Rozhraní. Dědičnost. Dokumentace kódu.

August 28, 2014

1 Rozhraní – úvod

1.1 Co je rozhraní

- Rozhraní je vlastně *popis (specifikace) množiny vlastností*, *aniž* bychom tyto vlastnosti *ihned implementovali*. Vlastnostmi zde rozumíme především *metody*.
- Říkáme, že určitá třída *implementuje rozhraní*, pokud implementuje (tedy *má* - přímo sama nebo podědí) všechny vlastnosti (tj. metody), které jsou daným rozhraním předepsány.
- Javové rozhraní je tedy *množina hlaviček metod označená identifikátorem* - názvem rozhraní - a celých specifikací, tj. popisem, co přesně má metoda dělat - vstupy/výstupy metody, její vedlejší efekty...)

1.2 Deklarace rozhraní

- Vypadá i umisťuje se do souborů podobně jako deklarace třídy
- Všechny metody v rozhraní musí být `public` a v jejich hlavičce se to ani nemusí uvádět.
- Těla metod v deklaraci rozhraní se nepíší vůbec, ani prázdné závorky.

Příklad deklarace rozhraní

```
public interface Informing {
   void writeInfo();
}
```

1.3 Implementace rozhraní

Příklad

```
public class Person implements Informing {
   ...
   public void writeInfo() {
      ...
   }
}
```

Čteme: Třída `Person` implementuje rozhraní `Informing`.

1. Třída v hlavičce uvede `implements NázevRozhraní`
2. Třída implementuje všechny metody předepsané rozhraním

1.4 Využití rozhraní

1. Potřebujeme-li u jisté proměnné právě jen funkcionalitu popsanou určitým rozhraním,
2. tuto proměnnou můžeme pak deklarovat jako typu rozhraní - ne přímo třídy, která rozhraní implementuje.

Příklad

```
Informing petr = new Person("Petr Novák", 1945);
petr.writeInfo(); // "petr" stačí deklarovat jen jako Informing
                  // jiné metody než předepsané tímto intf.
 // nepotřebujeme!
```

1.5 Dvě třídy implementující totéž rozhraní

Totéž rozhraní může implementovat více tříd, často konceptuálně zcela nesouvisejících:

- Rozhraní `Going` ("jdoucí") implementují dvě třídy:
- `Car` (auto má schopnost "jít", tedy jet)
- `Clock` (hodiny také "jdou")
- `Person` (člověk také může chodit)

Viz příklad - projekt v BlueJ - `car_clock`

1.6 Implementace více rozhraní

V Javě, na rozdíl od C++ neexistuje vícenásobná dědičnost, třída má tedy *nejvýše jednoho předka* (kromě třídy `Object` má právě jednoho).

- to nám ušetří řadu komplikací (dědění dvěma/více cestami)
- ale je třeba to něčím nahradit

Pokud po třídě chceme, aby disponovala vlastnostmi z několika různých množin (skupin), můžeme ji deklarovat tak, že

- implementuje více rozhraní

2 Dědičnost

2.1 Dědičnost

V realitě jsme často svědci toho, že třídy jsou **podtřídami** jiných:

- tj. všechny objekty podtřídy jsou zároveň objekty nadtřídy, např. každý objekt typu (třídy) `ChovatelPsu` je současně typu `Clovek` nebo
- např. každý objekt typu (třídy) `Pes` je současně typu `DomaciZvire` (alespoň v našem výseku reality neexistují i psi "nedomácí"...)

Podtřída je tedy "zjemněním" nadtřídy:

- přebírá její vlastnosti a zpravidla přidává další, **rozšiřuje** svou nadtřídu/předka

V Javě je *každá* uživatelem definovaná třída potomkem nějaké jiné – neuvedeme-li předka explicitně, je předkem vestavěná třída `Object`

2.2 Terminologie dědičnosti

- Nadtřídě (superclass) se také říká "(bezprostřední) předek", "rodičovská třída"
- Podtřídě (subclass) se také říká "(bezprostřední) potomek", "dceřinná třída"

Dědění může mít i více "generací", např.`Person` – `Employee` – `Manager` (osoba je rodičovskou třídou zaměstnance, ten je rodičovskou třídou manažera)Přeneseně tedy předkem (nikoli bezprostředním) manažera je člověk.

2.3 Jak zapisujeme dědění

Klíčovým slovem `extends` :

```
public class Employee extends Person {
   // ... popis vlastností (proměnných, metod...)
   // zaměstnance navíc oproti (obecnému) člověku...
}
```

2.4 Dědičnost a vlastnosti tříd (1)

Jak víme, třídy popisují skupiny objektů podobných vlastnostíTřídy mohou mít tyto skupiny **vlastností**:

- Metody - procedury/funkce, které pracují (především) s objekty této třídy
- Proměnné - pojmenované datové prvky (hodnoty) uchovávané v každém objektu této třídy

Vlastnosti jsou ve třídě "schované", tzv. **zapouzdřené** (encapsulated)

2.5 Dědičnost a vlastnosti tříd (2)

Třída připomíná záznam (record) známý např. z Pascalu (nebo struct z C), ty však zapouzdřují jen proměnné, nikoli metody.Dědičnost (alespoň v javovém smyslu) znamená, že dceřinná třída (podtřída, potomek)

- má *všechny* vlastnosti (metody, proměnné) nadtřídy
- + přidává vlastnosti uvedené přímo v deklaraci podtřídy

2.6 Příklad

Cíl: vylepšit třídu `Account`Postup:

1. Zdokonalíme náš příklad s účtem tak, aby si účet "hlídal", kolik se z něj převádí peněz
2. Zdokonalenou verzi třídy `Account` nazveme `CreditAccount`

Vzorový zdroják samotný nepůjde přeložit, protože nemáme třídu, na níž závisí.

2.7 Příklad – zdrojový kód

```
public class CreditAccount extends Account {
   // private double balance; zdědí se z nadtřídy "Account"
   // zde pamatuji navíc, kolik mohu "jít do mínusu"
   private double creditLimit;

   public void add(double amount) {
      if (balance + creditLimit + amount >= 0) {
         // přes "super" zavoláme původní "neopatrnou" metodu
         super.add(amount);
      } else {
         System.err.println("Nelze odebrat částku " + (-amount));
      }
   }
   // writeInfo(), transferTo(Account to, double amount) se zdědí
}
```

2.8 Příklad – co tam bylo nového

- Klíčové slovo `extends` - značí, že třída `CreditAccount` je potomkem/podtřídou/rozšířením/dceřinnou třídou (*subclass*) třídy `Account`.
- Konstrukce `super.metoda(...);` značí, že je volána metoda rodičovské třídy/předka/nadtřídy (*superclass*). *Kdyby se nevolala překrytá metoda,* ***`super`*** *by se neuvádělo.*
- Větvení `if() ... else ...` - složené závorky se používají k uzavření příkazů do posloupnosti/sekvence.

2.9 Další příklad

Demoprojekt `private_account`:

- výchozí třída `Account`
- podědíme do třídy `PrivateAccount` (osobní/privátní účet)
- zde přibude nová vlastnost - proměnná "vlastník" nesoucí odkaz na osobu vlastnící tento účet.

2.10 Do třetice - víceúrovňová dědičnost

Neplést s vícenásobnou, ta v Javě není možná. Více úrovněmi myslíme častou situaci, kdy ze třídy odvodíme podtřídu, z ní zase podtřídu...Demoprojekt `checked_private_account`:

- výchozí třída `Account` (obyčejný účet)
- podědíme do třídy `PrivateAccount` (osobní/privátní účet)
- z ní podědíme do třídy `CheckedPrivateAccount` (osobní účet s kontrolou minimálního zůstatku)