

Java 1 • lekce 4

Filip Jirsák

28.2.2023

online

Algoritmizace

Algoritmus je postup na vyřešení nějakého problému.

- Klasický příklad algoritmu je recept (třeba na svíčkovou).
- Algoritmus je obvykle univerzální, neřeší jeden konkrétní problém, ale skupinu podobných problémů:
 - ne *uvař svíčkovou pro 5 lidí*,
 - ale *uvař svíčkovou pro x lidí*.

Algoritmy 1

Existuje obrovské množství už hotových či popsaných algoritmů.

- Spousta už jich je implementovaná v Javě, ať už ve formě knihoven, nebo jsou přímo součástí standardní knihovny Javy.
- Nebo se dají najít popisy na internetu, třeba na Wikipedii.
- Příklady algoritmů: řazení prvků v seznamu podle abecedy, vyhledání v seřazeném seznamu, výpočet obvodu kruhu z poloměru, nakreslení prasátka pomocí želvy Žofky...

Algoritmy 2

Algoritmy pro počítače jsou často jiné, než algoritmy pro lidi. Počítače umí jen jednoduché úkoly, za to je umí provádět velmi rychle.

- Například hledání v seřazeném seznamu (např. v rejstříku knihy).
- Animace různých řadicích algoritmů: <https://www.toptal.com/developers/sorting-algorithms>
- Porovnání 24 řadicích algoritmů na YouTube: <https://www.youtube.com/watch?v=BeoCbJPuvSE>

Optimalizace

Jednotlivé algoritmy řešící stejnou věc se liší v tom, jak jsou výpočetně (časově) náročné a kolik paměti potřebují. **Optimalizace** je úprava algoritmu, aby byl rychlejší nebo potřeboval méně paměti.

- Jedna z častých chyb – předčasná optimalizace.
- Důležitější je, aby byl kód čitelný, než aby byl optimální.
- Počítače jsou velmi rychlé, takže optimalizace se řeší až tehdy, když se ukáže, že je někde problém.
 - Pak se nejprve musí zjistit, v čem přesně problém spočívá.
- I to má své hranice – nepíšeme kód zjevně „hloupě“, když existuje efektivnější a srozumitelné řešení (viz třeba „bogosort“).
- Efektivita se označuje notací *velké O* (*big O*), např. $O(n)$, $O(n^2)$, $O(n \log n)$
- Během kurzu se optimalizací nebudeme zabývat – ale je dobré vědět, že něco takového existuje.

Návratová hodnota metody ``return``

Metoda může vrátet nějakou hodnotu nebo objekt jako výsledek volání.

- Parametry metody = vstup, návratová hodnota = výstup.
- Vrací se pouze jedna hodnota nebo objekt (na rozdíl od parametrů, kterých může být víc).
- Typ vrácené hodnoty se deklaruje před názvem metody.
 - Pokud metoda nevrací nic, místo typu je uvedeno klíčové slovo ``void``.
- Vrácení hodnoty zajišťuje klíčové slovo ``return``.
- ``return`` musí pokrýt všechny větve kódu.
- ``return`` ukončí provádění metody.
- Metody, které nic nevrací („typ“ ``void``), mohou použít ``return`` pro předčasné ukončení metody.

`return` 1

Jednoduché vrácení hodnoty:

```
public String dejPozdrav() {  
    return "Ahoj";  
}
```

Vrácení hodnoty z různých větví kódu:

```
// Pouze pro ilustraci, takhle se to nedělá!  
public int pocetDnuVMesici(int mesic) {  
    if (mesic == 2) { // 2=únor  
        return 28;  
    }  
    if (mesic == 4 || mesic == 6 || mesic == 9 || mesic == 11) { //4=duben, 6=červen, 9=září, 11=listopad  
        return 30;  
    }  
    return 31;  
}
```

`return` 2

Předčasné *vyskočení* z metody, která nic nevrací:

```
// Pouze pro ilustraci, takhle se to nedělá!
public void vypisZkratkuPracovníhoDne(int den) {
    if (den == 6 || den == 7) { // 6=sobota, 7=neděle
        return;
    }
    if (den == 1) {
        System.out.println("Po");
    } else if (den == 2) {
        System.out.println("Út");
    } else if (den == 3) {
        System.out.println("St");
    } else if (den == 4) {
        System.out.println("Čt");
    } else if (den == 5) {
        System.out.println("Pá");
    }
}
```


Cvičení – kočka a myš

<https://github.com/FilipJirsak-Czechitas/j1-lekce04>
