Viditelnost (práva přístupu)

Tomáš Pitner, upravil Marek Šabo

Viditelnost

- Přístup ke *třídám* i jejim *prvkům* lze (podobně jako např. v C++) regulovat.
- Přístupem se rozumí *jakékoli použití* dané třídy, prvku… prostě už vůbec výskyt daného identifikátorů tím může být omezen.
- Omezení přístupu/viditelnosti je *kontrolováno hned při překladu* → není-li přístup povolen, nelze program ani přeložit.
- Tímto způsobem lze regulovat přístup/viditelnosti, mezi celými třídami, nikoli pro jednotlivé objekty!
- Nelze tedy pomocí viditelnosti zajistit, že "objekt Franta může volat metodu `transferTo` na objektu bankovního účtu, který mu patří, a nikdo jiný ne".
- Jiný způsob zabezpečení představuje tzv. *security manager*, který lze aktivovat při spuštění JVM.

Granularita viditelnosti/omezení přístupu

- Přístup/viditelnost je v Javě regulován/a — pro celou *třídu* nebo — *jednotlivě po prvcích* (metodách, atributech), nikoli jako v C++ po blocích prvků.
- Omezení přístupu/viditelnosti je určeno uvedením jednoho z tzv. *modifikátoru přístupu* nebo *neuvedením žádného*.

Přístup ke třídě vs. prvkům

- Nastavení viditelnosti pomocí modifikátoru k celé třídě omezuje viditelnost *příslušné třídy jako takové* (vůbec možnost použití jejího názvu) i jejich *statických vlastnosti* (atributů, metod).
- Dílčí vlastnosti uvnitř třídy pak mohou mít viditelnost stejnou nebo nižší.

Typy viditelnosti/přístupu

Existují čtyři možnosti:

- `public` = veřejný
- `protected` = chráněný
- *modifikátor neuveden* = říká se *lokální v balíku* (package local) nebo *chráněný v balíku* (package protected) evt. "přátelský" (friend)
- `private` = soukromý

Kde jsou která omezení aplikovatelná?

Třídy mohou být:

veřejné

`public`

neveřejné
: *lokální v balíku*

soukromé
: ale to jen u tzv. *vnořených tříd* (tříd uvnitř ve třídách), mimo rozsah předmětu PB162

Vlastnosti tříd = atributy a metody mohou být:

veřejné
: `public`

chráněné
: `protected`

neveřejné
: *lokální v balíku*

soukromé
: `private`

Veřejný (`public`)

- Viditelnost `public` , přístupné odevšad

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

- Třída `Account` je veřejná, tj. lze např.
- vytvořit objekt typu `Account` i v metodě jiné třídy;
- deklarovat podtřídu třídy `Account` ve stejném i jiném balíku.
- Ovšem ne všechny vlastnosti uvnitř `Account` musejí být veřejné (ale mohou).

Použití `public`

- Veřejné bývají obvykle některé *konstruktory* (nikoli nezbytně všechny) a některé *metody*.
- Veřejné jsou typicky metody předepsané implementovaným *rozhraním*.
- Atributy téměř *nikdy*, jedině *konstanty ano* (nikoli nezbytně).

Chráněný (`protected`)

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

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

Použití `protected`

- Pro metody (občas) a atributy (málokdy).
- U *metod* tam, kde se nutně očekává použití z podtříd nebo překrývání.
- Vcelku často u *konstruktorů*, protože konstruktor se často volá právě ze stejné třídy nebo její podtřídy.

Lokální v balíku

- Viditelnost *lokální v balíku* = *přátelský* → přístupné jen *ze tříd stejného balíku,* už ale ne z podtříd, jsou-li v jiném balíku.

```
public class Account {
   Date created; // attribute is package-local
}
```

Použití "lokálního v balíku"

- Používá se málo, a to spíše u atributů než metod, ale dost často se vyskytuje z lenosti programátora, kterému se nechce psát `protected` či `private`.
- Nelze moc doporučit, protože svazuje viditelnost/přístupová práva s organizací do balíků (→ a ta se může přece jen měnit častěji než např. vztah *nadtřída-podtřída*.)
- Mohlo by mít význam, je-li práce rozdělena tak, že na jednom balíku pracuje jen jeden vývojář či skupina — pak si může přátelským přístupem chránit své neveřejné prvky/třídy → nesmí ovšem nikdo jiný chtít jeho třídy rozšiřovat a používat přitom přátelské prvky.
- Používá se pro neveřejné třídy napsané v jednom zdrojovém souboru se třídou veřejnou, když už to tak chceme psát.

Soukromý (`private`)

- Viditelnost `private` (*soukromý*), tj. viditelné/přístupné jen v rámci třídy, ani v podtřídách — používá se častěji pro proměnné než metody
- Označením `private` prvek *zneviditelníme i případným podtřídám*.

```
public class Account {
   private String owner;
   ...
}
```

- Atribut `owner` je soukromý = nelze k němu přímo přistoupit ani v podtřídě — je tedy třeba zpřístupnit proměnnou pro "vnější" potřeby jinak, např.
- přístupovými metodami `setOwner(String owner)` a `String getOwner()`.
- Výhodou pak je, že skrýváme konkrétní implementaci datové položky, tzn. případně vůbec nemusí existovat jako proměnná, ale může se v případě potřeby (volání metody `get`) spočítat.

Použití `private`

atributy

téměř vždy `private` kromě konstant, které chceme vidět odjinud,

metody

kromě těch nabízených veřejně je taky vhodné použít `private`,

- U metod trochu pozor, někdy je lepší nechat `protected` kvůli možnému překrývání v podtřídách.
- Je třeba rozmyslet dopředu, jak (zda vůbec) se bude třída moci dědit.
- Volbou `private` nic zásadně nepokazíme, případně lze při pozdějších modifikacích změnit na `protected`.

Shrnutí viditelnosti

- Obvykle se řídíme následujícím:

  metoda

  by měla být `public`, je-li užitečná i mimo třídu či balík — "navenek", což je typické u metod předepsaných v rozhraní. Je metoda určená/vhodná k překrytí případných podtřídách? pak `protected`, jinak klidně `private`.

  atribut

  by měl být `private` a zcela výjimečně `protected` tehdy, je-li potřeba přímý přístup v podtřídě. Téměř nikdy bychom neměli deklarovat atributy jako `public` (vyjma případů, kdy jde o konstanty určené ke sdílení vně)!

Viditelnost a umístění deklarací do souborů

- Třídy deklarované jako *veřejné* (`public`) musí být umístěné do souborů s názvem totožným s názvem třídy (a přípona `.java`) i na systémech Windows (vč. velikosti písmen).

- Kromě takové třídy však může být v tomtéž souboru i libovolný počet deklarací neveřejných tříd (lokálních v balíku).