

Neměnné objekty a záznamy (**record**)

Table of Contents

Neměnné objekty	1
Příklad neměnného objektu	1
Totéž jako záznam (record)	2
Co nabízí record ?	2
Metody v record	2
Výhody a nevýhody neměnných objektů	2
Neměnný objekt vs (final) odkaz	3
Vestavěné neměnné třídy	3

Neměnné objekty

- Neměnný (*immutable*) objekt nemůže být po jeho vytvoření modifikován
- Bezpečně víme, co v něm až do konce života bude
- Tudíž může být souběžně používán z více míst, aniž by hrozily nekonzistence
- Jsou prostředkem, jak psát robustní a bezpečný kód
- K jejich vytvoření nepotřebujeme žádný speciální nástroj
- Od nových verzí Javy (14+) lze pro tento účel s výhodou využít typy **record** (záznam)

Příklad neměnného objektu

```
public class Vertex1D {
    private int x;
    public Vertex1D(int x) {
        this.x = x;
    }
    public int getX() {
        return x;
    }
}
...
Vertex1D v = new Vertex1D(1);
// x of value 1 cannot be changed anymore
```

Totéž jako záznam (**record**)

- Namísto třídy `class` můžeme použít `record`
- Tím se definuje třída, kde každý objekt bude mít vlastnosti `name` a `address`, které se nastaví jednou a už nejdou změnit, čili jako výše `Vertex1D`.

```
public record Person (String name, String address) {}  
//...  
Person p = new Person("Pavel Holec", "Lipová 3, Brno");
```

Co nabízí **record**?

- `record` je vlastně typ *tříd neměnných objektů*.
- Překladač pro nás automaticky pro tuto třídu vytvoří:
 - konstruktor mající parametr pro nastavení každého atributu (hodnoty) v objektu
 - přístupové metody pro čtení atributů, např. `person.name()`

Metody v **record**

- Dále pak se "samy vytvoří":
 - metoda `equals` pro porovnání objektů: dva záznamy budou stejné \Leftrightarrow jsou stejné všechny odpovídající si atributy
 - metoda `hashCode()` konzistentní s `equals()`
 - "inteligentní" metoda `toString` vracející např. `Person[name=John Doe, address=100 Linda Ln.]`



Jména metod jsou odlišná od konvence používané u JavaBeans nebo obecně u javových objektů: tradičně by bylo `person.getName()`, ale u záznamu `person.name()`

Výhody a nevýhody neměnných objektů

Výhody

- je to vláknově bezpečné (*thread safe*) — objekt může být *bezpečně* používán více vlákny naráz
- programátor má jistotu, že se mu obsah objektu *nezmění* — silný předpoklad
- kód je *čitelnější, udržovanější i bezpečnější* (např. útočník nemůže změnit náš token)

Nevýhody

- chceme-li objekt být jen drobně změnit, musíme vytvořit nový
- to stojí čas a paměť

Neměnný objekt vs (**final**) odkaz

final odkaz

```
final Person p = new Person("Marek");  
// reference cannot be changed  
// p = new Person("Jan");  
// but object itself can be changed  
p.setName("Haha I changed it"); // this works!
```

Neměnný objekt

```
record Person(String name) {}  
Person p = new Person("Marek");  
// reference can be changed  
p = new Person("Jan");  
// but object itself cannot be changed
```

Vestavěné neměnné třídy

- Neměnnou třídou v Javě je **String**.
- Má to řadu dobrých důvodů — tytéž jednou definované řetězce lze používat souběžně z více míst programu.
- Nicméně i negativní stránky — někdy větší režie spojená s nemodifikovatelností — nutnost při sebemenší změně znovu vytvářet.
- Velkou skupinou vestavěných neměnných objektů jsou tzv. objektové obálky primitivních hodnot (třeba **Integer**).