Vnořené třídy

Table of Contents

Vnořené, vnitřní a anonymní třídy . . . 1
Vnořená třída . . . 1
Statická vnořená třída . . . 2
Příklad statické vnořené . . . 2
Příklad statické vnořené `Map.Entry` . . . 2
Vnitřní . . . 3
Příklad vnitřní . . . 3
Anonymní . . . 3

Vnořené, vnitřní a anonymní třídy

Vnořené

definované uvnitř jiné třídy jako její prvek - podobně jako metody nebo atributy

Lokální

definované uvnitř metody podobně jako lokální proměnná

Vnitřní

jako vnořené, ale potřebují ke své instanciaci *objekt* vnější třídy, v níž jsou definovány `Inner inner = outer.new Inner()`

Anonymní

vnitřní, ovšem *nepojmenované*, pojmenuje si je překladač sám stylem `Person$1`

Vnořená třída

Definovaná uvnitř jiné třídy jako její prvek - podobně jako metody nebo atributy.

Vnořená (static nested class)

- je uvozené klíčovým slovem `static`, pak se jedná o *statickou vnořenou třídu*
- objekt takové třídy potřebuje pro své vytvoření samotnou obklopující třídu, jednotlivý objekt není třeba
- `Nested nested = new Outer.Nested()`

Vnitřní (inner class)

- chybí-li `static`, jedná se o *nestatickou vnořenou třídu*
- objekt takové třídy potřebuje pro své vytvoření objekt své nadřazené, obklopující třídy
- `Inner inner = outer.new Inner()`

Statická vnořená třída

- Použití statické vnořené třídy je dobrá praktika — zapouzdření.
- Lze ji mít jako `private`, pak není vidět zvenčí.
- Přesto je použitelná v rámci atributů metod obklopující vnější třídy.
- I opačně — statická vnořená třída může používat *i privátní metody a atributy* své vnější třídy.

Příklad statické vnořené

```
public class Enclosing {
    private static int x = 1;
    public static class StaticNested {
        private void run() { ...}
    }
    public void test() {
        // StaticNested is of course visibile here -
        // even if it were private
        StaticNested nested = new StaticNested();
        nested.run();
    }
}
class Another {
    public void test() {
        // works since both Enclosing & StaticNested are visible here
        Enclosing.StaticNested nested = new Enclosing.StaticNested();
        nested.run();
    }
}
```

blíže viz Java Nested Classes

Příklad statické vnořené `Map.Entry`

```
public class MapDemo {
    public static void main(String[] args) {
        Map<String, String> map = new HashMap<>();
        for(Map.Entry<String, String> entry: map.entries()) {
            // do something for each (key, value) pair:
            // entry.getKey(), entry.getValue()
            // or entry.setValue(...)
        }
    }
}
```

blíže viz Interface Map.Entry<K,V>

Vnitřní

- *Vnitřní* = nestatická vnořená třída
- Objekt takové třídy potřebuje ke své instanciaci objekt své vnější třídy.
- `Inner inner = outer.new Inner()`

Příklad vnitřní

```
public class KarelGame {
    // will be visible from Karel, too
    private static enum Heading { LEFT, RIGHT }
    private int[] field = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
    // this Karel is bound to a KarelGame
    private Karel karel = new Karel();
    // inner class
    public class Karel {
        private int position = 0;
        public void put() {
            // the field in KarelGame is accessible from Karel's method
            // even though it is private
            if(field[position] < MAX_ITEMS) field[position]++;
```

Anonymní

- vnitřní, ovšem nepojmenované
- na jednom místě se definuje a zároveň i jednou použije (instanciuje)
- většinou jako implementace rozhraní nebo abstraktní třídy