

Balíky (packages)

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 **stejně 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.pb162` je možné a vhodné jméno balíku
- je světově unikátní, protože `cz.muni.fi` je obrácené doménové jméno fakulty (`fi.muni.cz`)
- `pb162` 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.pb162;  
// class Person is in this package  
public class Person {  
    // attributes, methods  
}
```



Všechna písmena názvu balíku by měla být dle konvencí *malá*, tedy nikoli

Plný název třídy vč. balíku

- Na třídu v balíku se odvoláváme plným názvem `cz.muni.fi.pb162.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/pb162`
- 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!**

(Pseudo)hierarchie balíků

- Balíky obvykle organizujeme do jakýchsi pseudohierarchií, např.:
 - `cz.muni.fi.pb162`
 - `cz.muni.fi.pb162.banking`
 - `cz.muni.fi.pb162.banking.credit`
- Nicméně není to tak, že by např. třída `cz.muni.fi.pb162.banking.Account` byla současně v balíku `cz.muni.fi.pb162.banking` a taky třeba `cz.muni.fi.pb162`.
- Je-li třída v balíku `cz.muni.fi.pb162.banking`, je pouze v něm a žádném jiném.
- Není ani v `cz.muni.fi.pb162`, ani v `cz.muni.fi.pb162.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

Třída `Person` v balíku `cz.muni.fi.pb162`

```
package cz.muni.fi.pb162;  
public class Person {  
    // attributes, methods  
}
```

Třída `Account` v tomtéž balíku

```
package cz.muni.fi.pb162;  
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.pb162.banking`
- použije pro třídu `Person` její plný název balíku
- nebo lze použít `import` názvu třídy

```
package cz.muni.fi.pb162.banking;  
public class Account {
```

```
private cz.muni.fi.pb162.Person owner; // full package name
}
```

Deklarace `import`

- Nebo lze pro vzájemné odvolávání pomocí deklaráce `import`

```
package cz.muni.fi.pb162.banking;
import cz.muni.fi.pb162.Person;
public class Account {
    private Person owner; // class name imported above
}
```

`import` sám nezajistí viditelnost

- Když bude `Person` neveřejná, nebude vidět do jiného balíku
- `import` je pak k ničemu
- následující soubor `Account.java` se nezkompiluje:

```
package cz.muni.fi.pb162;
... // in file cz/muni/fi/pb162/Person.java
class Person { // package-local class
    private Person owner; // owner of this Account is a Person
}
... // in file cz/muni/fi/pb162/banking/Account.java
package cz.muni.fi.pb162.banking;
import cz.muni.fi.pb162.Person;
public class Account {
    // class Person imported but invisible from here
    private Person owner;
}
```

Deklarace `import` třídy

- Příklad `import cz.muni.fi.pb162.Person;`
- Umožní použít identifikátor třídy (v našem případě `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.pb162.*;`
- 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.*` nepřístupní třídu `cz.muni.fi.pb162.Person`.