Balíky (packages)

Table of Contents

Organizace tříd do balíků . . . 1
Světově unikátní pojmenování balíků . . . 1
Příklad jména balíku . . . 2
Příklad třídy v balíku . . . 2
Plný název třídy vč. balíku . . . 2
Zápis třídy do zdrojového souboru . . . 2
Příslušnost třídy k balíku . . . 3
Pseudohierarchie balíků . . . 3
Příslušnost k balíkům — POZOR! . . . 3
Použití tříd v různých balících . . . 3
Odbočka: práva přístupu . . . 3
Příklad vzájemného použití tříd . . . 4
Příklad vzájemného použití tříd . . . 4
Deklarace `import` . . . 4
`import` sám nezajistí viditelnost . . . 5
`import` neviditelné třídy je k ničemu . . . 5
Deklarace `import` třídy . . . 5
Deklarace `import` celého balíku . . . 5
Balíky a moduly . . . 6

Organizace tříd do balíků

- Třídy zorganizujeme do balíků.
- V balíku jsou vždy umístěny *související* třídy.
- Co znamená *související*?
  - pracuje na nich **jeden tým**
  - jejich **objekty spolupracují**
  - jsou podobné **úrovni abstrakce**
  - jsou ze **stejné části reality**

Příklad: V balíku `geometry` jsou třídy reprezentující geometrické objekty (čtverec, trojúhelník, …).

Světově unikátní pojmenování balíků

- Aby se zabránilo kolizím (stejná jména pro různé třídy)

- konstruují se jména balíků jako pokud možno světově unikátní
- byla zvolena obdoba doménových internetových jmen (taky unikátní)

Příklad jména balíku

- `cz.muni.fi.pb112` je možné a vhodné jméno balíku
- je světově unikátní, protože `cz.muni.fi` je obrácené doménové jméno fakulty (`fi.muni.cz`)
- `pb112` je identifikátor, jehož jedinečnost už si v rámci organizace FI "uhlídáme"
- Pozor, jiné konvence mají balíky ve standardní vestavěné knihovně Java Core API (např. `java.util`)
- Občas jsou výjimky i jinde, např. používalo se `junit.framework`, i když to nebylo Java Core API.

Příklad třídy v balíku

```
package cz.muni.fi.pb112;
// class Person is in this package
public class Person {
   // attributes, methods
}
```

- Všechna písmena názvu balíku by měla být dle konvencí *malá*, tedy nikoli `Cz.Muni.Fi.PB112`.
- Takový krkolomný název by byl navíc různý od "hezkého" `cz.muni.fi.pb112`.
- Stejně tak raději žádná podtržítka v názvech.

Plný název třídy vč. balíku

- Na třídu v balíku se odvoláváme plným názvem `cz.muni.fi.pb112.Person`
- Pokud se odvoláváme na třídu ve stejném balíku (z jedné do druhé), pak stačí jen "holé" lokální jméno `Person`

Zápis třídy do zdrojového souboru

- Umístění do balíků souvisí s umístěním zdrojových souborů na disku
- Třída `Person` bude v souboru `Person.java`
- Tento soubor bude v adresáři `cz/muni/fi/pb112`
- Pozor na velká/malá písmena — v obsahu i názvu souboru i adresářů

Příslušnost třídy k balíku

- Deklarujeme ji syntaxí: `package název.balíku;`
- Uvádíme obvykle jako *první* deklaraci v zdrojovém souboru.
- Příslušnost k balíku musíme současně *potvrdit správným umístěním* zdrojového souboru do adresářové struktury.
- Neuvedeme-li příslušnost k balíku, stane se třída součástí tzv. *implicitního balíku* — prosím **nepoužívat!**
- Vždy tedy `package` uvádět!

Pseudohierarchie balíků

- Balíky obvykle organizujeme do jakýchsi pseudohierarchií, např.:
  - `cz.muni.fi.pb112`
  - `cz.muni.fi.pb112.banking`
  - `cz.muni.fi.pb112.banking.credit`

Příslušnost k balíkům — POZOR!

- Nicméně není to tak, že by např. třída `cz.muni.fi.pb112.banking.Account` byla současně v balíku `cz.muni.fi.pb112.banking` a taky třeba `cz.muni.fi.pb112`.
- Je-li třída v balíku `cz.muni.fi.pb112.banking`, je pouze v něm a žádném jiném.
- Není ani v `cz.muni.fi.pb112`, ani v `cz.muni.fi.pb112.banking.credit`.
- Prostě buď *je ve stejném* balíku nebo *je v jiném*.

Použití tříd v různých balících

- Balíky slouží k logickému rozčlenění kódu.
- Důsledkem je vzájemná "(ne)viditelnost" tříd.
- Velmi zjednodušeně: třídy v jednom balíku "mají k sobě blíž".
- Jsou-li v různých balících, vůbec o sobě nemusí vědět.

Odbočka: práva přístupu

- Kromě toho rozhoduje i to, jakou viditelnost (právo přístupu) použijeme.
- Pro tento účel slouží modifikátory `public`, `protected`, `private`.
- Objasníme později.

Příklad vzájemného použití tříd

- Třídy ve stejném balíku: snadné vzájemné použití

`Person` *v balíku* `cz.muni.fi.pb112`

```
package cz.muni.fi.pb112;
public class Person {
   // attributes, methods
}
```

`Account` *v tomtéž balíku*

```
package cz.muni.fi.pb112;
public class Account {
   private Person owner; // owner of this Account is a Person
}
```

Příklad vzájemného použití tříd

- Třídy v jiném balíku: nutné plné jméno
- Třída `Account` v jiném balíku `cz.muni.fi.pb112.banking`
- použije pro třídu `Person` její plný název balíku
- nebo lze použít `import` názvu třídy

```
package cz.muni.fi.pb112.banking;
public class Account {
   private cz.muni.fi.pb112.Person owner; // full package name
}
```

Deklarace `import`

- Nebo lze pro vzájemné odvolávání se pomocí deklarace `import`

```
package cz.muni.fi.pb112.banking;
import cz.muni.fi.pb112.Person;
public class Account {
   private Person owner; // class name imported above
}
```

`import` sám nezajistí viditelnost

Když bude Person neveřejná, nebude vidět do jiného balíku:

```
package cz.muni.fi.pb112;
... // in file cz/muni/fi/pb112/Person.java
class Person { // package-local class
   private Person owner; // owner of this Account is a Person
}
```

`import` neviditelné třídy je k ničemu

Následující soubor `Account.java` se nezkompiluje:

```
... // in file cz/muni/fi/pb112/banking/Account.java
package cz.muni.fi.pb112.banking;
import cz.muni.fi.pb112.Person;
public class Account {
   // class Person imported but invisible from here
   private Person owner;
}
```

Deklarace `import` třídy

- Příklad `import cz.muni.fi.pb112.Person;`
- Umožní použít identifikátor třídy (v našem případe `Person`) v rámci jiné třídy.
- Píšeme obvykle ihned po deklaraci příslušnosti k balíku (`package názevbalíku;`).
- Import není nutné deklarovat mezi třídami téhož balíku!

Deklarace `import` celého balíku

- Import **všech tříd** z balíku provedeme např. `import cz.muni.fi.pb112.*;`
- Doporučuje se "import s hvězdičkou" nepoužívat vůbec; máme-li více `*` importů:
  - Problém 1: a obojí z nich obsahují třídu `Person`, která z nich se použije? (nepoužije se žádná, dostanete kompilační chybu)
  - Problém 2: nepoznáme na první pohled, ze kterého balíku identifikátor pochází
  - Řešení: *Nebudeme to používat*!
- Pozn.: Hvězdičkou **nezpřístupníme** třídy z "podbalíků" — např. `import cz.*` nezpřístupní třídu `cz.muni.fi.pb112.Person`.

Balíky a moduly

- Jelikož větší projekty jsou postaveny z více balíků (někdy i desítek balíků a stovek tříd), potřebujeme větší celky.
- Od Javy 9 se nabízí možnost organizovat balíky do *modulů*.
- Kromě tříd v balících mohou moduly obsahovat i další "resources", jako jsou konfigurace, obrázky, data…
- Probereme později nebo jinde :)
- A Guide to Java Modularity