Polymorfismus

Polymorfismus

Obecně máme několik typů polymorfismu (Polymorfismus (Wikipedia)):

- Ad-hoc polymorfismus
- Polymorfismus přetěžování operátorů
- Parametrický polymorfismus
- Podtypový polymorfismus

Ad-hoc polymorfismus

- objektům odvozeným z *různých tříd* volat tutéž metodu se stejným významem
- v Javě realizovatelné pomocí rozhraní
- např. třídy `Dog` i `Car` mohou implementovat rozhraní `Registered`

Polymorfismus přetěžování operátorů

- provedení rozdílné operace v závislosti na typu operandů
- například `+` se může chovat jinak na čísla, řetězce nebo matice
- v Javě není možné definovat vlastní přetěžování operátorů
- nicméně polymorfní operátory existují, typicky `+` (čísla, řetězce)
- nebo operátor podmíněného výrazu: `cond ? expr1 : expr2`
- ale například v C++ lze dodefinovat chování operátoru `+` tak, že bude sčítat matice

Parametrický polymorfismus

- V Javě realizován *parametrickými typy*.
- Například kolekce jako `List` nebo `Set` jsou realizovány jako *generické typy* (generics).
- Tyto se dají typově parametrizovat a tak mít například seznam osob `List<Person>`.

Podtypový polymorfismus

- umožněn děděním mezi objektovými třídami
- například `Employee`, `Manager` nebo `Student` jsou všechno osoby `Person` a mají tedy jméno `getName()`

Nevhodně realizovaný polymorfismus

- Typicky metoda, která si poradí se vstupem různého typu.
- Příklad `getPerimeter(Shape shape)` fungující pro různé (pod)typy `Shape`
- Nicméně má to jen nevýhody; lepší by bylo, kdyby každý typ měl svou nestatickou metodu `getPerimeter`

Příklad pseudo-polymorfismu

```
---
public static double getPerimeter(Shape shape) throws IllegalArgumentException {
    if (shape instanceof Rectangle r) {
        return 2 * r.length() + 2 * r.width();
    } else if (shape instanceof Circle c) {
        return 2 * c.radius() * Math.PI;
    } else {
        throw new IllegalArgumentException("Unrecognized shape");
    }
}
---
```

Polymorfismus pomocí vzorů ve výrazu `switch`

- Podrobněji v Pattern Matching for switch Expressions and Statements
- V Javě 17+ to polymorfní větvení lze napsat přece jen elegantněji: výrazem `switch` se vzory

```
---
public static double getPerimeter(Shape shape) throws IllegalArgumentException {
    return switch (shape) {
        case Rectangle r -> 2 * r.length() + 2 * r.width();
        case Circle c    -> 2 * c.radius() * Math.PI;
        default          -> throw new IllegalArgumentException("Unrecognized shape");
    };
}
---
```

Polymorfismus pomocí vzorů v příkazu `switch`

- Varianta téhož s pomocí *příkazu* `switch` a nikoli výrazu:

```
---
public static double getPerimeter(Shape shape) throws IllegalArgumentException {
    switch (shape) {
        case Rectangle r -> return 2 * r.length() + 2 * r.width();
        case Circle c    -> return 2 * c.radius() * Math.PI;
        default          -> throw new IllegalArgumentException("Unrecognized
shape");
    };
}
---
```

Typová bezpečnost vzorů ve `switch`

- V obou variantách - výraz i příkaz switch se vzory - musí být vzory ve `switch` *exhaustivní*,
- tzn. musí pokrývat všechny typy řídicího výrazu.
- Jednoduše řešitelné pomocí `default` na konci.