Istanbul
Java User Group
https://istanbul-jug.org

Java 14

Rahman Usta
rahmanusta@kodedu.com

/ustarahman

/rahmanusta

Java 14 Features

- JEP 358 - Helpful NullPointerExceptions
- JEP 361 - Switch Expressions (Standard)
- JEP 359 - Records Component (Preview)
- JEP 368 - Multiline Text Blocks (Second Preview)
- JEP 305 - Pattern Matching for instanceof (Preview)

All 16 features: https://openjdk.java.net/projects/jdk/14/

New release every 6 months

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

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

Java 14 => 2020 (March)

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

Preview Features

- Not standard yet
- Ready to try, review and feedback
- May be changed, even removed

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!
-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)

Pre Switch Expressions

```
int speedLimit;
switch (vehicleType) {
   case BIKE:
   case SCOOTER:
      speedLimit = 40;
      break;
   case MOTORBIKE:
   case AUTOMOBILE:
      speedLimit = 140;
      break;
   case TRUCK:
      speedLimit = 80;
      break;
   default:
      throw new IllegalStateException("No case found for: " + vehicleType);
}

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;
   default -> throw new IllegalStateException("No case found for: " + vehicleType);
};

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

default is not required if all enum cases are covered.

Switch Expressions

```
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 : 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 359: Records (Preview)

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 368: Text Blocks (Second Preview)

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

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

Line terminator required after opening delimiter

```
// 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("Merhaba", "Dünya");
```

```
<html>⏎
•••<body>⏎
•••••<p>Merhaba,•Dünya</p>⏎
•••</body>⏎
</html>⏎
```

JEP 305: Pattern Matching for instanceof (Preview)

Pre Pattern Matching

```
Object obj = "Merhaba Dünya";

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);
}
```

Try Java 14

Open-source builds

- https://jdk.java.net/14/

Online Java Shell

- https://tryjshell.org/

Code samples

- https://github.com/rahmanusta/java14-edu

Thank you!