Java efektivně

Lukáš Zapletal
liberix.cz

Pokročilejší techniky programování v Javě

Java efektivně

Tato prezentace vychází
kompletně z knihy
J. Blocha: **Effective Java** (A-W 2001)

u nás

Java efektivně - 57 zásad
softwarového experta (Grada 2002)

Java efektivně - generické typy

- anotace nejsou šablonami z C++
- podobnost s C#
- Příklad:

```
Map<String, User> = usersMap
   new HashMap<String, User>();
...
User user = usersMap.get(“uzivatel232”);
...
usersMap.put(“uzivatel232”, user);
```

Java efektivně - generické typy

- můžete je použít jak u třídy, tak u metody

```
public static <E> void render(
    Map<String, E> set) {
```

- E e = set.get(“header”);
- ...
- renderer.process(e);
- ...
- }
- volání: object.<Page>render(pagesMap);

Java efektivně - generické typy

```
class Stack<GO extends GraphicsObject> {
 public void push(GO) {
 ...
 }
 public GO pop() {
 ...
 }
}
```

Java efektivně - generické typy

- použití:
  Stack<Line> stk = new Stack<Line>();
  ...
  Line line = stk.pop();

- u GT nemůžete použít primitivní typy
- generické typy si zaslouží více studia (očišťování, zakázané operace, žolíky, kolize...)
- dobrá kniha: Pecinovský: Java 5.0 - novinky

Java efektivně

- používejte jedináčka, když je to třeba
- vytvořte private konstruktor, když je třeba
- pozor na finalize
- šetřte vytvářením nových instancí
- když to jde, použijte tovární metody
- nepotřebné objekty odstraňujte nastavením na **null**

Java efektivně

- dobře se seznamte s Java API, zejména **kolekcemi** a utilitami pro kolekce (nakl. KOPP: P. Herout: Bohatství knihoven)
- překrýváte-li equals, **vždy** překryjte i hashCode
- **equals** pište se zvýšenou pozorností
  - reflexivnost: x.equals(x)
  - symetričnost: y.equals(x) == x.equals(y)
  - tranzitivita: (pokud x=y a y=z, pak x=z)
  - pokud se objekt nemění, nemění se výsledek eq.
  - žádný objekt se nesmí rovnat null

Java efektivně

```
1 boolean equals(Object o) {
2  if (this == o) return true;
3  if (o instanceof MojeTrida == false)
4   return false;
5  MojeTrida t = (MojeTrida) o;
6  return this.a == t.a && this.b == t.b && ... ;
7 }
```

ř. 2 - optimalizace
ř. 3 - pokud to není náš typ, nemůže nastat shoda
ř. 5 - nesmíme vyhodit ClassCastException
ř. 6 - postupně porovnáváme všechny členské proměnné
(resp. ty, které se nastavují - ale nevypočítávají z jiných)

Vždy porovnávejte nejdříve ty, u kterých je nejpraděpodobnější, že nedojde ke shodě.

Java efektivně - anotace

- znáte z C#
- v Javě je několik základních anotací
  - @Deprecated
  - @SupressWarnings
  - @Override (a další)
- anotace se často využívají u dalších knihoven, které je rozšiřují (@Entity, @Test ...)

Java efektivně

- hashCode musí počítat návratovou hodnotu ze všech prvků, které identifikují objekt (id např.)
- hashCode vrací celé číslo (int)
- nebojte se překrývat metodu toString()
- raději nepoužívejte metodu clone() a nahraďte tuto metodu kopírovacím konstruktorem
  - Object.clone() vrací mělkou kopii objektu, pakliže objekt implementuje Cloneable
  - musíte znát spoustu fint, jak to správně udělat
  - KK je daleko jednodušší

Java efektivně

- neměnitelná třída (invariant) - třída, jejíž instance nelze měnit; všechny informace se poskytnou v konstruktoru a pak jsou neměnné
  - nemá mutátory
  - nemělo by jít zbytečně překrývat metody
  - všechny atributy jsou soukromé a finální
  - všechny odkazy na jiné objekty by měly být také neměnné (můžete udělat defenzivní kopii)

Java efektivně

- neměnné třídy mají velké výhody:
  - jsou jednoduché a dobře se na nich staví
  - jsou zabezpečené z hlediska vláken
  - můžete je sdílet (jedna instance pro celý program)
- ale i nevýhody
  - každá změna = nová instance
  - je nutno vytváření optimalizovat (například továrními metodami)

Java efektivně

- “**Třídy by měly být neměnitelné, pokud neexistuje dobrý důvod k tomu, učinit je měnitelnými.**” Joshua Bloch, Effective Java
- třídy navrhujte jako neměnitelné
- až se ukáže, že to nejde, ještě jednou se zamyslete a teprve pak je udělejte měnitelnými
- když už musí být měnitelné, omezte měnitelnost na co největší možnou míru
- příklady: PhoneNumber, Numeric, BigDecimal

Java efektivně

- dávejte přednost kompozici před dědičností
- dědění buď povolte a zdokumentujte, nebo zakažte (final)
- preferujte rozhraní namísto abstraktní třídy
- nebojte se statických metod (Collections.sort)
- **vždy** kontrolujte vstupní parametry každé metody a vyvolávejte IllegalArgumentException, NullPointerException a IndexOutOfBoundsException (a jiné)

Java efektivně

- vytvářejte **defenzivní kopie**, když je to možné
- nevěřte těm, kteří používají vaše třídy
- vyhazujte výjimky namísto návratu a testování hodnoty **null**
- vyhýbejte se typům float a double, pokud potřebujete počítat přesně
  - například na počítačovou grafiku stačí
  - počítání peněz - použijte třídu BigDecimal

Java efektivně

- pozor na spojování řetězců (něco kompilátor odhalí, ale ne vše) - používejte StringBuffer
- odkazujte se na objekty jejich rozhraními
- reflexi používejte s rozvahou
- nebojte se vytvářet vlastní třídy výjimek
- kontrolované výjimky používejte u zotavitelných stavů a výjimky za běhu k fatálním chybám
- dávejte přednost standardním výjimkám

Java efektivně

- **nikdy** neignorujte výjimky: catch (Výjimka) { }
- pokud objekt selže (vyhodí výjimku), snažte se docílit toho, aby byl stav objektu jako před vyvoláním výjimky (objekt byl dále použitelný)
- do výjimky napište přesné důvody selhání
- nebojte se zapozdřených výjimek
- výjimky dokumentujte
- kontrolované výjimky jsou silnou součástí Javy

Java efektivně

- pozor na vlákna u měnitelných objektů
- kdykoli více vláken sdílí měnitelná data, pak musí každé vlákno při čtení i zápisu dostat zámek (musíme synchronizovat)
- v synchronizačních blocích musíte vykonávat co nejméně operací - hrozí deadlock
- dokumentujte zabezpečení vláken
- seznamte se s metodami wait, notify, notifyAll a s třídou Thread a jí podobným

Java efektivně

- to jsou jen hlavní body této skvělé knížky (některá témata například Serializace jsem vynechal)
- a samozřejmě klasické:
  - dobře volte pojmenování metod, tříd, proměnných
  - používejte zažité konvence
  - optimalizujte rozumně
  - dokumentujte