

Java 1 • lekce 2

Filip Jirsák

14. 2. 2023

online



Třída

Třída je základní jednotka aplikace v Javě.

```
// Soubor Hlavni.program.java v adresáři cz/czechitas/java/lekce02.
package cz.czechitas.java.lekce02;

public class HlavniProgram {

    public static void main(String[] args) {
        System.out.println("Ahoj světe!");
    }
}
```

- Běžné třídy musí být uložené v souboru, který se jmenuje stejně, jako třída (včetně velikosti písmen) + přípona ` `.java` .
 - Např. třída `HlavniProgram` bude v souboru `HlavniProgram.java` .
- Adresáře odpovídají ` package` v záhlaví třídy. Jendotlivé adresáře jsou v package oddělené tečkou ` .` .
- Názvy tříd se píšou *CamelCase* (počáteční písmena slov jsou velká) s velkým písmenem na začátku.

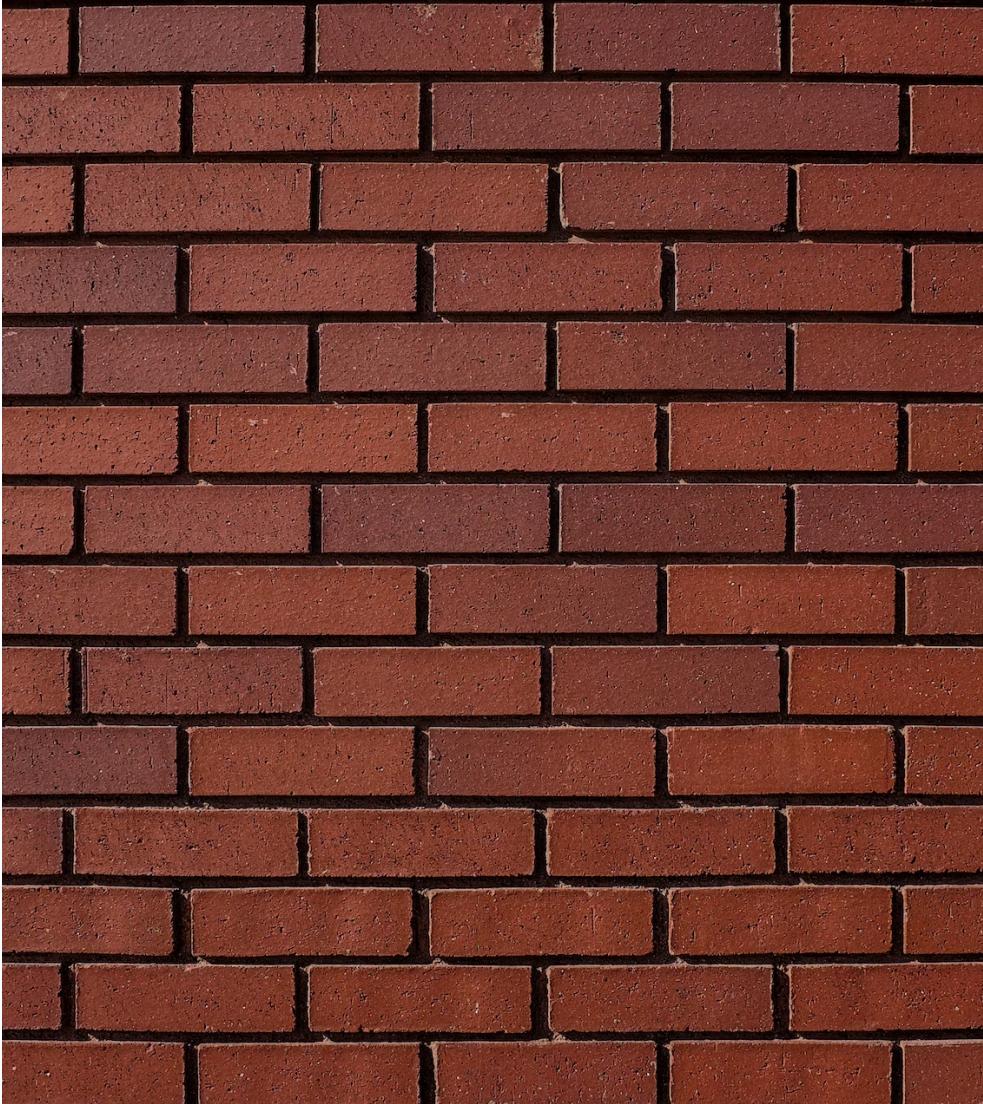
Spouštěcí metoda aplikace

```
public class Application {  
  
    public static void main(String[] args) {  
        System.out.println("Ahoj světe!");  
    }  
}
```

- Vždy veřejná statická metoda pojmenovaná `main`
 - Nemusí mít žádný argument
 - Nebo má jeden argument typu pole řetězců `String[]`
- V aplikaci jich může být víc, spouští se konkrétní třída
- V IntelliJ stačí napsat `psvm` a stisknout 

Základní stavební bloky

- Výrazy (expressions)
- Příkazy (statements)
- Bloky (blocks)



Výrazy (expressions)

Výsledkem výrazu je vždy nějaká hodnota.

- Hodnotu je možné uložit do proměnné.
- Hodnotu je možné předat jako parametr do metody.
- Hodnotu je možné hned zapomenout (když není potřeba a někdo nám ji *vnutil*).

```
"Nějaký text"    //text, řetězec znaků, string
42               //celé číslo
123.45          //desetinné číslo
true             //logická hodnota – pravda
false            //logická hodnota – nepravda
'c'              //znak (vždy právě jeden)
Math.random()    //volání metody, která vrací hodnotu
null             //speciální neexistující hodnota
5 + 7            //aritmetické operace s čísly
14 > 23          //porovnání čísel – výsledkem je logická hodnota
"Ahoj " + "světe!" //spojování řetězců
"Ahoj" + ' ' + "světe!" //k řetězci lze připojit i znak
"Odpověď na otázku života, vesmíru a všebec je: " + 42 //k řetězci lze připojit i číslo, ale nedělá se to
```

Příkazy (statements)

Příkaz je něco, co samo o sobě něco udělá. Například:

- Uloží hodnotu do proměnné.
- Zavolá metodu.
- Vyhodnotí podmínku.

```
var text = "Nějaký text";    //do proměnné pojmenované `text` uloží text/string "Nějaký text"
var cislo = 42;
var nahodneCislo = Math.random();    //volání metody a uložení výsledku
System.out.println("Ahoj světe");    //volání metody, která nic nevrací - voláme ji proto, že sama něco dělá

//podmíněný příkaz
if (cislo < 100) {
    System.out.println("Číslo je menší než sto.");
}
```

Příkaz (kromě bloku) se ukončuje středníkem `;`.

 V IntelliJ Idea stačí napsat `sout` a stisknout **Tab**, Idea sama vypíše celý příkaz `System.out.println();`.

Bloky (blocks)

Používají se tam, kam by normálně patřil jeden příkaz, ale my tam potřebujeme uvést více příkazů.

```
int vek = 17;  
System.out.println("Jak je to s tvým věkem?");  
if (vek > 12 && vek <= 18) {  
    System.out.println("Už nejsi malé dítě.");  
    System.out.println("Ale ještě nejsi dospělý.");  
}  
System.out.println("Ale nic si z toho nedělej.");
```

Bloky (blocks)

Patří ke slušnému vychování psát bloky i tam, kde použijeme jen jeden příkaz.

✓ Takhle ano

```
int vek = 23;
if (vek ≥ 18) {
    System.out.println("Už jsi dospělý.");
}
```

✗ Takhle ne

```
int vek = 23;
if (vek ≥ 18)
    System.out.println("Už jsi dospělý.);
```

```
int vek = 23;
if (vek ≥ 18) System.out.println("Už jsi dospělý.");
```

! Pak se totiž snadno stane chyba

```
int vek = 23;
if (vek ≥ 18)
    System.out.println("Už jsi dospělý.");
    System.out.println("Takže můžeš k volbám.");
```

Podmínka

```
if (cislo < 100) {  
    System.out.println("Číslo je menší než sto.");  
}
```

```
if (cislo < 100) {  
    System.out.println("Číslo je menší než sto.");  
} else {  
    System.out.println("Číslo je 100 nebo více.");  
}
```

```
if (cislo < 100) {  
    System.out.println("Číslo je menší než sto.");  
}
```

```
} else if (cislo < 1000) {  
    System.out.println("Číslo je menší než tisíc.");  
}
```

```
if (cislo < 100) {  
    System.out.println("Číslo je menší než sto.");  
} else if (cislo < 1000) {  
    System.out.println("Číslo je menší než tisíc.");  
} else if (cislo < 1_000_000) {  
    System.out.println("Číslo je menší než milion.");  
} else {  
    System.out.println("Číslo je tisíc nebo více.");  
}
```

Cyklus `while`

„Dokud je splněna podmínka, opakuj cyklus.“

```
var random = new Random();

var cislo = random.nextInt(6) + 1;
while (cislo != 6) {
    System.out.println("Padlo číslo " + cislo + ". Hoď ještě jednou.");
    cislo = random.nextInt(6) + 1;
}
System.out.println("Hodil jsi 6, nasad figurku a hraj.");
```

! Pozor na nekonečný cyklus!

💡 Znak `≠` se zapisuje jako ``!=`` následovaný ``=`` – některé programy tuto kombinaci znaků zobrazí jako ``≠``.

Cyklus `for`

„Opakuj cyklus n -krát.“

```
for (var i = 0; i < 10; i++) {  
    System.out.println("Opiš 10x za trest.");  
}
```

```
var i = 0;  
while (i < 10) {  
    System.out.println("Opiš 10x za trest.");  
    i++;  
}
```

Nastav výchozí hodnotu.

Opakuj, dokud je splněna podmínka.

Proved na konci každé obrátky cyklu.

`i++` znamená *vem hodnotu z `i`, přičti k ní jedničku a výsledek ulož zpět do `i`.*

💡 V IntelliJ Idea stačí napsat `for i` a stisknout `Tab`, Idea vytvoří kostru cyklu. Pomocí `Tab` pak procházíte jednotlivá místa, která je potřeba doplnit.

💡 Programátoři často počítají od `0`. Když začnete počítat od `0` a v podmínce použijete `<` místo `≤` (psáno jako `<` a `=`), bude v podmínce to číslo (tady `10`), kolikrát se má cyklus opakovat.

Cyklus `for`

„Opakuj cyklus n -krát.“

```
for (var i = 0; i < 10; i++) {  
    System.out.println("Opiš 10x za trest.");  
}
```

```
var i = 0;  
while (i < 10) {  
    System.out.println("Opiš 10x za trest.");  
    i++;  
}
```

Nastav výchozí hodnotu.

Opakuj, dokud je splněna podmínka.

Proved na konci každé obrátky cyklu.

`i++` znamená *vem hodnotu z `i`, přičti k ní jedničku a výsledek ulož zpět do `i`.*

💡 V IntelliJ Idea stačí napsat `for i` a stisknout `Tab`, Idea vytvoří kostru cyklu. Pomocí `Tab` pak procházíte jednotlivá místa, která je potřeba doplnit.

💡 Programátoři často počítají od `0`. Když začnete počítat od `0` a v podmínce použijete `<` místo `≤` (psáno jako `<` a `=`), bude v podmínce to číslo (tady `10`), kolikrát se má cyklus opakovat.

Cyklus `for`

„Opakuj cyklus n -krát.“

```
for (var i = 0; i < 10; i++) {  
    System.out.println("Opiš 10x za trest.");  
}
```

```
var i = 0;  
while (i < 10) {  
    System.out.println("Opiš 10x za trest.");  
    i++;  
}
```

Nastav výchozí hodnotu.

Opakuj, dokud je splněna podmínka.

Proved na konci každé obrátky cyklu.

`i++` znamená *vem hodnotu z `i`, přičti k ní jedničku a výsledek ulož zpět do `i`.*

💡 V IntelliJ Idea stačí napsat `for i` a stisknout `Tab`, Idea vytvoří kostru cyklu. Pomocí `Tab` pak procházíte jednotlivá místa, která je potřeba doplnit.

💡 Programátoři často počítají od `0`. Když začnete počítat od `0` a v podmínce použijete `<` místo `≤` (psáno jako `<` a `=`), bude v podmínce to číslo (tady `10`), kolikrát se má cyklus opakovat.

Cyklus `for`

„Opakuj cyklus n -krát.“

```
for (var i = 0; i < 10; i++) {  
    System.out.println("Opiš 10x za trest.");  
}
```

```
var i = 0;  
while (i < 10) {  
    System.out.println("Opiš 10x za trest.");  
    i++;  
}
```

Nastav výchozí hodnotu.

Opakuj, dokud je splněna podmínka.

Proveď na konci každé obrátky cyklu.

`i++` znamená *vem hodnotu z `i`, přičti k ní jedničku a výsledek ulož zpět do `i`.*

💡 V IntelliJ Idea stačí napsat `for i` a stisknout `Tab`, Idea vytvoří kostru cyklu. Pomocí `Tab` pak procházíte jednotlivá místa, která je potřeba doplnit.

💡 Programátoři často počítají od `0`. Když začnete počítat od `0` a v podmínce použijete `<` místo `≤` (psáno jako `<` a `=`), bude v podmínce to číslo (tady `10`), kolikrát se má cyklus opakovat.