Metody v Javě

Metody jsou podprogramy, které manipulují s členskými proměnnými, čímž mění stav objektu. Metody tedy popisují **schopnosti** (“dovednosti”, vlastnosti, …) objektu.

Metody se dělí na **dva typy**:

1. **Metody třídy** (též označované jako **statické metody**)
2. **Metody instance**

Parametry se rovněž dělí na **dva typy:**

1. **Skutečné parametry** - ty, které posíláme do metody při jejím volání
2. **Formální parametry** - ty, se kterými byla metoda deklarována

Nepsaná pravidla při tvoření metod

* je vhodné psát takové metody, které provádějí **jednu** a **pouze jednu** jasně specifikovanou **činnost**
* název metody by mělo být **krátké**, **výstižné** a v celém programu tvořené podle **jednotného schématu**, např. otevriSoubor(), otevriOkno()....
* název metody by měl začínat malým písmenem a splňovat tzv. **CamelCase** notaci (každé následující slovo má první písmeno velké např. nactiStudentyZeSouboru())

Deklarace metody

Zahrnuje jak hlavičku metody, tj. jméno metody, typ návratové hodnoty a případné typy a jména jejích formálních parametrů, tak i její tělo, ve kterém je uložen výkonný kód metody.

* Tělo metody je uzavřeno do složených závorek { a }, může obsahovat jak příkazy, tak i deklarace proměnných
* Výstupní hodnota metody se předává příkazem:

return (výraz);

, který vypočte hodnotu výrazu výraz, přiřadí ji jako návratovou hodnotu metody a tuto metodu ukončí

Příklad metody, která vrátí větší ze svých dvou parametrů:

static int max(int a, int b) {

if (a > b) {

return a;

} else {

return b;

}

}

🛈 V Javě není nutné uvádět funkční prototypy, aby překladač věděl, jakého návratového typu je metoda. V Javě je lhostejné, zda deklarace metody leží před nebo za místem volání metody. To je sice možné i v C, ale je tam vyžadováno dodatečné úsilí v podobě funkčních prototypů např. int max(int a, int b); nad funkcí main()

Metoda bez parametrů

Metoda , která nemá žádné parametry, musí být deklarována i volána včetně obou kulatých závorek. Například:

int secti() {

int a, b;

Scanner sc = new Scanner(System.in);

a = sc.nextInt();

b = sc.nextInt();

return a + b;

}

🛈 Zapomenete-li uvést prázdné kulaté závorky, např.: j = secti; vypíše překladač chybu: variable secti not found in class..

🛈 Není možné uvést místo neexistujících formálních parametrů klíčové slovo void, jako v C, např.: int secti(void) {, Překladač hlásí chybu: <identifier> expected

Metoda bez návratového typu - Procedura

Pro tyto případy se používá návratový typ void (tj. prázdný). Metoda se pak deklaruje, jako metoda vracející typ void , např.:

void tiskPenez(int koruny) {

System.out.println(“Cena: “ + koruny + “ Kc”);

}

volání je pak: tiskPenez(a + b)

nebo: tiskPenez(10)

nebo: tiskPenez(a)

🛈 Příkaz return u procedur není nutný. Pokud není uveden, nahrazuje ho uzavírací závorka } metody.

🛈 Příkaz return se pak používá pro nucené ukončení metody před dosažením jejího konce, např. po nějaké podmínce

Procedura bez parametrů

Je nejjednodušším typem metody a používá se typicky zejména pro nejrůznější tisky. Vypadá takto:

void tisk() {

System.out.println(“ahoj”);

}

Metoda s více parametry různých typů

Potřebuje-li metoda větší množství formálních parametrů, vypisují se jednotlivě včetně svých typů a oddělují se čárkami. V deklaraci metody si můžeme zvolit pořadí parametrů libovolně, při volání toto pořadí dodržet musíme. Například

double secti(int a, double b) {

return a + b;

}

Rekurzivní metody

Java umožňuje vytvářet rekurzivní metody. Hlavička těchto metod se nijak neliší od hlavičky metody nerekurzivní. Rozdíl je pouze v těle metody, kde rekurzivní metoda volá sama sebe.

Příklad metody pro výpočet faktoriálu:

long fakt(long n) {

if (n > 1) {

return n \* fakt(n - 1);

} else {

return 1;

}

}

Způsoby předávání skutečných parametrů metodám

Java umožňuje pouze jeden způsob **předávání parametrů** a to **hodnotou** (call-by-value). To znamená, že skutečné parametry mohou být v metodě pouze **čteny**. Lze je sice pomocí přiřazovacího příkazu změnit, ale pouze dočasně, protože se ve skutečnosti mění jen lokální kopie.

🛈 Možnost **volání odkazem** Java neřeší pomocí pointerů jako je tomu u jazyka C. Problém je řešen s využitím prostředků orientovaného programování např. pomocí getterů a setterů

🛈 Stejně tak Java neumožňuje využívat metod s proměnným počtem parametrů. Tento handicap se řeší pomocí přetížení metod

Přetížené metody

Jsou takové metody, které mají stejná jména, ale různé hlavičky. To, že je metoda přetížená, znamená, že se musí její formální parametry lišit **počtem** nebo **typem** nebo **pořadím**, případně kombinacemi těchto způsobů.

Metodu nelze přetížit pouhou změnou typu návratové hodnoty - je nutné to zařídit minimálně jedním ze tří výše uvedených způsobů.

int ctverec(int i) { return i \* i; }

double ctverec(double i) { return i \* i; }

🛈 Je jedno, zda je deklarace volané metody před místem volání nebo za ním. Jediné omezení je v tom, že každá metoda musí patřit do nějaké třídy.

Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky