>\n" +
              "    </body>\n" +
              "</html>\n";
```

Text Blocks

```
String html = """
              <html>
                  <body>
                      <p>Hello, world</p>
                  </body>
              </html>
              """;
```

```
<html>⏎
••••<body>⏎
••••••••<p>Hello,•world</p>⏎
••••</body>⏎
</html>⏎
```

Text Blocks

```java
// ""
var text = """
            """;
```

Line terminator required after opening delimiter

```java
// illegal text block start
var text = """""";
```

Text Blocks : Indentation

```
String html = """
                  <html>
                     <body>
                       <p>Hello, world</p>
                     </body>
                  </html>
              """;
```

```
••••<html>⏎
•••••••<body>⏎
•••••••••<p>Hello,•world</p>⏎
•••••••</body>⏎
••••</html>⏎
```

Text Blocks : Indentation

```
String html = """
                <html>
                   <body>
                     <p>Hello, world</p>
                   </body>
                </html>
                """;
```

```
<html>⏎
•••<body>⏎
•••••<p>Hello,•world</p>⏎
•••</body>⏎
</html>⏎
```

Text Blocks : Espace line terminator

```
String html = """
              <html> \
                 <body> \
                   <p>Hello, world</p> \
                 </body> \
              </html> \
              """;
```

```
<html>••••<body>••••••<p>Hello,•world</p>•••</body>•</html>•
```

Text Blocks : Single space character

```
String html = """
       <html>                  \s
          <body>               \s
            <p>Hello, world</p>\s
          </body>              \s
       </html>                 \s
       """;
```

```
<html>•••••••••••••••••••⏎
•••<body>••••••••••••••••⏎
•••••<p>Hello,•world</p>•⏎
•••</body>•••••••••••••••⏎
</html>••••••••••••••••••⏎
```

Text Blocks : String#formatted

```
String html = """
              <html>
                 <body>
                   <p>%s, %s</p>
                 </body>
              </html>
              """;
html.formatted("Hello", "world");
```

```
<html>⏎
•••<body>⏎
•••••<p>Hello,•world</p>⏎
•••</body>⏎
</html>⏎
```

Try Java 15

Open-source builds

- https://jdk.java.net/15

Online Java Shell

- https://tryjshell.org/

Code samples

- https://github.com/rahmanusta/java15-edu

Thank you!