

1-AIN-172:

Programovanie (4) (alias JAVA pre C++ programátorov)



Peter Borovanský
KAI, I-18

borovan@ii.fmph.uniba.sk

<http://dai.fmph.uniba.sk/courses/JAVA/>





Čo je na stránke predmetu

Prednáška: <http://dai.fmph.uniba.sk/courses/JAVA>

- ??, ??:??, 2hod, ?? **Programovanie 4** p 2ia*

Cvičenia :

- Štvrtok, 16:30, H6 (Peter Gergel', Peter Borovanský)
- Štvrtok, 18:10, H3 (Juraj Holas, Peter Borovanský)
- domáce úlohy opravujú (Andrej Jursa)

Používame systém LIST: <http://capek.ii.fmph.uniba.sk/list/>

Kontakt [všetci cvičiaci a ja]: prog4java@lists.dai.fmph.uniba.sk

Konzultačné hodiny:

- Štvrtok 14:00-16:00
- **kedykoľvek po e-dohode s vyučujúcim** ☺ ☺ ☺



Hodnotenie

A	86-100
B	77-85
C	68-76
D	59-67
E	50-58
Fx	0-49

- 2x**quadterm** (2x15), **midterm** (25), semestrálny **projekt** (15), **skúška** (30),
- **midterm** je písomný test v jedinečnom termíne **11.4. 18:10**, nedá sa opakovať,
- dva **quadtermy** sú testy pri počítačoch v terminálke počas riadnych cvičení,
- **cvičenia sú povinné**, akceptujú sa 2 absencie za semester, žiadne PN-ky...
- cvičenia končia povinnou **domácou úlohou**, ktorej elektronické odovzdanie sa očakáva do termínu cca 7-8 dní. **Neodovzdanie, resp. kolektívne riešenia sú penalizované zápornými bodmi (-3, -2, -1 body podľa nedostatkov)**,
- semestrálny projekt je nutná podmienka ku skúške (musí byť; uznaný cvičiacim pred termínom skúšky), témy projektov budú zverejnené cca po Veľkej noci,
- v nepravidelne sa objavujú **prémiové úlohy**, ktoré sú na zlepšenie bodovej bilancie jednotlivca pri skúške (kolektívne riešenia sa opäť neakceptujú),
- predtermín (**bypass excelencie**) bude pre záujemcov 3.3. 16:30, záujemci sa prihlasujú e-mailom do 1.3.
- **v prípade akýchkoľvek individuálnych problémov sa skúste skontaktovať (čím skôr s cvičiacim, vyučujúcim, Podporným centrom I-23, resp. štúd.oddelením,**
- ak študent dosiahne za semester **[quadtermy+midterm+projekt+DU] aspoň 60 bodov**, automaticky dostáva hodnotenie **A** bez skúšky
- ak študent nazbiera počas semestra **[quadtermy+midterm+DU] < 20** bodov, automaticky dostáva hodnotenie **Fx** okrem nasledujúcich prípadov:



Sylabus

- [18.2.] Úvod do Javy (história a kontext)
- [**pondelok, 22.2. 14:00 F2/F1**] Komponenty jazyka (pre C++ programátora)
- [3.3.] Triedy a objekty (dedenie, ukrývanie, konštruktory a deštruktory)
- [10.3.] Triedy, objekty (pokračovanie)
- [17.3.] Lineárne dátové štruktúry
- [31.3.] Java Collections
- [11.4., 18:10] Midterm [2008...2015]
- [7.4.] Java I/O (výnimky a serializácia)
- [14.4.] Vlákna, konkurentné procesy, jednoduché simulácie v Java Fx
- [21.4.] Vlákna - komunikácia, synchronizácia - pokračovanie
- [28.4.] Java Fx - základné komponenty, spracovanie udalostí
- [5.5.] JavaFx - pokračovanie
- [14.5.] JavaFx - záver, Reflection Model
- [21.5.] Komunikácia medzi aplikáciami



Budúcnosť (eventuálna)

Programovanie (4) = Java úvod

4.semester

<http://dai.fmph.uniba.sk/courses/JAVA/>

→ Vývoj mobilných aplikácií = Android/Java

5.semester

<http://dai.fmph.uniba.sk/courses/VMA/>

Programovacie paradigmy = Go, Haskell, Prolog

5.semester

<http://dai.fmph.uniba.sk/courses/PARA/>

→ Funkcionálne programovanie = Haskell++

6.semester

<http://dai.fmph.uniba.sk/courses/FPRO/>

2.semester MAG



Cieľ kurzu

- oboznámiť sa s jazykom JAVA (syntaxou a sémantikou jednotlivých jazykových konštrukcií)
- ukázať špecifické princípy a vlastnosti jazyka JAVA (keďže o princípoch OOP ste počuli už na dvoch prednáškach, v iných kontextoch)
- byť schopný písať jednoduché aplikácie s GUI
- a v poslednej rade, aj zaprogramovať si ...

Cieľom kurzu nie je:

- písanie aplikácií pre mobilné platformy
 - Android v kurze VMA (<http://dai.fmph.uniba.sk/courses/VMA/>)
 - ... *ale kto si to chce skúsiť, môže v rámci záverečného projektu*
- písanie aplikácií JavaEE
 - Programovanie 5 (<http://dai.fmph.uniba.sk/courses/java2/>)
 - písanie klient-server aplikácií a servletov,
 - návrhové vzory ☹

It would be a tragic statement of the universe if
Java was the last language that swept through.
James Gosling



Úvodná prednáška

dnes bude:

- trochu histórie a filozofie jazyka Java
- neformálny úvod do OOP-jazyka Java (*na príkladoch zo šikmej plochy*)
- základné (numerické) dátové typy
- syntax (niektorých) príkazov

Cvičenie:

- urobiť prvý program (editovanie, kompilácia a spustenie programu),
- uistiť sa, že časť príkazových konštrukcií už poznáme z jazyka C++
- komfortná práca so základnými typmi, int, long, float, char, String, ...

literatúra (vid' linky na stránke predmetu):

- Thinking in Java, 3rd Ed. - 2.kapitola Everything is an Object
(<http://www.ibiblio.org/pub/docs/books/eckel/TIJ-3rd-edition4.0.zip>)
- Naučte se Javu – úvod (<http://interval.cz/clanky/naucte-se-javu-uvod/>)
Naučte se Javu – dátové typy (<http://interval.cz/clanky/naucte-se-javu-datove-typy/>)

OOP jazyky

JAVA nie je zd'aleka prvý O-O programovací jazyk:
(viac sa dozviete napr. na predmete Programovacie paradigmy
<http://dai.fmph.uniba.sk/courses/PARA/>)

- SIMULA, 1960
mala triedy, objekty, dedenie, virtuálne metódy, GC
 - Smalltalk, 1971-80, Xerox PARC
všetko sú objekty, je dynamicky typovaný a interaktívny interpreter
 - C++, 1983, Bell Labs
 - **Java, 1990, Sun Microsystems**
 - 1991, jazyk Oak (neskôr premenovaný na Java)
 - 1993, jazyk Java ako jazyk pre web, WWW
 - 1995, oficiálne predstavenie JAVA
 - Eiffel, 1995,
viacnásobná dedičnosť, generické typy/templates
 - Microsoft Visual J++, J#, C#, .NET,
 - Borland – Delphi, Builder, JBuilder
- ... a dnes už je všetko objektové, len programátori ostali procedurálni*



James Gosling
Unix
Emacs
>15r.SUN
Oracle
Google

terminológia skôr než začnete
"browsovať",
"sťahovať",
"inštalovať"

Základné pojmy

- Java Development Kit (jdk) (<http://java.sun.com/>, <http://www.oracle.com/technetwork/java>)
vývojové prostredie, súbor knižníc a nástrojov (javac - kompilátor, javadoc – generátor dokumentácie, ...)
 - verzie: jdk-8-OS, napr. [jdk-8u31-windows-x64.exe](#)
 - edície: Standard Ed. (SE), Enterprise Ed. (EE), Micro Ed. (ME), ...
- Virtual Machine – interpretér byte kódu s knižnicami / Java Runtime Environment / java plugin do browsera
 - verzie: jre-8-OS, napr. [jre-8u31-windows-x64.exe](#)
- druhy programov, ktoré v prednáške :
 - spomenieme: aplikácie (a applety, možno...),
 - nespomenieme: servlety, midlety, activity, ...
- prostredia pre vývoj - Java Integrated Environment
 - Eclipse (<http://www.eclipse.org/>)
 - NetBeans (<http://www.netbeans.org/>)
 - JBuilder (<http://www.embarcadero.com/products/jbuilder>)
 - IntelliJIdea (<http://www.jetbrains.com/idea/>)

... no a syntax-zvýrazňujúci editor a java-command line kompilátor javac

V rámci prednášky/cvičení používame
JDK 1.8 v prostredí Eclipse Mars

Pre riešenie úloh/projektov môžete
používať iné prostredia. Skúška a
quadtermy sú však v Eclipse pod Win.

An API that isn't comprehensible isn't usable.



Vývojové nástroje (JDK)

JDK 1.0 – nepoužíva sa,

JDK 1.1 – stará Java (zmeny v jazyku),

JDK 1.2 – Java 2 SDK (nové knižnice, Swing,...),

JDK 1.3 – zmeny v API, rýchlejšia,

JDK 1.4 – stabilná,

JDK 1.5 – jazykové úpravy, generics, ...

JDK 1.6 – XML web servisy, JavaDB, monitoring

JDK 1.7 – nové jazykové konštrukcie, ktoré potešia, ale dá sa prežiť bez nich

(<http://code.joejag.com/2009/new-language-features-in-java-7/>)

- underscore konštanty
- switch príkaz podľa reťazca
- try-catch konštrukcia
- type inference v generických konštrukciách

JDK 1.8 – (<https://jdk8.java.net/download.html>)

- funkcionálny koncept, tzv. lambda výrazy
- paralelné triedenie poľa
- Small Virtual Machine < 3MB



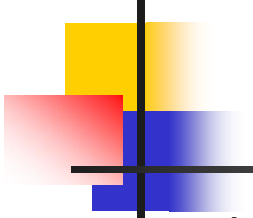
Vývoj programátorských kultúr

Skôr, než sa vrhneme do programovania, krátka rekapitulácia toho, čím programovanie prešlo, z hľadiska programátorskej kultúry

- **Neštruktúrované programovanie**
 - základné (numerické, znakové, reťazcové) typy
- **Štruktúrované programovanie**
 - záznamy a množiny, procedúry a štruktúrované príkazy (cykly, ...)
- **Objektovo-orientované programovanie**
 - triedno-inštančný model
 - polymorfizmus a dedenie
 - dynamická väzba

Tieto veci si rozviníme v úvodných troch prednáškach.
Ilustrované sú na troch nasledujúcich príkladoch.

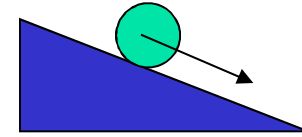
Neštruktúrované programovanie

- 
- do počítača prenášame len jednotlivé parametre entity, ktoré môžeme merať v reálnom svete (napr. rýchlosť, polohu, čas, ...)
 - potrebujeme na to:
 - premenné (realne, celočíselne, ...),
 - hlavný program, event. podprogramy či procedúry
 - základné typy premenných: integer, real, double, boolean, complex, ...
 - polia, 1,2,3-trojrozmerné polia
 - statické dátové štruktúry/typy, dynamické len dĺžka polí

Upozornenie:

nasledujúci text obsahuje malé ilustračné príklady kódu v Jave, bez toho, aby ste čokoľvek o syntaxi jazyka vedeli. Tá príde neskôr. Je to zámer 😊

Neštruktúrované programovanie



```
public class Gulicka1 {
```

```
    public static void main(String[] args) {
```

```
        double x=0.0, y=5.0, fi=0.56;
```

```
        int t;
```

```
        for (t=0; t<10; t++) {
```

```
            x += Math.cos(fi);
```

```
            y -= Math.sin(fi);
```

```
        }
```

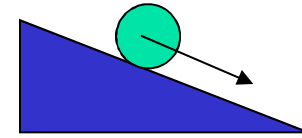
```
    }
```

```
}
```

Aj keď ešte nevieme napísať program, dobrý jazyk by mal byť dostatočne intuitívny na to, aby sme ho vedeli čítať a rozumeli mu (aspoň približne...)

Súbor: [Gulicka1.java](#)

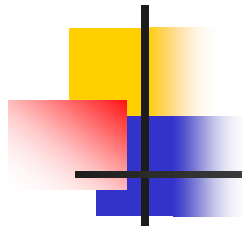
Neštruktúrované programovanie



Ku každej prednáške sú na stránke predmetu umiestnené zdrojové kódy programov, ktoré často obsahujú doplňujúce informácie, komentáre, ... Čítajte ich.

```
/**
 * nestrukturovany priklad
 * @author PB
 */
public class Gulicka1 {
    // definicia hlavneho programu musi zacinat "public static void main(String[] args)"
    public static void main(String[] args) {
        double x=0.0, y=5.0, fi=0.56; // definicia troch realnych premennych s inicializaciou hodnot
        int t;                          // definicia celociselnej premennej cyklu
        for (t=0; t<10; t++) {          // cyklus for t=0 to 9 do
            x += Math.cos(fi);          // priradenie x = x+cos(fi)
            y += Math.sin(fi);          // priradenie y = y+sin(fi)
        }
    }
}
```

Súbor: [Gulicka1.java](#)



Procedúry, knižnice

procedúra - implementácia na inom mieste než použitie
procedúra – abstrakcia (spoločných často používaných častí kódu)
knižnica/package - zoskupenie viacerých procedúr

```
public class Gulicka2 {  
    public static double posunX(  
        double x, double fi) {  
        return x+Math.cos(fi);  
    }  
  
    public static double posunY(  
        double y, double fi) {  
        return y-Math.sin(fi);  
    }  
}
```

```
public static void main(String[] args) {  
    double x=0.0, y=5.0, fi=0.56;  
    int t;  
    for (t=0; t<10; t++) {  
        x = posunX(x, fi);  
        y = posunY(y,fi);  
    }  
}
```



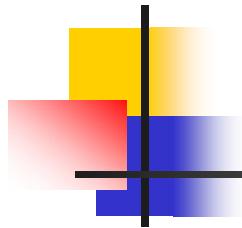
Štruktúrované programovanie

dáta:

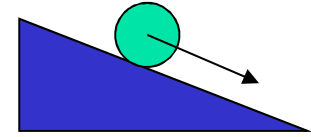
- entita = štruktúra
- štruktúra môže mať viac parametrov
- predstavuje dodefinovaný zložený typ
- štruktúra typu záznam (record/struct/class)
- varianty (case of, union) v jave nie sú
- skladanie štruktúr
- dynamika: statické aj dynamické štruktúry

riadenie:

- štruktúrované príkazy
- procedúry a funkcie s parametrami
- rekurzia



Štruktúrované programovanie



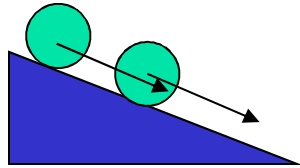
```
public class Gulicka3 {  
    static double x;  
    static double y;  
  
    public static void posun(double fi) {  
        x += Math.cos(fi);  
        y -= Math.sin(fi);  
    }  
}
```

```
public static void main(String[] args) {  
    x=0.0; y=5.0;  
    double fi=0.56;  
    int t;  
    for (t=0; t<10; t++) {  
        posun(fi);  
    }  
}
```

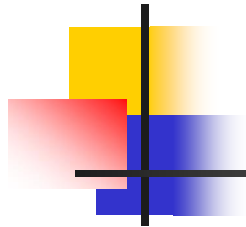


Objektovo-orientované programovanie

- entita obsahuje nielen dáta, ale aj kód (metódy), ktorý s nimi manipuluje
- štruktúra má viac atribútov a metód
- triedno-inštančný prístup:
 - každý objekt vzniká ako/je inštancia triedy
 - trieda definuje jeho atribúty a metódy
 - zložený typ je obohatený na triedu
 - štruktúra je obohatená na objekt
 - z premenných sa stávajú atribúty
 - z funkcií a procedúr metódy
- dynamika: hlavne dynamické štruktúry, statické napr. atribúty triedy



Objekt Gulicka



```
public class Gulicka {  
    double x;  
    double y;  
  
    public Gulicka(double xx,  
                    double yy) {  
        x = xx; y = yy;  
    }  
    public void posun(double fi) {  
        x += Math.cos(fi);  
        y -= Math.sin(fi);  
    }  
}
```

```
public class Gulicka4 {  
    public static void main(String[] args) {  
        Gulicka g = new Gulicka(0.0,5.0);  
        Gulicka h = new Gulicka(1.0,4.0);  
        double fi=0.56;  
        int t;  
        for (t=0; t<10; t++) {  
            g.posun(fi);  
            h.posun(fi);  
        }  
    }  
}
```

trieda Prvy je definovana v súbore Prvy.java



Prvý

hlavička hlav.programu

```
public class Prvy {
```

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

```
}
```

volanie kompilátora

```
javac Prvy.java
```

```
java Prvy
```

```
Ahoj
```

volanie interpretera



Základné celočíselné typy

neexistuje neznamienková verzia *unsigned*

- **byte**
java.lang.**Byte** [8 bitov]
-128 .. 127
- **short**
java.lang.**Short** [16 bitov]
 -2^{15} .. $2^{15}-1$
- **int**
java.lang.**Integer** [32 bitov]
 -2^{31} .. $2^{31}-1$
- **long**
java.lang.**Long** [64 bitov]
MIN_VALUE .. MAX_VALUE



Základné typy

Znaky (Unicode, 16 bitov)

- **char**
java.lang.**Character**

Reťazce

- **String**
java.lang.**String**

Reálne čísla

- **float**
java.lang.**Float**
- **double**
java.lang.**Double**

Logické hodnoty

- **boolean**
java.lang.**Boolean**



Konštanty

Java 7

Notácia s _

514_000

0b1010 – binárne

0xFF_FF

3.1415926535

_8979323846

_2643383279

_5028841971

_6939937510

_5820974944

_5923078164

- Desiatkové: 32,12,....
- Osmičkové: 0126, 015, 01
- Šestnástkové: 0x56,0x1,0xCD,...
- Long int: 123456789123L
- Znakové: 'A','%','\u00E1',
 - '\n' (nový riadok),
 - '\t' (tabulátor),
 - '\\' (backslash),
 - ...
- Reťazcové: " toto je reťazec v Java"
- Logické typu boolean: true, false
- Reálne float, double: 15.8, 7E23, 3.14F,...

Deklarácia premenných a konštánt

```
int    i, j;
char   c;
float  f, g;
int    j = 1;
final int MAX = 10;      // definícia konštanty
...
      MAX = 11;          // chyba
```

```
public class Konstanta {
    public static final int MAX = 10;

    public static void main(String[] args) {
        System.out.println("MAX = " + MAX);
        System.out.println("MAX = " + Konstanta.MAX);
    }
}
```

MAX = 10
MAX = 10



Warm-up

(zamyslite sa pred cvičením – v zostave Cvičenie 1)

1) V ktorých z nasledujúcich možností uvedená konštanta zodpovedá preddefinovanej hodnote daného typu:

- A. `int -> 0`
 - B. `String -> "null"`
 - C. `Dog -> null`
 - D. `char -> '\u0000'`
 - E. `float -> 0.0f`
 - F. `boolean -> true`
-

2) Ktoré z nasledujúcich možností predstavujú korektnú deklaráciu premennej typu `char`:

- A. `char c1 = 064770;`
 - B. `char c2 = 'face';`
 - C. `char c3 = 0xbeef;`
 - D. `char c4 = '\u0022;`
 - E. `char c5 = '\iface';`
 - F. `char c6 = '\uface';`
-

3) Ktoré z nasledujúcich možností predstavujú korektnú deklaráciu premennej typu `float`:

- A. `float f1 = -343;`
- B. `float f2 = 3.14;`
- C. `float f3 = 0x12345;`
- D. `float f4 = 42e7;`
- E. `float f5 = 2001.0D;`
- F. `float f6 = 2.81F;`

4) Ktoré z nasledujúcich možností predstavujú korektnú deklaráciu premennej typu `String`:

- A. `String s1 = null;`
 - B. `String s2 = 'null';`
 - C. `String s3 = (String) 'abc';`
 - D. `String s4 = (String) '\ufeed';`
-

5) Ktoré z nasledujúcich možností predstavujú korektnú deklaráciu premennej typu `boolean`:

- A. `boolean b1 = 0;`
 - B. `boolean b2 = 'false';`
 - C. `boolean b3 = false;`
 - D. `boolean b4 = Boolean.false();`
 - E. `boolean b5 = no;`
-

6) Numerický interval typu `char` je:

- A. `-128 to 127`
- B. `-(215) to (215) - 1`
- C. `0 to 32767`
- D. `0 to 65535`

Java nemá predprocesor a la C++
nehľadáte #ifdef ... #endif



Komentáre

```
public class Komentare {                                // Píšte komentáre, sú zdravé !

    public static void main(String[] args) {
        double ucet;
        int pocetPiv = 5;
        ucet = pocetPiv * 1.0;                          // typický komentár
        System.out.println("Platis = " + ucet);

        ucet = pocetPiv * /* 1.0 */ 1.30; /* 1.0 je za desinku */
        System.out.println("Platis = " + ucet);

    }
}
```

Platis = 5.0
Platis = 6.5

Komentáre pre dokumentáciu

```
/**  
 *  
 */
```

```
/**  
 * príklad s dvomi funkciami (resp. procedurami s vystupnou hodnotou)  
 * @author PB  
 */  
public class Gulicka2 {  
    /**  
     * definicia funkcie posunX  
     * @param x - suradnica gulicky  
     * @param fi - sklon sikmej plochy  
     * @return vrati novu X-ovu suradnicu gulicky  
     */  
    public static double posunX(double x, double fi) {  
        return x+Math.cos(fi);  
    }  
    /**  
     * toto je hlavny program  
     * @param args - argumenty prikazoveho riadku, ktore zatiaľ nevyuzivame  
     */  
    public static void main(String[] args) {  
        double x=0.0, y=5.0, fi=0.56;  
        for (int t=0; t<10; t++) { // definicia premennej cyklu t priamo v cykle  
            x = posunX(x, fi); // volanie funkcie s dvomi argumentami  
            y = posunY(y, fi); // a priradenie vyslednej hodnoty do premennej  
        }  
    }  
}
```

Method Summary

static void	main (java.lang.String[] args) toto je hlavny program
static double	posunX (double x, double fi) definicia funkcie posunX
static double	posunY (double y, double fi) definicia funkcie posunY

Method Detail

posunX

```
public static double posunX(double x,  
                           double fi)
```

definicia funkcie posunX

Parameters:

x - - suradnica gulicky
fi - - sklon sikmej plochy

Returns:

vrati novu X-ovu suradnicu gulicky



javadoc – generátor dokumentácie

Ako písať dokumentáciu

- <http://java.sun.com/j2se/javadoc/>

Kde nájsť dokumentáciu k JDK SE 1.6 (lokálna kópia prístupná aj počas testov a skúšky)

Najbežnejšie tagy

- @author
- @version
- @param
- @return
- @exception
- @see

```
/**
 * priklad programu, ktory cita cele cislo z konzoly do premennej N,
 * na ktoru potom vypise prvych <code>N</code> fibonacciho cisel.
 * <br>
 * Fib.cisla su dane vztahom
 * <br>
 * <ul>
 * <li>fib(1)=0, </li>
 * <li>fib(2)=1, </li>
 * <li>fib(N+2)=fib(N)+fib(N+1)</li>
 * </ul>
 * <br>
 * Pozn.:program pouziva triedu Input ako pomocku na cistanie cisla
 * @author PB
 * @version 2009
 */
```



Výpis na konzolu

- vstup a výstup cez konzolu (a cez dátové streamy) zabezpečuje implicitne viditeľný package java.io
- pre začiatok vystačíme s metódami System.out.print a System.out.println

```
public class Vystup {  
    public static void main(String[] args) {  
        int i = 4;  
        int j = 7;  
        System.out.print("Toto je hodnota premennej i: " + i + "\n");  
        System.out.println("Toto je premenna i: "+i+" a toto j: "+j);  
        System.out.println("Sucet nie je " + i + j);  
        System.out.println("Sucet je " + (i + j));  
    }  
}
```

Toto je hodnota premennej i: 4
Toto je premenna i: 4 a toto j: 7
Sucet nie je 47
Sucet je 11

Súbor: Vystup.java

- nepíšte then
- zátvorkujte logický výraz
- používanie { } nie je chyba ☺



if-then-else

```
if (booleovský výraz)
    príkaz;
else
    príkaz;
```

```
if (d > 0)
    x = d*d;
else
    x = d/2;
```

```
if (i > 0) {
    if (j > 0) {
        j++; i--;
    }
    else {
        i++;
    }
}
```

// { } zložený príkaz, begin-end

// else patrí k najvnútornejšiemu if

podmienový výraz

// príklad: max = (i > j) ? i : j;

(booleovský výraz)?výraz1:výraz2



Priradenie verzus porovnanie

```
float f;                // definícia
f = 3.14;               // inicializácia/priradenie

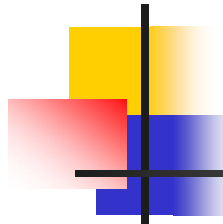
int    j, i = 5;        // definícia s inicializáciou
boolean b = true;

if (i == (j = 5)) {     // priradenie a porovnanie
    System.out.println(i);
}

if (b = (j == 5)) {     // porovnanie a priradenie
    System.out.println(j);
}

i = j = 7;              // j = 7; i = 7;

i += j;                 // i = i + j
```



cykly

while (**booleovský výraz**)
 příkaz;

```
while (N > 0) { N = N-1; A = A+A; }  
while (N-- > 0) { A = A+A; }  
while (N-- > 0) A += A;
```

do
 příkaz;
while (**booleovský výraz**);

```
do {  
    A += A;  
} while (N-- > 0);
```

for (**výraz štart; výraz stop; výraz iter**)
 příkaz;

```
for(int i=0; i<N; i++) { ... }  
for(i=1; i<=N; i++) { ... }  
for(i=N; i>0; i--) { ... }
```




break, continue

break - vyskočenie z najvnútornejšieho cyklu (alebo označeného návěstím)

continue - na začiatok najvnútornejšieho cyklu (alebo označeného návěstím)

```
int i = 0;
while (i++ < N) {
    if (našiel som) break;
}
// našiel som ...
```

```
for(int i = 0; i < N; i++) {
    ...
    if (zlý prvok) continue; // zobor ďalší
    ...
}
```

navestie:

```
for (int n = 0; n < 4; n++) {
    for (int m = 0; m < 2; m++) {
        if (n == 2 && m == 1)
            continue navestie;
        System.out.print(n + "-" + m + " ");
    }
}
```



switch, return

```
switch (citajZnak()) {  
    case 'a' :  
    case 'b' :  
    case 'c' :  
        System.out.print("1");  
        break;  
    case 'd' :  
        System.out.print("2");  
        break;  
    default :  
        System.out.print("3");  
}  
  
return výraz;  
    // result výraz;
```

```
// String-switch je novinka v Java 7  
public static void main(String[] args) {  
    if (args.length == 0) return;  
    switch(args[0]) {  
        case "load":  
            System.out.println("citaj");  
            break;  
        case "save":  
        case "saveAs":  
            System.out.println("pis");  
            break;  
        default:  
            System.out.println("ine");  
    }  
}
```

Súbor: Switch.java

Goto

(sú)Boj „skutočných programátorov“ a „pojedačov koláčov“

E.Dijkstra: *Go To Statement Considered Harmful*, CACM, 1968

F.Rubin: "'GOTO Considered Harmful' Considered Harmful", CACM, 1987

D.Moore: ""'GOTO Considered Harmful' Considered Harmful?," CACM, 1987



[Goto in Java](#)



Operátory ++ a --

```
public class PlusPlus {
```

```
    public static void main(String[] args) {
```

```
        int i = 5, j = 1, k;
```

```
        i++;
```

```
        System.out.println("i = " + i);
```

```
        j = ++i;
```

```
        System.out.println("j = " + j + ", i = " + i);
```

```
        j = i++;
```

```
        System.out.println("j = " + j + ", i = " + i);
```

```
        k = --j + 2;
```

```
        System.out.println("k = " + k + ", j = " + j);
```

```
        i = j % 4;
```

```
    }
```

```
}
```

i = 6

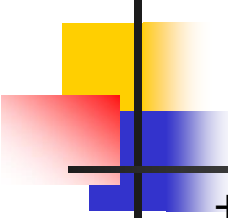
j = 7, i = 7

j = 7, i = 8

k = 8, j = 6
// modulo

i = 2

Skrátená forma, ostatné operátory



+=	a += b;	// a = a+b	
-=	a -= b;	// a = a-b	
*=	a *= b;	// a = a*b	
/=	a /= b;	// a = a/b	// delenie alebo div
%=	a %= b;	// a = a%b	// modulo
... a mnoho ďalších			

== rovný	// a == 0
!= nerovný	// (a != 0) == false
&& log.súčin(boolovské and)	// (a >= 0) && (a <= 0)
log.súčet(boolovské or)	// (a + a == a) (a * a == a)
! log.negácia(boolovské not)	// !(a!=0)
~ bitová negácia	// (~a) == -1
& bitové and	// a & (~a)
bitové or	// a (~a)
^ bitové xor	// a ^ (~a)
<< shift left (<< n je násobenie 2 ⁿ)	// (a+1) << 2
>> shift right (>> n je delenie 2 ⁿ)	// (a+1) >> 1
	// (a-1) >> 4
>>> unsigned right shift	// (a-1) >>> 4

//-1
//268435455
Súbor: Operatory.java



Hádanka

o po íta funkcia quiz ?

Príklady zlých odpovedí:

- n-té prvo íslo
- n^2
- 2^n

```
public static long quiz(int n) {  
    long a = 0, b = 1;  
    if (n <= 0) return -1;  
    for (; n-->0; a += b, b -=a, b =-b);  
    return a;  
}
```



Bitové operácie

&	and
	or
^	xor
<<	shift left
>>	shift right
>>>	unsigned right shift
~	negation

```
byte i = 7 & 9;
```

```
byte i = 7 | 9;
```

```
if (i % 2 == 0) System.out.println(i + " je párne");  
if ((i & 1) == 0) System.out.println(i + " je párne");
```

```
byte stav = 0;  
byte bit2 = 0x4;  
stav |= bit2;  
if ((stav & bit2) == bit2) ...  
stav &= ~bit2;
```

```
// 8-bitový vektor  
//  $4_{16} = 0b100_2$   
// nastav bit 2  
// testuj bit 2  
// zmaž bit 2
```

```
byte x = 5; x <<= 3;  
int x = 256; x >>= 4;  
int x = 16; x >>= 2;  
int x = -16; x >>= 2;
```

```
//  $40_{10} = (101)_2 <<= 3 = (101000)_2$   
//  $16_{10} = (100000000)_2 >>= 4 = (10000)_2$   
//  $4_{10} = (10000)_2 >>= 2 = (100)_2$   
//  $1073741820_{10} = (11111...10000)_2 >>= 2 =$   
//  $(11111...100)_2$ 
```

```
byte i = 7 ^ 5;
```

```
// 2
```

málo kto pozná a používa...

Skrátený súčet, súčin

```
int i, j, k;  
i = 1; j = 2; k = 3;  
if (i == 1 || ++j == 2) k = 4;  
System.out.println("i = " + i + ", j = " + j + ", k = " + k);
```

toto sa nevyhodnotí, lebo $i == 1$
a $true ||$ hocičo je $true...$

$i = 1, j = 2, k = 4$

```
i = 1; j = 2; k = 3;  
if (i == 1 | ++j == 2) k = 4;  
System.out.println("i = " + i + ", j = " + j + ", k = " + k);
```

teraz sa to vyhodnotí

$i = 1, j = 3, k = 4$

```
i = 1; j = 2; k = 3;  
if (i == 2 && ++j == 3) k = 4;  
System.out.println("i = " + i + ", j = " + j + ", k = " + k);
```

toto sa nevyhodnotí, lebo $i != 2$

$i = 1, j = 2, k = 3$

```
i = 1; j = 2; k = 3;  
if (i == 2 & ++j == 3) k = 4;  
System.out.println("i = " + i + ", j = " + j + ", k = " + k);
```

teraz sa to vyhodnotí, aj keď $i != 2$

$i = 1, j = 3, k = 3$



Priority

.	[index]	(typ)	najvyššia
!	++	--	
*	/	%	
+	-		
<<	>>	>>>	
<	<=	>=	>
==	!=		
&			
^			
&&			
?:_			
=	+=	...	najnižšia

Príklady:

`a += (1F/b),` `(a == 0) && (b == 1),` `(c=readChar())!=`n``



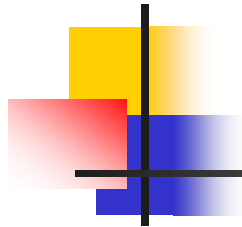
Vstup z konzoly

Vstup nie je natoľko priamočiary, aby sme ho detailne pochopili v prvej prednáške. Preto dočasne používame triedu Input, ktorá sa nachádza v balíku 01_java.zip. Neskôr bude vstupu a výstupu venovaná celá prednáška

```
public class Vstup {  
    public static void main(String[] args) {  
        Input in = new Input();  
        System.out.println("Vase meno:");  
        final String meno = in.nextLine();  
  
        System.out.println("Vas vek:");  
        final int vek = in.nextInt();  
  
        int suma = 0;  
        while (in.hasNextInt())  
            suma += in.nextInt();  
        System.out.println("sucet:"+suma);  
    }  
}
```

Vase meno:
peter
Vas vek:
12
1
2
3
4
5
sucet:15

Súbor: **Vstup.java**



Fibonacci – príklad na cvičenie

```
public class Fibonacci {  
  
    public static void main(String[] args) {  
        Input in = new Input();  
        System.out.println("Zadaj N:");  
        int N = in.nextInt();  
        long a = 1;  
        long b = 0;  
        while (N-- > 0) {  
            System.out.println(b);  
            a = a + b;  
            b = a - b;  
        }  
    }  
}
```

Zadaj N:

10

0

1

1

2

3

5

8

13

21

34



Pascalov trojuholník

Napište program, ktorý spočíta a vypíše kombinačné čísla v tvare približne:

```
public class Pascal {  
    public static void main(String[] args) {  
        for(int n=0; n < 6; n++) {  
            for(int k=n; k<5; k++)  
                System.out.print("\t");  
            System.out.print("1");  
            for (int k = 0, a=1; k <n; k++) {  
                a = a*(n-k)/(k+1);  
                System.out.print("\t\t" + a);  
            }  
            System.out.println();  
        }  
    }  
}
```

(Note: The original image contains a green comment line: `// C(n,k+1) = C(n,k)(n-k)/(k+1)`)*

1
1 1
1 2 1
1 3 3 1



Záver

Cieľom úvodnej prednášky s cvičeniami je aby ste vytvorili váš prvý program v jazyku JAVA, v prostredí Eclipse.

Prostriedky, ktoré zatiaľ poznáme, sú:

- základné (číselne) typy, definovanie premenných a konštánt,
- modul s hlavným programom bez procedúr-metód,
- základné riadiace príkazy vetvenia a cyklu,
- primitívna forma výstupu hodnoty na konzolu,
- vstup z konzoly s pomocnou barličkou (Input.java),
- komentáre –
pomôžu nielen vám, ale aj cvičiacim pri hodnotení vašich kódov