Java 14 – nowości

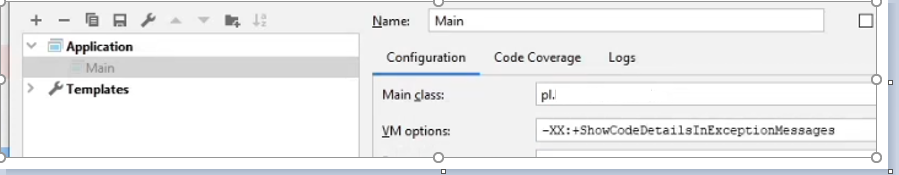
Pojawiły się praktycznie dwie zmiany, które ułatwiają pisanie kodu. Pozostałe cztery są jeszcze w fazie preview.

* **Helpful NullPointerExceptions**
* **Switch Expressions (standard)**
* Packaging Tool (incubator)
* Text Blocks (second preview)
* Records (preview)
* Pattern Matching for instanceof (preview)

# Helpful NullPointerException

Od tej pory będziemy wiedzieć gdzie tak naprawdę wystąpił null.

Najpierw należy ustawić opcję kompilacji w **Edit Configuration**



Jeśli nie da się uruchomić (java 5 not supported) to trzeba zmienić info o compiler w Settings

# Switch Expressions (standard)

Weszło już w wersji 12. Zostało poprawione w wersji 13. Od wersji 14 można korzystać z tego jako standard. Już nie trzeba używać **break**

switch (number) {  
 case 0 -> System.*out*.println("zero");  
 case 1 -> System.*out*.println("jeden");  
 case 2 -> System.*out*.println("trzy");  
}

Switch może nam dać także jakąś wartość

var result = switch (number) {  
 case 0 -> {  
 yield 0;  
 }  
 case 1,2 -> 1;  
 default -> {  
 yield "default";  
 }  
};  
System.*out*.println("Result "+result);

yield to jest odpowiednik returna. Jak widać, nie trzeba nawet pisać yield, ponieważ wystarczy podać wynik (case 1,2)

# PackagingTool (incubator)

Pozwala spakować naszą aplikację do różnych typów.

Aby spakować trzeba w powershellu w folderze target wykonać następującą komendę

**jpackage –n nazwa –i lib - - main-jar .\nazwa.jar –d out**

# Text Blocks (second preview)

W Java 13 text blocks były w wersji preview. Text Blocks zostały już dodane do Java 15.

String hello= """  
 Hello \s\s\s  
 How \  
 are \  
 you   
 """;

A wynikiem będzie:

Hello (trzy spacje)

How are you

# Records (preview)

public record Person(String name,String surname) {  
}

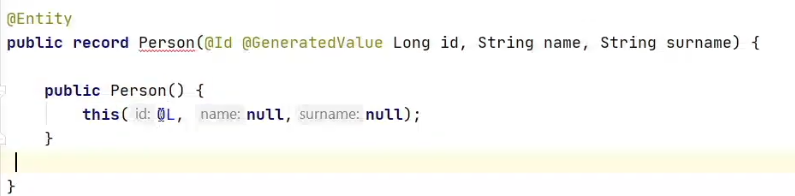
jest kompilowany jako

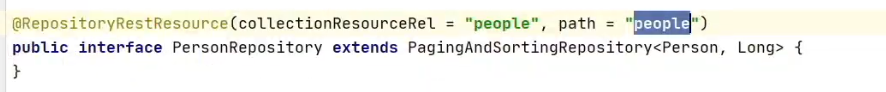
public final class Person extends java.lang.Record {  
 private final java.lang.String name;  
 private final java.lang.String surname;  
  
 public Person(java.lang.String name, java.lang.String surname) { /\* compiled code \*/ }  
  
 public java.lang.String toString() { /\* compiled code \*/ }  
  
 public final int hashCode() { /\* compiled code \*/ }  
  
 public final boolean equals(java.lang.Object o) { /\* compiled code \*/ }  
  
 public java.lang.String name() { /\* compiled code \*/ }  
  
 public java.lang.String surname() { /\* compiled code \*/ }  
}

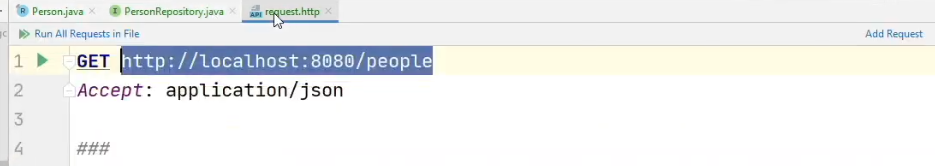
Do rekordu można dopisywać własne metody. Zmienne możemy też dodawać, ale one muszą już być statyczne. Rekord jest immutable czyli Thread-Safe.

**Przykład spring**

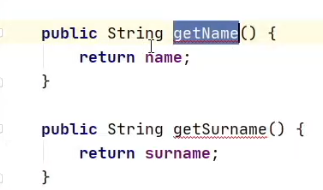
Zamiast Spring Web dajemy Rest Repositories







Żeby to jednak dobrze działało to musimy dodać gettery z nazwą get na początku do rekordu Person



Kolejna wada: W Hibernate możemy dziedziczyć, a rekord jest już klasą finalną a takiej klasy dziedziczyć nie można.

# Pattern Matching for instanceof (preview)

Object person = new Person("Jacek","Tomasz");  
if (person instanceof Person newPerson){  
 System.*out*.println(newPerson);  
}

Nie trzeba więc już rzutować