Java™ Records for the Intrigued

Piotr Przybył

feat. Rustam Mehmandarov

© 2021 Piotr Przybył. Licensed under CC BY-NC-SA 4.0
Java and OpenJDK are trademarks or registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.
Image by Tibor Janosi Mozes from Pixabay
$ whoami

👤 Piotr Przybył
👤 Пётр Пшыбыл
🐦 piotrprz
👨‍💻 Remote Freelance Software Gardener

馗 SOFTWARE GARDEN.dev

👨‍💻 Trainer
$ env EXPERT=Rustam

=$((Rustam Mehmandarov)

Twitter rmehmandarov

who are you
CAVEAT AVDITORES!

A.K.A. Safe harbour statement: don’t assume anything based on this presentation. Verify on your own. Errare humanum est.
Records

standard feature

introduced in JEP-395
Record

new kind of type declaration

record is a restricted form of class extends java.lang.Record

"transparent carriers for immutable data"

record Complex(double real, double imaginary) {}
Records have

- a name
- record components \((\geq 0)\)
- which become private final fields
- generated accessors
- generated full canonical constructor
- generated equals, hashCode, toString
- body {}
Records can

- (re-)define constructors: compact canonical, full canonical, custom
- have own implementations of generated methods
- (which should obey the invariants/rules)
- have extra methods
- have static fields & methods
- implement interfaces
- be generic
Records can’t

- extend classes or be extended
- have setters *
- have any "extra" instance fields
- have "less visible" canonical constructors
- declare native methods
- assign components in compact constructors
Records

- are not Java Beans
- are POJOs with accessors (without getters)
- think "named tuples"
- sometimes require overriding equals() and hashCode()!
Nesting records

- nesting records just like static classes
- local records are very handy for streams, reduce and collectors as intermediate result type
Local declarations

As a by-product, interfaces and enums can be declared as local too (apart from records).
Reflection of records

- new method `Class.isRecord()`
- new method `Class.getRecordComponents()`
Annotations

Annotations from components get "propagated" where their `@Target(ElementType....)` permits to.
Serialization of records

- re-construction of objects using constructor
- libraries and frameworks need to adapt...?
copy()

Records don't have any `copy()` or `with()` method.

```java
record Pair <FIRST, SECOND>(FIRST first, SECOND second) {}
var pair = new Pair<>("first", "second");
var copy = new Pair<>(pair.first(), pair.second());
```

That's why it's important for accessors, `equals()` and `hashCode()` to obey contracts!
Using records with other features

- already support PM with `instanceof`
- and sealed hierarchies
- in the future will support PM with deconstruction
More on records

softwaregarden.dev/en/tags/records
Będę myśleć o Twojej prezentacji po nocach :)

Good job :)}
?” 🔄 How was it?

Java™ Records for the Intrigued

Большое спасибо!

📷 Piotr Przybył
/rss/ piotrprz

https://SoftwareGarden.dev