Viditelnost (práva přístupu)

Table of Contents

Základní principy OOP

Viditelnost souvisí s fundamenty objektového programování:

- Zapouzdření
- Dědičnost
- Polymorfismus

Dědičnost

- *Dědičnost* umožňuje vztah *X is Y* mezi objekty
- např. `Human` is `LivingEntity`.
- Umožňuje rozšířit již existující třídu, tedy vytvořit její podtřídu (podtyp), která od svého rodiče zdědí jeho atributy a metody.

Proč?

- Znovupoužití kódu. Nemusíme v každé živé bytosti (člověku, psu…) znovu programovat, co už umí `LivingEntity`.

Polymorfismus

- *Polymorfismus* je schopnost objektu měnit vnímání počas běhu.
- `Object o = new String("String, but can be seen as general Object")`.
- Na místo, kde je očekávána instance třídy `Object`, je možné dosadit instanci jakékoli její podtřídy.

Proč?

- Kde je očekávána živá bytost `LivingEntity`, lze nasadit člověka `Human`.
- V množině živých bytostí můžeme mít současně lidi i psy.

Zapouzdření

- *Zapouzdření* je zabalení dat a metod do jedné komponenty (třídy).
- Zabalená "věc", objekt, nejenže soustřeďuje více položek (dat i metod), ale může některé z nich *navenek skrývat*.
- Princip je, že *Cokoli, co nemusí být viditelné, ani viditelné být nemá.*

Proč?

- Co není vidět, nelze zneužít. Můžeme to později bez újmy odebrat či změnit.

Viditelnost

- Co nemusí být vidět, ať vidět není.
- Daná věc (atribut, metoda) v objektu je, ale ne všichni ji vidí.
- Použití *tříd* i jejich metod a atributů lze regulovat (uvedením tzv. *modifikátoru přístupu*).
- Nastavením správné viditelnosti jsme schopni docílit skutečného zapouzdření.
- Omezení viditelnosti je *kontrolováno při překladu* → není-li přístup povolen, nelze program přeložit.
- Metody i atributy uvnitř třídy mohou mít viditelnost stejnou jako třída nebo nižší.

Typy viditelnosti / přístupu

- `public` = veřejný
- `protected` = chráněný
- *modifikátor neuveden* = říká se *privátní v balíku* či *lokální v balíku* (package-private, package-local)
- `private` = soukromý

Tabulka viditelností

Modifier	Class	Package	Subclass (diff. package)	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Třídy nemohou být `protected`, ale mohou být *package-local*.

- Access Levels table

Použití typů viditelnosti v tomto kurzu

public
: třídy/rozhraní, metody, konstanty

private
: atributy, metody, konstanty

protected
: pravděpobně nebudeme používat, výjimečně metody, atributy

package-local
: pravděpobně nebudeme používat

Veřejný, `public`

Přístupné odevšad

```
public class Account { ... }
```

- U třídy `Account` lze např.
  - vytvořit objekt typu `Account` v metodě jiné třídy
  - deklarovat podtřídu třídy `Account` ve stejném i jiném balíku
- ne všechny vlastnosti uvnitř `Account` musejí vždy být veřejné
- veřejné bývají obvykle některé *konstruktory* a některé *metody*
- veřejné jsou typicky metody předepsané implementovaným *rozhraním*
- třídy deklarované jako *veřejné* musí být umístěné do souboru s totožným názvem: `Account.java`

Soukromý, `private`

Viditelné **jen** v rámci třídy, přesněji mezi `{ }`.

```
public class Account {
   private String owner;
   ...
   public void add(Account another) {
     another.owner; // can be accessed!
     ...
   }
}
```

`private String owner`

- K atributu `owner` nelze přistoupit ani v podtřídě, pouze v dané třídě.
- Pro zpřístupnění proměnné pro "vnější" potřeby je nutno použít gettery/settery.
- Skrýváme konkrétní implementaci datové položky.
- Např. metoda `getAge()` nemusí existovat jako proměnná, ale může se v případě volání spočítat.

Volbou `private` nic zásadně nepokazíme.

Soukromé třídy

Třídy mohou mít viditelnost `private`.

Proč by někdo chtěl privátní třídu?

```
public class SomeClass {
  private class InnerDataStructure { ... }
  // code using InnerDataStructure
}
```

- Používá se u vnořených tříd (tříd uvnitř tříd).
- Mimo rozsah předmětu, nebudeme používat!

Ve stejném souboru může být libovolný počet deklarací neveřejných (package-local) tříd. Není to však hezké.

Lokální v balíku, `package-private`

- Přístupné jen ze tříd **stejného balíku**, používá se málo.

- Jsou-li podtřídy v jiném balíku, třída není přístupná!

```
package cz.some.pkg;

class Account {
// package-private class
// available only in cz.some.pkg
}
```

- Svazuje viditelnost s organizací do balíků (ta se může měnit častěji než např. vztah *nadtřída-podtřída*).
- Občasné využití, když nechceme mít konstruktor `private` nebo rozhraní `public`.

Chráněný, `protected`

Viditelnost `protected` , tj. přístupné jen z *podtříd* a *tříd stejného balíku*.

```
public class Account {
   // attribute can be protected (but it is better to have it private)
   protected float creditLimit;
}
```

- U *metod* tam, kde se nutně očekává použití z podtříd nebo překrývání.
- Vcelku často u *konstruktorů* — často se volá právě ze stejné (pod)třídy.

Shrnutí viditelnosti

Obvykle se řídíme následujícím:

metoda

- obvykle `public`, je-li užitečná i mimo třídu či balík
- `protected` je-li je vhodná k překrytí v případných podtřídách
- jinak `private`

atribut

- obvykle `private`
- výjimečně `protected`, je-li potřeba přímý přístup v podtřídě

třída

- obvykle `public`
- výjimečně `package-private` nebo `private`