

# Java - úvod do programování

Lektor: Ondrej Mihályi



**ictPRO**



## 1. den

- Základy jazyka Java - Datové typy, operátory, řídicí příkazy
- Úvod do objektově orientovaného programování

## 2. den

- Návrh a tvorba tříd, metod, objektů
- Vytváření balíků a implementace rozhraní

## 3. den

- Vstupní a výstupní operace
- Tvorba samostatných aplikací



- Základy jazyka Java
  - Datové typy, Proměnné, Operátory, Řídící příkazy
- Seznámení s vývojovým prostředím
- Složené datové typy
  - Pole, Textové řetězce
- Úvod do objektově orientovaného programování



# Základy jazyka Java

# Charakteristika jazyka Java



- Jeden ze 3 nejpoužívanějších jazyků
- Na mnoha různých platformách a zařízeních
- Podobné jazyky: JavaScript, C, C++
- První verze jazyka Java v roce 1996
- Rozvoj jazyka z větší části řízen firmou Oracle



```
String[] cars = {"Volvo", "BMW", "Ford"};  
for (String i : cars) {  
    System.out.println(i);  
}
```



- Striktně typovaný
  - Proměnná (hodnota) má vždy daný typ
- Kompilovaný
  - je potřeba program “přeložit” před spuštěním
- Interpretovaný
  - “přeložený” program se spouští příkazem **java**
- Objektově-orientovaný
  - kód a data jsou seskupené do objektů



- Vývojové prostředí
  - **Eclipse IDE** - stáhnout z <https://www.eclipse.org/downloads/>
    - Při instalaci vyberte **Eclipse IDE for Java Developers**
- Jazyk Java
  - Eclipse IDE instaluje i jazyk Java, není třeba nic dalšího
  - Více možností instalace Javy:  
<https://foojay.io/java-quick-start/install-java/>
  - Pozor, nepoužívat <https://www.java.com/download/> - zastaralé





- Typy informací, které možné uložit do paměti
- Základní (primitive types)
  - celá čísla, reálná čísla, boolean (pravda/nepravda), textový znak
- Pole (více položek stejného typu)
- Textový řetězec (String)
- Typy pro datum a čas
- Třídy

# Základní datový typ: int (celé číslo)



- **int** je zkratka ze slova “integer” - celé číslo
- všechna celá čísla (kladná, záporná, nula)
- Běžné číselné operace:
  - plus:  $5 + 5$ ; mínus:  $3 - 2$ ; násobení:  $5 * 4$
- Porovnání:  $2 < 3$ ;  $3 \leq 5$ ;  $6 \geq 3$
- Rovnost:  $3 == 3$  (!! zdvojené =);  $3 != 4$
- Celočíselné dělení:  $25 / 5 = 5$ ;  $5 / 3 = 1$
- Zbytek po dělení:  $5 \% 3 = 2$



- JShell - interaktivní spouštění Java příkazů
- Spuštění z příkazové řádky:
  - spusti příkazovou řádku - ve Windows v menu spusti příkaz cmd.exe
  - spusti příkaz **jshell**
- Spuštění v prohlížeči
  - <https://onecompiler.com/jshell/3yuzktpve>



Zadej následující příkazy v JShell:

- $1 + 2$
- $3 - 1$
- $3 - 5$
- $2 * 3$
- $(2 - 5) * 3$
- $24 / 8$
- $24 / 7$
- $24 \% 7$
- $3 < 5$
- $4 > 7$
- $23 + 4 >= 5$
- $5 + 5 == 10$
- $5 + 5 != 10$

# Základní datový typ: double (reálné číslo)



- **double** je typ pro přesné uložení reálných čísel
- Reálná čísla z desetinným rozvojem (kladná, záporná, nula)
- Zápis čísla:
  - Desetinné číslo s desetinnou tečkou: **4.56**
  - Vědecký zápis: **1.8732e3** (stejně jako 1873.2)
- Běžné číselné operace a porovnání jako u typu **int**
- Dělení:  $25.3 / 5$ , Dělení celého čísla:  $24.0 / 5 = 4.8$

# Implicitní změna typu: $\text{int} \rightarrow \text{double}$



Číselné operace fungují jenom nad stejným typem

Pokud je jedno číslo **double** a druhé **int** (nezáleží na pořadí), číslo typu **int** se automaticky změní na typ **double**

Příklady:

- $2 + 2 \rightarrow 2$  ( $\text{int} + \text{int} \rightarrow \text{int}$ )
- $3.2 + 2 \rightarrow 3.2 + 2.0 \rightarrow 5.2$  ( $\text{double} + \text{int} \rightarrow \text{double}$ )
- $3 / 2 \rightarrow 1$  ( $\text{int} / \text{int} \rightarrow \text{int}$ ) - celočíselné dělení
- $3.0 / 2 \rightarrow 3.0 / 2.0 \rightarrow 1.5$  ( $\text{double} / \text{int} \rightarrow \text{double}$ )



Zadej následující příkazy v JShell:

- $1.5 + 2.5$
- $3.1 - 1$
- $3.1 < 5.5$
- $2.4 * 3.2$
- $(2.3 - 5) * 3.1$
- $(2.3 - 5) * 3.1 \geq -28$
- $24 / 7$
- $24.0 / 7$
- $24 / 7.0$

# Základní datový typ: boolean (pravda/nepravda)



název **boolean** vychází z booleovské algebry

jenom dvě hodnoty: **true** nebo **false**

výsledek porovnání je boolean:

- $1 < 2 \rightarrow \text{true}$
- $3 == 4 \rightarrow \text{false}$

logické operace:

- `true && false` (AND)
- `true || false` (OR)
- `! true`  $\rightarrow$  `false` (NOT)





Zadej následující příkazy v JShell:

- true
- false
- 1 == 1
- 2 < 1
- true && true
- true && false
- true || false
- false || false
- ! true
- 1 == 1 && 1 == 2
- 2 < 1 || 1 < 2

# Základní datový typ: char (textový znak)



- **char** je zkratka ze slova character (znak)
- Obsahuje jediný znak
- 16-bit unikódové hodnoty (UTF-16)
- Zápis znaku:
  - Znak v jednoduchých uvozovkách: 'A' nebo '@' nebo '語'
  - Může obsahovat unikód reference: '\u20AC' je stejné jako '€'
- Žádné operace, jenom s konverzí na jiný typ



- Speciální typ **String** pro textový řetězec (více znaků)
- Text ve zdvojených uvozovkách: **"Text"**
- Může obsahovat jeden znak, více znaků, nebo žádný:
  - **"A"** - jeden znak, není stejné jako **'A'**
  - **"Toto je text"** - více znaků
  - **" "** - žádný znak, nazývá se "prázdný" řetězec
- Operace zřetězení (spojení)
  - **"Toto" + " je text" = "Toto je text"**

# Implicitní změna typu: char → int, String



- Číselné operace vyvolají automatickou konverzi na **int**
  - **char** se zmení na odpovídající číselný kód
  - Výsledek většinou nedává zmysl
    - 'A' + 'B' → 65 + 66 → 131
- Operace zřetězení s textovým řetězcem
  - **char** se změní na String s jedním znakem:
    - "Znak " + 'A' → "Znak " + "A" → "Znak A"
    - "" + 'A' + 'B' → "" + "A" + "B" → "AB"

# Implicitní změna typu: cokoliv → String



- Operace zřetězení s textovým řetězcem a jiným typem
  - `"Text" + 'A'`
  - `36.0 + "Text"`
- Jiný typ se automaticky změní na text
- **char** se změní na String s jedním znakem
- **int** se změní na String s textovou reprezentací
  - `36 → "36"`
- **double** se změní na String s textovou reprezentací
  - `36.0 → "36.0"`



Zadej následující příkazy v JShell:

- `'A'`
- `'\u20AC'`
- `"Toto je text"`
- `"Toto je \u20AC"`
- `"Toto" + " je text"`
- `"Znak " + 'A'`
- `'A' + 'B'`
- `"" + 'A' + 'B'`



- Proměnné (variables) pojmenovávají datovou informaci
- Jméno - velká a malá písmena
  - záleží na velikosti písmen
  - povolené i podtržítka (\_), znak \$, a číslice (ale ne na začátku)
    - přesto je nedoporučuji používat
- Pevně daný a neměnný datový typ
- Příklady deklarace proměnné:
  - `int numberOfPeople`
  - `String temporaryValue`

# Přiřazení hodnoty do proměnné



- Operátor přiřazení: =
- Na levé straně název proměnné, na pravé straně hodnota
  - `numberOfPeople = 2 + 3`
- Může být i přímo v deklaraci
  - `int numberOfPeople = 2 + 3`
- hodnota může obsahovat proměnnou - použije se její hodnota
  - `int numberOfPeople = 5`
  - `int numberOfHands = 2 * numberOfPeople`





Zadej následující příkazy v JShell:

- `int numberOfPeople`
- `numberOfPeople = 2 * 3`
- `int numberOfHands = 2 * numberOfPeople`
- `double myNumber = 23.5`
- `String nadpis = "Počet rukou"`
- `String vysledek = nadpis + " je " + numberOfHands`
- `System.out.println(vysledek)`
- `System.out.println("Výsledek je: " + vysledek)`

# Cvičení: boolean proměnné



Zadej následující příkazy v JShell:

- `boolean kladneCislo`
- `kladneCislo = 2 > 0`
- `boolean pravda = true`
- `boolean nepravda = ! pravda`
- `int cislo = 10`
- `( cislo > 0 ) && ( cislo % 2 == 0 )`
- `boolean jeKladneSudeCislo = cislo > 0`
- `System.out.println(jeKladneSudeCislo)`



- Vytvoř dvě číselné proměnné a ulož do nich číslo
  - do další proměnné ulož jejich součet
- Vytvoř 3 proměnné a do čtvrté proměnné ulož výsledek nějakého výpočtu, který je obsahuje
- Vytvoř 2 číselné nebo znakové proměnné
  - použi je ve výpočtu, který obsahuje operace AND (&&) a OR (||)
  - ulož výsledek od boolean proměnné
- Vytvoř dvě znakové proměnné a do další textové proměnné ulož jejich zřetězení



- Příkazy jsou části jazyka, které se kombinují a vytvoří program
- Jednoduché příkazy, např. přiřazení do proměnné
- Větvení - vykonej buď to nebo to
- Opakování - vykonej stejný příkaz nebo blok příkazů několikrát
  - buď zadaný počet opakování
  - nebo pokud je splněna podmínka



- Příkaz, který se nedá dělit na víc příkazů
- V programu za příkazem vždy následuje středník ;
- Např. přiřazení nebo deklarace proměnné:
  - `int a;`
  - `int a = 2;`
  - `a = 3;`



## **if** (pokud), **else** (jinak)

- Obsahuje:
  - Podmínku (hodnota typu boolean)
  - Příkaz, který se vykoná, když je podmínka splněna (má hodnotu **true**)
  - Může obsahovat příkaz, který se vykoná, když podmínka není splněna

# Větvení - příklad s jedním příkazem



```
int a;  
if ( 1 < 2 ) {  
    a = 10;  
}
```

# Cvičení: Větvení s jedním příkazem



```
int a;  
if ( 1 < 2 ) {  
    a = 10;  
}  
System.out.println(a)
```

```
int c = 1;  
if ( c < 2 ) {  
    c = 10;  
}  
System.out.println(c)
```

```
int b = 3;  
if ( b < 2 ) {  
    b = 10;  
}  
System.out.println(b)
```

```
int d = 1;  
int e = 2;  
int vetsiCislo = d;  
if ( e > d ) {  
    vetsiCislo = e;  
}  
System.out.println(vetsiCislo)
```



# Větvení - příklad se dvěma příkazy



```
int a;  
if ( 1 < 2 ) {  
    a = 10;  
} else {  
    a = 5;  
}
```

```
int a = 9;  
String rad;  
if ( a > 100 ) {  
    rad = "Stovky";  
} else if ( a > 10 ) {  
    rad = "Desítky";  
} else {  
    rad = "Jednotky";  
}
```

# Cvičení: Větvení se dvěma příkazy



```
int d = 1;
int e = 2;
int vetsiCislo;
if ( e > d ) {
    vetsiCislo = e;
} else {
    vetsiCislo = d;
}
System.out.println(vetsiCislo)
```

```
int a = 5;
String popis;
if ( a < 0 ) {
    popis = "Záporné";
} else if ( a == 0 ) {
    popis = "Nula";
} else {
    popis = "Kladné";
}
System.out.println(popis)
```

# Opakování s podmínkou



- Příkazy se opakují pokud platí podmínka (má hodnotu **true**)
- **while** (podmínka) { příkazy }

- Příklad:

```
int a = 0;
while (a < 10) {
    a = a * 2;
}
```

- Další varianty opakování s podmínkou: **do ... while**, **for**
  - Ukážeme si později

# Pevný počet opakování



- Opakování s pevným počet opakování v Java jazyku neexistuje
- Dá se napsat pomocí číselné proměnné - počítadla

```
int index = 1;  
while (index <= 10) {  
    index = index + 1;  
    příkazy  
}
```

- Nebo použít opakování pro všechny prvky v seznamu
  - Ukážeme si až poznáme datové typy pro seznamy



- **System.out.println(textový výraz)**
  - vyhodnotí textový výraz a výsledek vypíše na obrazovku
  - **println** je zkratka z “print line”
- Textový výraz - cokoliv, co smíme uložit do textové proměnné
- Příklady:
  - `System.out.println("Tento text se vypíše na obrazovku");`
  - `System.out.println("Máme " + 1500 + " Kč");`
  - `System.out.println("Máme " + suma + " " + mena);`

# Cvičení: Výpis na obrazovku



```
System.out.println("Tento text se vypíše na obrazovku");
```

```
System.out.println("Máme " + 1500 + " Kč");
```

```
double suma = 1500.0;  
String mena = "Kč";  
System.out.println("Máme " + suma + " " + mena);
```

```
boolean vysledek = 1 < 2;  
String text = "Výsledek je: " + vysledek;  
System.out.println(text);
```



Zapište následující opakování pomocí příkazu **while**:

- zdvojnásobujte číslo 1 dokud není větší než 100
- uložte text "**-A-**" do proměnné a do další proměnné uložte text, který tento text obsahuje 100 krát
- Vypište všechna čísla od 1 do 20 na výstup
- Vypište sudá čísla od 2 do 20 na výstup
- Vypište všechna čísla od 1 do 20 na výstup a napište, jestli má jednu číslici (je menší než 10) nebo dvě číslice
  - použijte **if** pro větvení uvnitř opakování

# Textové řetězce a třída String



Typ String není jednoduchý typ ale třída

Metody pro práci s textem (length, startsWith,...)

- Volání metody: **text.metoda()**
- Volání metody s parametry: **text.metoda(parametr1, paametr2)**
- Porovnání pomocí **==** ne vždy funguje
  - K porovnání slouží metoda **equals**





- Vybrané metody pro práci s textem
  - length - počet znaků v textu
  - startsWith, endsWith - jestli začíná/končí daným textem
  - substring - vyber část textu
  - toUpperCase - všechna písmena změn na velká
  - indexOf - najdi text uvnitř
  - equals - porovná, jestli jsou texty obsahovo stejné
  - compareTo - porovná abecedně

# Cvičení: Typ String (1/2)



- `System.out.println( "Učím se Javu".length() )`
- `String text = "Učím se Javu"`
- `System.out.println( text.startsWith("Učím se") )`
- `String velkaPismena = "Java".toUpperCase()`
- `System.out.println( velkaPismena == "JAVA" )`
- `System.out.println( velkaPismena.equals("JAVA") )`

# Cvičení: Typ String (2/2)



- `String text = "Učím se Javu"`
- `System.out.println( text.substring(5) )`
- `System.out.println( text.substring(0, 5) )`
- `System.out.println( text.indexOf("Javu") )`
- `String zacatek = text.substring(0, text.indexOf("Javu"))`
- `System.out.println( zacatek )`
  
- `System.out.println("Auto".compareTo("Letadlo"))`



- Pole umožňuje ukládat více hodnot stejného typu
- Předem daný počet hodnot, nemožno zvětšovat/zmenšovat
- Atribut length: **pole.length**
  - pozor, ne pole.length() - není to metoda jako u typu String
- Příklady vytvoření pole
  - Pole pro 10 čísel: **int[] ciska = new int[10]**
  - Pole pro konkrétní textové hodnoty:  
**String[] jmena = new String[] { "Petra", "Jan", "Tomáš" }**

# Pole hodnot - přístup k hodnotám



- Každá hodnota má svoje číslo (index)
- Index je přiřazován od 0
- Poslední index je počet hodnot v poli mínus 1
- Příklady přístupu k hodnotám:
  - První položka : **jmena[ 0 ]**
  - Třetí položka: **jmena[ 3 - 1]**
  - Položka s indexem v proměnné index: **jmena[ index ]**
  - Poslední položka: **jmena[ jmena.length - 1]**

# Cvičení: Pole hodnot



```
int[] ciska = new int[10];  
ciska[0] = 999;  
System.out.println( ciska[0] );
```

```
String[] jmena = new String[] { "Petra", "Jan", "Tomáš" };  
int index = 0;  
while (index < jmena.length) {  
    System.out.println( jmena[index] );  
    index++;  
}
```

```
int[] ciska = new int[] { 1, 2, 3};  
System.out.println( ciska[100] );
```



Můžeme použít **for** pro opakování pro všechny hodnoty

Příklady:

```
for (int cislo : cisla) {  
    System.out.println( cislo );  
}
```

```
for (String jmeno : jmena) {  
    System.out.println( jmeno );  
}
```

# Cvičení: Opakování for



```
int[] cisla = new int[] { 1, 2, 3};  
for (int cislo : cisla) {  
    System.out.println( cislo );  
}
```

```
String[] jmena = new String[] { "Petra", "Jan", "Tomáš" };  
for (String jmeno : jmena) {  
    System.out.println( jmeno );  
}
```





- Vytvořte pole čísel od 1 do 10, vypište je pomocí **for**
- Vytvořte pole několik náhodných čísel a vypište, kolik čísel je v poli
- Vytvořte pole s několika jmény (např. "Petra", "Jan", "Tomáš") a vypište je spolu s jejich délkami (počtem písmen)
- Vytvořte pole 10 čísel v náhodném pořadí, poté pomocí **for** najezněte největší z nich a vypište ho