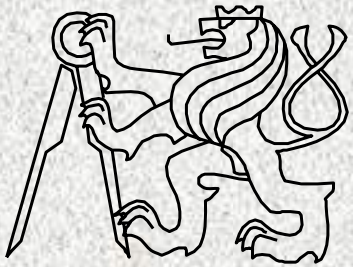
Java – operace a výrazy



A0B36PR1-Programování 1

Fakulta elektrotechnická
České vysoké učení technické

JAVA – struktura programu

```
public class Main {

 public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n;
        System.out.println("Zadej počet ");
        n = sc.nextInt();
        while (n > 0) {
            System.out.println("Vypis " + n);
            n = zmensi(n);
        }
    }

 static int zmensi(int n) {
        n=n-1;
        return n;
    }

} // doc
```

Literály

Přímý zápis hodnoty v programu se nazývá literál

- `int` `34`
- `double` `25.3  1.2E-3`
- `boolean` `false  true`
- `char` `'a' ' 1' ' +'`
- `long` `10000000000000L`
- `float` `4F,  258.52f`

`Pozor! 4.7 - 4.7F = 1.9073486345888568E-7`

Dalším často potřebným literálem je literál typu **`String`** (není to primitivní typ):

`"abcd"  "Nazdar"`

Pojmenované konstanty

Kromě literálů lze v jazyku Java používat pojmenované konstanty, které se deklarují podobně jako inicializované proměnné, v deklaraci je navíc klíčové slovo **`final`**

Příklad deklarace konstant:

```
final int MAX = 100;
final String NAZEV_PREDMETU = "Programovaní 1";
```

Konvence: Jména konstant píšeme velkými písmeny, mezi slovy podtržítka

Konstantě nelze změnit hodnotu přiřazovacím příkazem

```
MAX = 20;       // chyba při překladu
```

Komentáře

Slouží pro komentování kódu, nejsou uvažovány při překladu

```
// toto je koncořádkový komentář
int a = 10;
a = 24; // a=54;
// hodnota a je 24
/*  toto je začátek víceřádkového komentáře
a = 54; - neprovede se
// v zakomentovaném kódu může být vnořen tento
  komentář
*/      ... konec komentáře
```

Komentáře II

```
/*  // po smazání /* zbude prázdný komentář
d = 7;
/**/
---
Přepínací komentář:
/*
Blok A
/*/
Blok B
/**/
1. Blok A zakomentován, B ne
2. po smazání /* blok A platný, B zakomentován.
```

Komentáře III

- Generování dokumentace z komentářů

```
/**
 *
 * @param args
 */
 public static void main(String[] args) {
```

- Generate Javadoc v Run
- Shift F1
- název projektu -> Properties, Documenting > "Include Private and Package Private Members"
- Úprava v Tools-Template

Pro zájemce!

Přiřazovací příkaz

- Slouží pro přiřazení hodnoty proměnné
- Tvar přiřazovacího příkazu:

  <*proměnná*> = <*výraz*>;
- Příklad:

```
x = y + z; // proměnné x se přiřadí součet hodnot
  //proměnných y  a z
x = x + 1; // hodnota proměnné x se zvětší o 1
```

- Proměnné v jazyku Java lze přiřadit pouze hodnotu jejího typu, jinak ztráta přesnosti či chyba
- Příklady **nedovolených** přiřazovacích příkazů:

```
boolean b; int i; double d;
b = 1;//Java je jazyk se silnou typovou kontrolou!!!
i = 1.4;
d = true;
```

Přiřazovací příkaz II

- přířazovací operátory – zkrácený zápis

  <proměnná> = <proměnná> <OP> <výraz> ~

  <proměnná> <OP>= <výraz>

Př.: `j += 5; ~ j = j + 5;`

- přiřazení je výraz !!!

Příklad:

```
int x, y;
x = 6;
// přířazení má hodnotu 6 a lze tedy opět přiřadit!!
// asociativita přiřazení je R, zprava
y = x = x + 6; //stejné jako y = ( x = ( x + 6 ) );
```

Typové konverze

- Typová konverze je operace, která hodnotu nějakého typu převede na hodnotu jiného typu
- Typová konverze může být *implicitní* (vyvolá se automaticky) nebo *explicitní* (v programu je třeba ji explicitně předepsat)
- Konverze typu **`int`** na **`double`** je v jazyku Java implicitní:
  - kde se očekává hodnota typu **`double`**, může být uvedena hodnota typu **`int`**, která se automaticky převede na hodnotu typu *double*

```
double x; int i = 1;
x = i;// hodnota 1 typu int se automaticky převede
      // na hodnotu 1.0 typu double
```

- implicitní konverze je bezpečná

Typové konverze II

- Převod hodnoty typu **`double`** na **`int`** (odseknutím necelé části) je třeba **explicitně** předepsat
- Příklad:

```
double x = 1.2; int i;
i = (int)x;// hodnota 1.2 typu double se odseknutím
  //necelé části převede na hodnotu 1 typu int
```

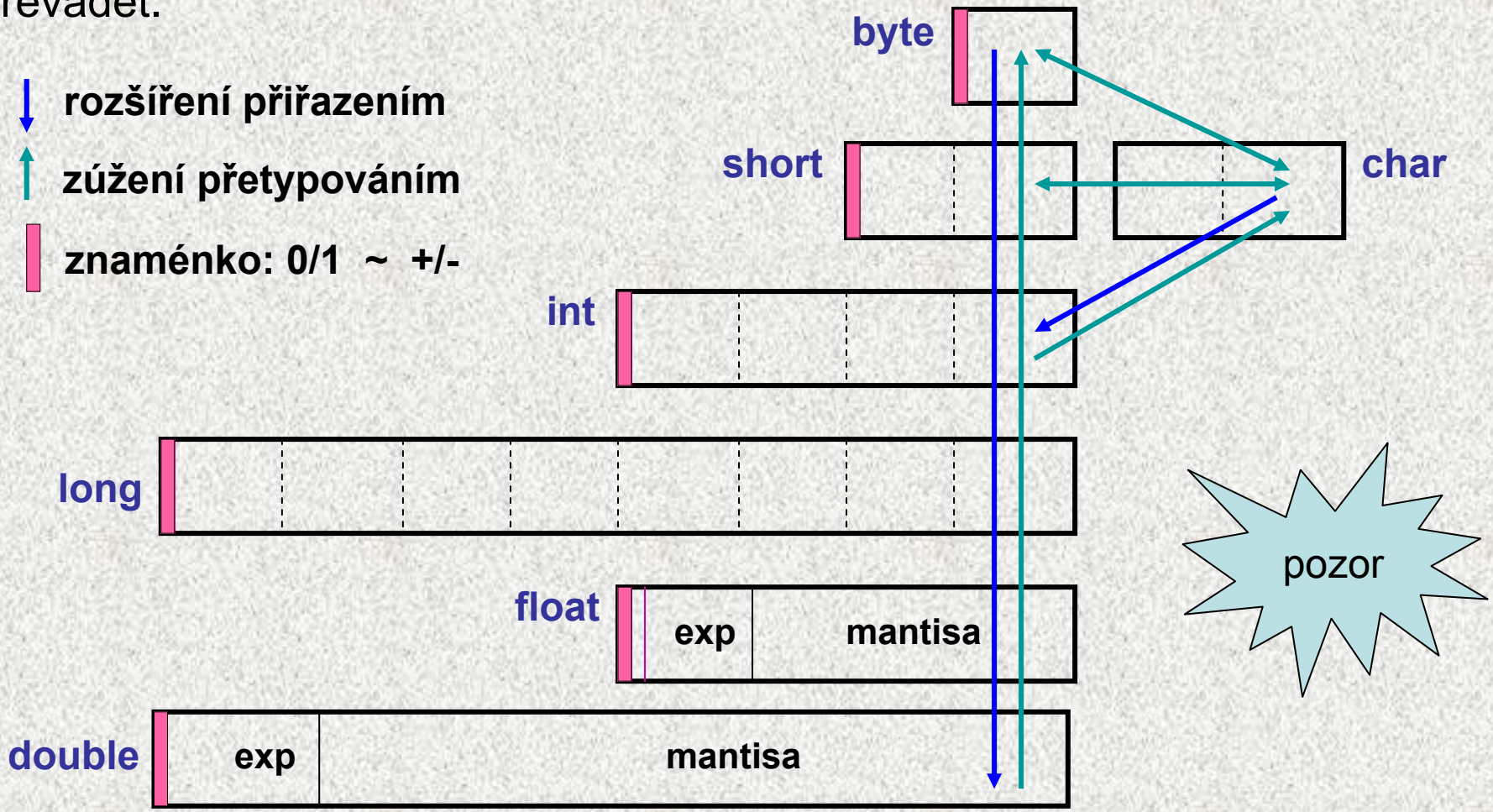
- Explicitní konverze je potenciálně nebezpečná

```
double d = 1E30;
int i = (int)d;   // i je 2147483647 asi 2E10

long l = 5000000000L;
int i = (int)l;// i je 705032704 - oříznuté 4 //bajty
```

Konverze primitivních typů

Primitivní typy jsou vzájemně nekompatibilní, ale jejich hodnoty lze převádět.



Příkazy výstupu I

- Pro výpis dat na obrazovku se v Javě používá příkaz

```
System.out.println(parametr);
// výpis na stdout
```

- Data daná parametrem se vypíší na aktuální řádek a přejde se na další řádek
- Pro výpis dat na aktuální řádek bez přechodu na další řádek lze použít příkaz

```
System.out.print(<parametr>);
```

- Příklad výpisu textu, řetězce:

```
System.out.println("Nazdar");
```

Příkazy výstupu II

Metoda println je přetížena pro všechny primitivní datové typy, pro String a pro všechny objekty

- *n* tedy může být typu **`int, double, float, char,..:`**

```
System.out.println(n);
```

- Primitivní datové typy se zobrazují jako řetězce
- Lze je spojovat navzájem pomocí operátoru +:

```
System.out.println("hodnota proměnné n = " + n);
```

- Formátovaný výstup (zohledňující národní zvyklosti jako je desetinná čárka) je realizován metodou **`printf`**, např.

```
System.out.printf("Hodnota proměnné i je %15d%n",i);
// vypíše obsah proměnné i na 15 míst, předpokládá celé číslo
   %d, poté odřádkuje %n
```

- národní zvyklosti se řídí `Locale.setDefault(Locale.ENGLISH);`

`//ObvodKruhu, FormatovanyVystup`

Druhý program

```
package pr1_3;
   public class DruhyProgram {
     public static void main(String[] args) {
       int x = 10, y;
       y = x + 20;
       System.out.println("hodnota proměnné x je "+x);
       System.out.println("hodnota proměnné y je "+y);
     }
   }
```

- Program po překladu a spuštění vypíše:

hodnota proměnné x je 10

hodnota proměnné y je 30

Poznámka k příkazu výstupu

- Co vypíší následující příkazy:

příkazy	výstup
`int x = 1, y = 2;  System.out.println(x + y);`	**3**
`int x = 1, y = 2;` `System.out.println("součet je "+ (x + y));`	**součet je 3**
`int x = 1, y = 2;` `System.out.println("součet je " + x + y);`	**součet je 12**

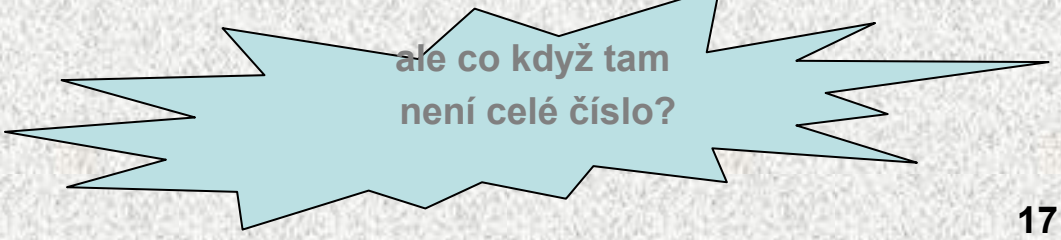
- Příčinou „podivného“ výstupu posledního programu je, že operátor + (a většina dalších) je asociativní zleva, tzn. výraz

`"součet je"+x+y` se vyhodnotí takto: (`"součet je "+x)+y`

Základy vstupu

V dalších příkladech budeme potřebovat příkazy pro vstup číselných dat zadaných na klávesnici – poslouží nám třída **Scanner**

```
package pr1_3;
import java.util.*;
   // třída Scanner je v knihovně java.util
public class TretiProgram {
public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  int x, y, z;
  System.out.println("Zadejte dvě celá čísla");
  x = sc.nextInt(); //nacti celé číslo
  y = sc.nextInt();
  z = x + y;
System.out.println("Součet čísel:"+x+"+"+y+" = "+z);
    }
}
```



Vstup pomocí třídy Scanner

- Využití služeb třídy *Scanner* je třeba deklarovat příkazem

```
import java.util.*;
// načte všechny třídy z knihovny java.util
```

- Je třeba vytvořit objekt třídy *Scanner a napojit jej na standardní vstupní proud:* `Scanner sc = new Scanner(System.in);`
- Třída *Scanner* poskytuje tyto služby:

  `sc.nextInt()` : přečte celé číslo z řádku zadaného klávesnicí (řádek je zakončen klávesou *Enter,* číslo je zakončeno mezerou nebo *Enter*) a vrátí je jako funkční hodnotu typu *int*

  `sc.nextDouble()` : přečte číslo z řádku zadaného klávesnicí a vrátí je jako funkční hodnotu typu *double,* jako oddělovač použijte . nebo čárku v závislosti na definovaném jazyku OS. Lze změnit pomocí před vytvořením Scanneru `Locale.setDefault(Locale.ENGLISH);`

  `sc.nextLine()` : přečte zbytek řádku zadaného klávesnicí a vrátí je jako funkční hodnotu typu *String*

Formátovaný výstup

```
System.out.printf("Cislo Pi = %6.3f %n ", Math.PI);
```

Cislo Pi = 3,142

Specifikace formátu

```
%[$indexParametru][modifikátor][šířka][.přesnost]konverze
```

konverze - povinný parametr

- **typ celé číslo d,o,x - dekadicky, oktalově a hexadecimálně**
- **typ double f je desetinný zápis; e,E vědecký s exponentem**

šířka - počet sázených míst, zarovnání vpravo

.přesnost - počet desetinných míst

modifikátor - v závislosti na typu konverze určuje další vlastnosti, například pro konverzi f (typ double)

- **symbol + určuje, že má být vždy sázeno znaménko,**
- **symbol - určuje zarovnání vlevo,**
- **symbol** 0 **doplnění čísla zleva nulami.**

%n přechod na další řádek a další (formátování data, měny, …)

Pro info

Formátovaný výstup

```
package pr1_3;
public class FormatovanyVystup {
public static void main(String[] args) {
System.out.printf("Cislo Pi = %6.3f %n", Math.PI);
System.out.printf("Cislo Pi = %8.3f %n", Math.PI);
System.out.printf("Cislo Pi = %6.5e %n", Math.PI);
System.out.printf("Cislo Pi = %6.5g %n", Math.PI);
System.out.printf("Cislo Pi = %6d %n", 33);
System.out.printf("Cislo Pi = %4d %n", -33);
    }
}
```

```
Cislo Pi =  3,142
Cislo Pi =    3,142
Cislo Pi = 3.14159e+00
Cislo Pi = 3.1416
Cislo Pi =     33
Cislo Pi =  -33
```

Výrazy

- Výraz předepisuje výpočet hodnoty určitého typu, je to struktura obsahující operandy, operátory a závorky

Příklad: {sin(3.3) + 18.33 – a

- Výraz může obsahovat:
  - proměnné
  - konstanty
  - volání funkcí
  - binární operátory a unární operátory
  - závorky
- Pořadí operací předepsaných výrazem je dáno:
  - prioritou operátorů
  - asociativitou operátorů
- Příklad:

výraz	pořadí operací	zdůvodnění
x + y * z	x + (y * z)	*** má vyšší prioritu než +**
x + y + z	(x + y) + z	**+ je asociativní zleva**

Výraz versus příkaz

Příkaz provádí akci, je zakončen středníkem

```
a=10;
a = Math.abs(x);
System.out.println("Ahoj světe");
```

Výraz má určený typ a hodnotu, není zakončen středníkem

`23` ~ typ **`int`**, hodnota je 23

`14+16/2` ~ typ **`int`**, hodnota 22

`Y=8` ~ typ **`int`**, hodnota 8

Výraz versus příkaz:

- **Výraz má typ a hodnotu**
- **Přiřazení je výraz a jeho hodnotou je hodnota přiřazená levé straně příkazu**
- **Přiřazení se stává příkazem, je-li ukončeno středníkem**

Asociativita a priorita operací

- Binární operace na množině *S* **asociativní**, jestliže platí
  - (*x* * *y*) * *z* = *x* * (y * z) ,pro každé *x*, *y* a *z* v *S*.
- U neasociativních operací je tedy třeba buď důsledně závorkovat, nebo se dohodnout na implicitním pořadí provádění operací – pak se někdy mluví o operacích *asociativních zleva* či *asociativních zprava*.
- **Priorita binárních operací** vyjadřuje v algebře pořadí, v jakém se provádějí binární operace
- Příklady:
  - Odčítání levě asociativní,
    - výraz 10 − 5 − 3 se chápe jako (10 − 5) − 3,
  - Umocňování je asociativní zprava
    - 3 + 5^2 = 28 nebo 3 × 5^2 = 75 ??? $2^{3^4} = 2^{(3^4)}$
  - Přiřazení je asociativní zprava
    - y = y + 8

Aritmetické operátory

Pro operandy typu **int** a **double** budeme požívat tyto aritmetické operátory (seřazeno sestupně podle priority):

unární – (změna znaménka)

binární `*`, `/` a `%` (násobení, dělení a zbytek po dělení)

binární + a – (sčítání a odčítání)

Jsou-li oba operandy stejného typu, výsledek aritmetické operace je téhož typu

Jsou-li operandy různého typu, operand typu **int** se implicitní konverzí převede na hodnotu typu *double* a výsledkem operace je hodnota typu double

- Výsledkem dělení operandů typu **int** je celá část podílu
  např. `7 / 3` je 2 `-7 / 3` je -2
- Pro zbytek po dělení platí: $x \% y = x - (x / y) * y$
  např: `7 % 3` je 1 `-7 % 3` je -1 `7 % -3` je 1 `-7 % -3` je -1

Aritmetické operátory II

Obdobně pro typ double ☺:

3,8 % 1,6 = 0,6

speciální inkrementační a dekrementační operátory:

++x, x++, --x, x--

```
int i = 1, a = 9;
a = i++;
a = ++i;
a= ++(i++);
```

```
 i:          a:
  1          9
  2          1
  3          3
tak to už neprojde
```

Relační operátory

Hodnoty všech jednoduchých typů jsou uspořádané a lze je porovnávat relačními operátory

Budeme používat tyto relační operátory (priorita je menší než priorita aritmetických operátorů):

`>, <, >=, <=` (větší, menší , větší nebo rovno, menší nebo rovno)

`==, !=` (rovná se, nerovná se)

Výsledek relační operace je typu **`boolean`** (**`true`**, když relace označená operátorem platí, **`false`** v opačném případě)

Jestliže při porovnávání číselných hodnot jsou operandy různého typu, operand typu **int** se implicitní konverzí převede na hodnotu typu **double**

```
int i=10; double x=12.3; boolean b;
System.out.println(i==10); // vypíše true
System.out.println(i+1==10);      // vypíše false
b = i>x;                  // proměnné b se přiřadí false
System.out.println(Math.sqrt(100)==10);  // ??
```

Logické operátory

Logické operátory jsou definovány pro hodnoty typu *boolean*

unární ! (negace)

binární && resp. & (konjunkce, logický součin)

binární || resp. | (disjunkce, logický součet)

x	y	!x	x && y	x || y
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

Negace má stejnou prioritu, jako změna znaménka, logický součin má nižší prioritu než relační operátory

Operace && a || se vyhodnocují zkráceným způsobem, druhý operand se nevyhodnocuje, když lze výsledek určit již z prvního operandu

Logické operátory II

```
int n = 10; boolean b1 = false, b2 = true;

System.out.println(1<=n && n<=20);  // vypíše se true

System.out.println(b1 || !b2); // vypíše se     false

if (y != 0 && x/y < z)   // zkrácené vyhodnocení
```

Pozor:pořadí priorit je `&` a `|` , `&&` a `||` tedy:

- false && true || false = >> false
- false && true | false = >> false
- true && false | true = >> true

Matematické funkce

Při číselných výpočtech často potřebujeme matematické funkce jako *abs, sin, cos, sqrt* (druhá odmocnina), *log* (přirozený logaritmus) atd.

Tyto funkce poskytuje třída *Math*, nemusí se importovat, je v java.lang

```
import java.util.*; // pro Scanner
public class Prepona {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Zadejte délku odvěsen pravoúhlého
  trojúhelníka");
double x = sc.nextDouble();
double y = sc.nextDouble();
double z = Math.sqrt(x*x+y*y);
System.out.println("Délka přepony je "+z);
   }
}
```

Knihovní třída Math

- Některé z funkcí poskytovaných třídou *Math*

```
public static int abs(int a)
public static double abs(double a)
public static int max(int a, int b)
public static double max(double a, double b)
public static int min(int a, int b)
public static double min(double a, double b)
public static double sqrt(double a)
public static double sin(double a)
public static double random()
```

  - vrátí pseudonáhodné číslo větší nebo rovno 0.0 a menší než 1.0

- Třída poskytuje též dvě konstanty: `E` a `PI`

Knihovní třída Math - příklad

```
import java.util.*;

public class ObvodKruhu {
    public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("zadejte poloměr kruhu");
    double r = sc.nextInt();
    System.out.println("obvod kruhu je "+2*Math.PI*r);
       }
   }
```

Nekonečno a nečíslo, typ double

```
public class Nekonecno {
public static void main(String[] args) {
double nula = 0.0;
double vysledek = +5.0 / nula;
System.out.println(vysledek);
if (Double.isInfinite(vysledek) == true)
 System.out.println("nekonecno");
vysledek = -5.0 / nula;
System.out.println(vysledek);
if (Double.isInfinite(vysledek) == true)
 System.out.println(" -nekonecno");
System.out.println("MAX = " + Float.MAX_VALUE + ", 2 *
                    MAX = " + (2 * Float.MAX_VALUE));
vysledek = nula / nula;
System.out.println(vysledek);
if (Double.isNaN(vysledek) == true)
 System.out.println("neni cislo");
 }
}
```

Infinity

nekonecno

-Infinity

-nekonecno

MAX = 3.4028235E38,
2 * MAX = Infinity

NaN

neni cislo

PRO ZÁJEMCE