Členění kódu

- Deklarace metod
- Statické metody třídy
- Parametry metod
- Volání metody
- Formální a skutečné parametry
- Knihovní třída

SHRNUTÍ PŘEDCHOZÍCH TÉMAT

Program v jazyce Java – opakování

- Program v jazyce Java – třídy
- **Třídy** umístěné v **balících** – příslušnost do balíku
- Třída má **jméno** a **tělo**
  - Jméno – identifikátor – začínat velkým písmenem, další slova začínat velkým písmenem
  - Tělo obsahuje deklarace atributů a metod
- **Atributy** – členské proměnné – uchování dat
- **Metoda** obsahuje programový kód (výkonný kód, příkazy), příkazy se postupně provedou po zavolání metody
- **Tělo metody**
  - Příkazy
    - Příkazy jazyka Java
  - Příkazy se postupně provedou po volání metody

```
public class <jméno třídy> {

  // schéma deklarace atributu/proměnné
 <modifikátory> <typ> <jméno>;

  // schéma metody
  <modifikátory> <typ> <jméno>
   (<deklarace parametrů>) {
      <příkazy>
  }


}
```

METODY

Deklarace metody

- Deklarace metody

```
<modifikátory> <typ> <jméno> ( <formální_parametry>) {
        <tělo_metody>
}
```

- Modifikátory – `static, public, final, private, protected …`
- Typ – buď typ hodnoty, kterou bude metoda vracet, nebo klíčové slovo `void`, pokud metoda hodnotu nevrací
- Jméno – identifikátor metody
- Formální parametry
  - Deklarace formálních parametrů, jejichž hodnota bude používána při výpočtu – viz dále
- Jméno metody a parametry – **signatura metody** – různé metody různá signatura
- Tělo metody
  - Příkazový blok – tělo metody je tvořeno příkazy, které se postupně provedou po volání metody
  - Konstrukce pro zápis příkazu v těle metody – viz předchozí přednáška
  - Příkaz `return` – okamžité ukončení vykonávání metody – příkaz se v těle metody může vyskytnout vícekrát (v různých větvích, ukončení metody s různou hodnotou v různých případech)

    Metoda **konkrétního typu** – ukončení – `return <hodnota>;` – hodnota je vrácena jménem metody, hodnota musí být kompatibilní vzhledem k přiřazení s typem metody; pokud má metoda určený typ, musí poskytovat/vracet hodnotu určeného typu, tj. musí končit příkazem return <hodnota>

    Metoda **nevracející hodnotu** – ukončení – `return;` – metoda nevracející hodnotu nemusí příkaz `return` obsahovat, v takovém případě je běh metody po vykonání příkazů v těle metody ukončen

Ukázka členění kódu

```
public class VypoctyObrazcu {

    public static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        boolean konec = false;
        do {
            vypisMenu();
            int volba = nactiVolbu();
            // ...
        } while (!konec);
    }

    private static void vypisMenu() {
        System.out.println("1. Vypocet ctverce");
        System.out.println("2. Vypocet obdelnika");
        System.out.println("3. Vypocet kruhu");
        System.out.println("4. Vypocet trojuhelnika");
        System.out.println("0. Konec programu");
    }

    private static int nactiVolbu() {
        System.out.print("Zadej volbu ");
        return sc.nextInt();
    }
}
```

Ukázka členění kódu

```
public class VypoctyCelaCisla {

    public static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        boolean konec = false;
        while (!konec) {
            vypisMenu();
            int volba = nactiVolbu();
            // ...
        }
    }

   private static void vypisMenu() {
        System.out.println("1. Vypis delitelu cisla");
        System.out.println("2. Test prvociselnosti cisla");
        System.out.println("3. Rozklad cisla na  soucin prvocisel");
        System.out.println("4. Nejvetsi spolecny delitel dvou cisel");
        System.out.println("5. Nejmensi spolecny nasobek dvou cisel");
        System.out.println("0. Konec programu");
    }

    private static int nactiVolbu() {
        System.out.print("Zadej volbu ");
        return sc.nextInt();
    }
}
```

Parametry metod

- Formální parametry – v deklaraci
  - Deklarace formálních parametrů v záhlaví metody – v okrouhlých závorkách za jménem metody
  - Jednotlivé deklarace od sebe odděleny čárkou
  - Deklarace parametru – `<typ> <identifikátor>`
  - V každé deklaraci je uveden typ a identifikátor parametru
  - Identifikátor používáme v kódu pro přístup k příslušné hodnotě
  - Typ parametru
    - Parametry primitivních datových typů
    - Parametry typu pole (popřípadě vícerozměrné pole)
    - Parametry objektového typu (typu, který je deklarován jako třída)

- Skutečné parametry – při volání
  - Volání metody – jménem
  - Parametry, pro které má být proveden aktuální výpočet
  - Pořadí, počet a typ skutečných parametrů musí být shodný s pořadím, počtem a typem formálních parametrů uvedených při deklaraci metody

- Metoda `main` a její parametr

Ukázka metod s parametry

```
package alp.p030.metody;

public class MathTools {
   public static long mocnina(int a, int n) {
        long vysledek = 1;
        for (int i = 1; i <= n; i++) {
            vysledek *= a;
        }
        return vysledek;
    }

    public static double mocnina(double a, int n) {
        for (int i = 1; i <= Math.abs(n); i++) {
            vysledek *= a;
        }
        if (n < 0) return 1/vysledek;
        else return vysledek;
    }

    public static double mocnina(double a, double x) {
        return Math.exp(x * Math.log(a));
    }

    // ...
}
```

Volání metody

- Metodu voláme jménem – do okrouhlých závorek za jménem uvedeme skutečné parametry
- Pokud je metoda v jiné třídě, musí jménu předcházet
  - Název třídy – pokud se jedná o metodu třídy (deklarace s modifikátorem `static`)
  - Název objektu – pokud se jedná o metodu objektu (zatím jsme používali metody objektů `sc`, `out`)
- Pokud je třída, v jiném balíku, nutno provádět import, nebo můžeme použít „***plně kvalifikované jméno***“ = „***fully kvalified name***“
- Metoda vracející hodnotu konkrétního typu
  - Použití ve výrazu
  - Použití na místě příkazu (pokut toto má smysl)
- Metoda nevracející hodnotu
  - Použití na místě příkazu
- Mechanismus volání metody
  - Volání metody – přidělení paměti pro běh (pro provedení metody) – paměť je přidělena v oblasti, které se říká „zásobník“
  - V jazyce Java jsou **parametry předávány hodnotou** – na zásobník do lokální paměti metody je uložena hodnota parametru
  - Paměť programu – obecně program ve vyšším programovacím jazyce
    - KÓD, GLOBÁLNÍ DATA, ZÁSOBNÍK (STACK), HALDA (HEAP)

Volání metod

Volání metod ve stejné třídě, ve které je metoda deklarována

- Odkaz na předchozí ukázku programu s metodami `main()`, `vypisMenu()`, `nactiVolbu()`
- V metodě `main` jsou volány další dvě uvedené metody
  - Tyto metody jsou volány pouze jménem (jedná se o statické metody, jsou volány ze statického kontextu – OK)

```
vypisMenu();
int volba = nactiVolbu();
```

  - Pokud metody používáme pouze interně v dané třídě mohou být deklarovány jako privátní

Volání metody z jiných tříd balíku, případně ze tříd v jiných balících

- Použití metod z dříve uvedené třídy `MathTools`
  - Pokud chceme metody `mocnina()` používat/volat v metodách jiných tříd, je nutné zapsat volání ve tvaru

```
MathTools.mocnina(a, n)
```

  - Pokud použijeme metody `mocnina()` v jiné třídě jiného balíku, je navíc nutné provést příslušný import

```
import alp.p030.metody.MathTools;
```

    nebo použít plně kvalifikované jméno

```
alp.p030.metody.MathTools(a, n)
```

Hlavní třída

- Hlavní třída programu – metoda `main`
  - Vstupní bod programu, volání metody
  - Deklarována s modifikátory `public`, `static`
  - Parametrem – pole textových řetězců

- Modifikátor `private/public`
  - Specifikace přístupu – pouze z dané třídy nebo i mimo danou třídu
- Modifikátor `static`
  - Deklarace metod třídy
- Parametr `args`
  - Parametry příkazové řádky při spouštění programu

- Hlavní třída – libovolné množství metod
  - Pokud se bude jednat o metody, které chceme použít/volat bez vytváření objektu třídy musí mít modifikátor `static`

Knihovní třída

- Do programu lze přidat libovolné množství dalších tříd mimo hlavní třídu

- Třídy
  - Vytvoření instance
  - Knihovní třídy

- Knihovní třída
  - Knihovna funkcí
  - Metody, atributy
  - Neslouží pro vytváření objektů/instancí – třída by měla znemožnit vytváření instancí (privátním konstruktorem, viz později)
  - Metody, atributy v knihovní třídě – deklarovány jako statické (metody, atributy třídy), deklarace s modifikátorem `static`
  - Příkladem takové třídy je třída `Math` nebo dříve uvedený fragment vlastní třídy `MathTools`