# Opakovanie

- Výraz
- Syntaktický strom
- Vyhodnocovací formulár
- Gramatika / EBNF
- Vyhodnotenie formuláru

# I. Java - Základné elementy

## 2. Dátové typy

- Primitívne dátové typy
- Deklarácia premenných
- Priradenie
- Side effects

# 2.1. Primitívne dátové typy

- Doteraz: iba programy s celými číslami
- Celé čísla nie sú pre niektoré účely spracovania informácií vhodná a efektívna reprezentácia:
  - Funkcie nad reálnymi číslami
  - Spracovanie textu

# Primitívne dátové typy

- celé čísla:
  ..., -3, -2, -1, 0, 1, 2, ..., 4711
- reálne čísla:
  0.321, $\pi$, e, 0.777..., $3.1*10^{-11}$
- znaky:
  ‘a‘, ‘b‘, ‘c‘, ... , ‘A‘, ‘B‘, ... ,‘0‘, ... ‘9‘, ‘&‘, `*` , ...
- znakové reťazce:
  “Programovanie!“, “Znakový reťazec: %$§.“
- pravdivostné hodnoty:
  false, true

# Celé čísla

- **byte** –128 až 127
- **short** –32768 až 32767 (2 Byte) v Jave
- **int** -2147483648 až 2147483647 (4 Byte) v Jave
- **long** -9223372036854775808 až 9223372036854775807 (8 Byte) v Jave

- Pravé celé čísla (bez obmedzenia) neexistujú (ako primitívne dátové typy) !!!

# „Reálne čísla“

- **`float`** ca. $-3*10^{38}$ až $3*10^{38}$ (4 Byte)
  ca. 6 bis 7 miest presnosť
  Príklady: 3.578671E+22F
  0.0145456E-15F

- **`double`** ca. $-2*10^{308}$ až $2*10^{308}$ (8 Byte)
  15 miest presnosť
  Príklady: 1.234567897897E+220D
  -3.956745656e-12

# Znaky (Character)

- **`char`** Znaky v tzv. Kódovaní Unicode
  **`http://www.unicode.org/charts/web.html`**
- V Unicode su zobraziteľné „všetky znaky sveta“ (2 Byte):

```
`a``, ... , `z`,`ä`, `ö`, `ü`, `ß`,
`A`, ... `Z`, ... , `0`, ... , `9`,
`+`, `- `,
`\t`, `\n`, `\r`, `\\`, `\`` , ...
`\u2200`= ∀, `\u24B8`= ©
```

# Znaky (Character)

- Znakové reťazce v Unicode! (Java)
- Znakové reťazce sú na rozdiel od znakov v dvojitých úvodzovkách:

```
"Toto je príklad pre jeden\n
znakový reťazec so zalomením
riadku!"
```

# Pozor!

- `'c'` je jeden znak!
- `"c"` je znakový reťazec, ktorý pozostáva z jedného znaku
- `"\u2200"` je tiež znakový reťazec, ktorý pozostáva z jedného znaku
- `"\\u2200"` je znakový reťazec, ktorý pozostáva zo 6 (!) znakov
  (`\\` reprezentuje spätné lomítko Backslash)

# Pozor!

Znakové reťazce nie sú prísne vzaté primitívny dátový typ! (vysvetlíme neskôr)

V Jave dátový typ String

# Pravdivostné hodnoty

- `boolean` **(nie je v jazyku C)**

  hodnoty: `false` a `true`

- **V C pre pravdivostné hodnoty int – 0 je false a 1 je true**

# Budeme používať

- `int`
- `double`
- `char`
- `boolean` a
- `String (nie je v jayzku C)`

Iné programovacie jazyky poskytujú podobné primitívne dátové typy (viac alebo menej typov, iné názvy typov):

z.B. `nat , real`

- Pre iné dátové typy platia väčšinou analogické pravidlá

# 2.1. Deklarácie premenných

- Každá premenná má jednoznačne priradený dátový typ
- Typ premennej je pevne daný deklaráciou premennej:

```
int n;
int sum;
double x;
String name;
```

# Deklarácie premenných

- Premenné môžu byť deklarované v tekte programu na hociktorom mieste (ale iba jeden krát, presnejšie vysvetlíme neskôr)
- Viaceré premenné môžu byť deklarované spolu:

```
int sum1, sum2, test;
double x, y, z;
```

# EBNF: Deklarácie premenných

```
<Deklaration> = <Typ> <ZoznamIdentifikátorov> ";"

<Typ> =     "int" | "long" | "byte" | "short" |
      "double"  | "float" |
      "char"    |
      "boolean" |
      "String"

<ZoznamIdentifikátorov> =
  <Identifikátor>                                |

  < ZoznamIdentifikátorov > "," <Identifikátor>

<Identifikátor> = ...
```

# Identifikátory

- Identifikátory pozostávajú z reťazca **Písmen** a **Číslic** (v Unicode), pričom prvý znak musí byť písmeno
- Pojem písmeno je v tomto kontexte dosť široký:
  `'ä', 'ö', 'ü', '_', '$'`

  **NIE:** `'+', '-', '[', '%', '©', ...`

  **Viac:**

```
Character.isJavaIdentifierStart(c)
Character.isJavaIdentifierPart(c)
```

# Identifikátory – mená premenných

- Všetky písmená v mene sú dôležité
- Rozlišujú sa malé a veľké písmená:
  `Sum` a `sum` sú dva rozdielne identifikátory.

- Konvencia: Premenná začína s malým písmenom

# Typy Konštánt

Každá konštanta je jednoznačného typu:

- `0, -1, 4711, 32768, -1000000`
  sú typu `int` (aspoň pre nás, lebo short atď.)
- `0.1, -1.7359e-34, 0.0`
  sú typu `double`
- `'a', '©'`
  sú typu `char`
- `"Bla bla", "Test\n Test\n"`
  sú typu `String`

# Typy výrazov

Každý výraz je jednoznačného typu:

```
//int n, m;
// double x, y;

1 * 2      // je typu int
n  +  m    // je typu int
x          // je typu double
x / y      // je typu double
x < y      // je typu boolean
```

# Typ operátorov

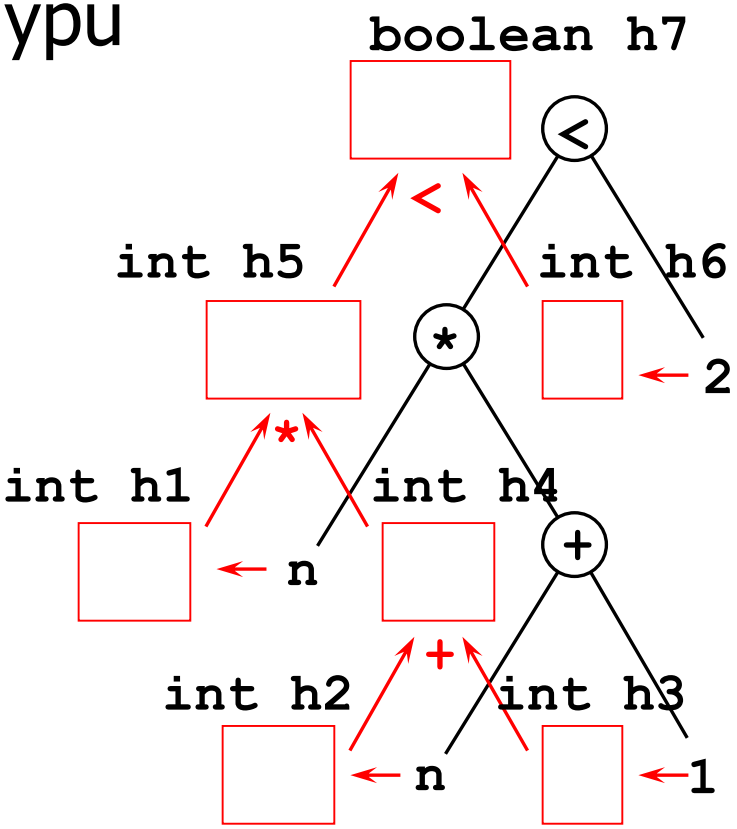
- Pre každý operátor (a pre každú funkciu) je jednoznačne dané, akeho typu sú operandy a akého typu je výsledok!
- Príklady:

```
Math.min: int, int → int
Math.abs: int → int
  . + . : int int → int
  . < . : int int → boolean
```

# Typ operátorov

získame (induktívne) z Typu
konštánt,
premenných, a
operátorov
a funkcií

```
int n; ...
n * (n + 1) < 2
```



# 2.3 Polymorfizmus

(polymorhph = „mnohoformný“)

Operátor + je rezervovaný pre sčítanie dvoch celých čísel typu **`int`**!

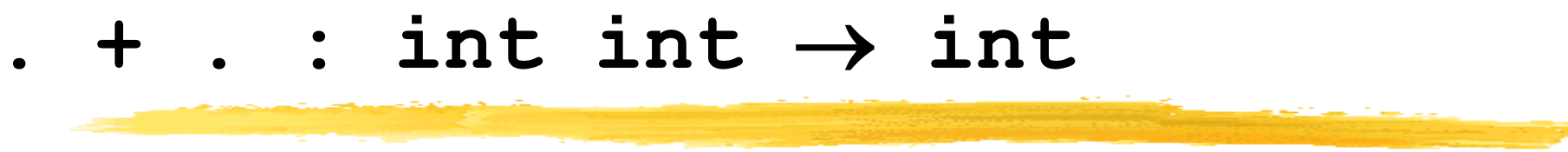
Radi by sme však + použili pre sčítanie dvoch reálnych čísel typu **`double`**!

**Dá sa to?**

# Polymorphismus

- Dá: Deďže typy subvýrazov, ktoré sú operandami + , sú známe, môžu byť tieto typy použité na identifikáciu operátora + . Operátor + označuje rôzne operácie:

```
. + . : int int → int
. + . : double double → double
. + . : String String → String
```




Sčíta dve čísla typu **`int`**, výsledkom je číslo typu **`int`**

## `. + . : double double → double`

Sčíta dve čísla typu **`double`**, výsledkom je číslo typu **`double`**

## `. + . : String String → String`

(V Jave) Zreťazí obidva znakové reťazce do výsledného reťazca

**`"abc" + "def"`** je **`"abcdef"`**

# Ďalšie (implicitné) formy

```
. + . : int double → double
. + . : double int → double
. + . : int String → String
. + . : String int → String
. + . : double String → String
. + . : String double → String
. + . : String boolean → String
. + . : boolean String → String
```

Vždy implicitná Konverzia na typ `double` resp. `String`

# Ďalšie operátory
# Operatoren

- To isté platí pre ďalšie aritmetické operátory: `-, *, /`
- `%` (Zvyšok po modulo delení) nie je definovaný pre reálne čísla
- Pre znakové reťazce nie sú definované operátory: `%, -, * a /`

# Príklady

```
"abc" + 1   je "abc1"
1 / 2       je 0      (celočíselné delenie)
1 / 2.0     je 0.5    (reálne delenie)

"abc" + false   je "abcfalse"
"abc" + (1 + 2) je "abc3"
```

Čo je výsledkom výrazu `"abc" + 1 + 2` ?

# Operátory

- **logické operácie:**
  - `!` Negácia (unárna)
  - `&&` Konjunkcia „a súčasne“ (binárna)
  - `||` Disjunkcia „alebo“ (binárna)
- **Porovnania:**
  - `==` Test na rovnosť (binárna)
  - `<, <=, >, >=` menšie atď. , resp. väčšie (binárne)
  - `!=` Test na nerovnosť (binárna)

iba pri číslach

# Priorita operátorov

- unárne operátory — najvyššia priorita
- * / %
- + -
- < <= > >=
- == !=
- &&
- ||

najnižšia priorita

# 3. Priradenie

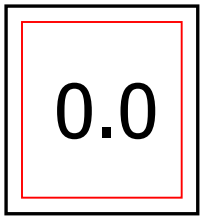
- Hodnota výrazu môže byť priradená premennej rovnakého typu:

```
int n, m;
boolean b;
...
m = n * (n + 1);
b = m < n;
```

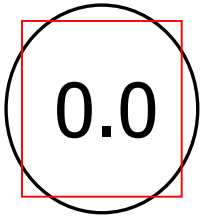
Operátor priradenia = má najnižšiu prioritu!

# Typy musia byť rovnaké!

```
double x;
x = 0.0;
```




```
int n;
n = 0.0;
```




Neskôr uvidíme, že nie vždy je to také prísne!

# Deklarácia a inicializácia

```
int n;
n = 0;
```

môžu byť zapísané spolu:

```
int n = 0;
```

V princípe musí byť premennej priradená hodnota predtým ako je použitá v nejakom výraze, ktorý sa bude vyhodnocovať.

# Ďalšie príklady

```
/* Predpokladáme, že int n
 * je už definované. */

int h1 = n, h2 = n, h3 = 1,
    h4 = h2 + h3, h5 = h1 * h4,
    h6 = 2, h7 = h5 / h6;
```

# 4. Komentáre

- Komentáre sú pre dokumentáciu programu KĽÚČOVÉ
- Komentáre v skutočnosti nepatria k programu, sú ignorované počítačom
- Rôzne varianty komentárov

# Jednoriadkové komentáre

```
// po dvoch lomítkach bude
// text do konca riadku
// ignorovaný, napr.
int i; // tu deklarujeme i
i = 0; // tu inicializujeme i
```

# Viacriadkové komentáre

```
/* Dlhšie komentáre sa píšu medzi
kombináciu lomítko-hviezdička na
začiatku komentáru a hviezdička
lomítko na konci komentáru */
```

# Dokumentácia

```
/**
   Komentáre s dvojitou hviezdičkou
  sú automaticky použité na
  vytvorenie dokumentácie. Taktiež je
  možné zadať špeciálne informácie k
  programu, napr:
    @author The Author
*/
```