

# 1-AIN-172:

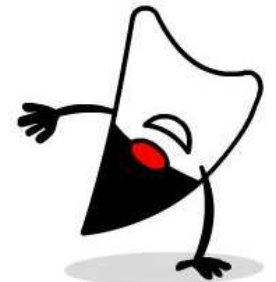
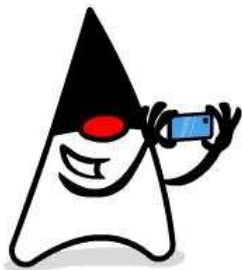
## Programovanie (4)

(alias JAVA pre *už skúsených* C++ programátorov)

Peter Borovanský  
KAI

[borovan@ii.fmph.uniba.sk](mailto: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>

- Pondelok, 8:10, 2hod **Programovanie 4**

## **Cvičenie:**

- Utorok+Streda, 16:30, H3/H6
- J-team:
  - Alexander Šimko
  - Matej Fandl, Lukáš Gajdošech, Jozef Kubík, Daniel Kyselica

## **Používame:**

- **systém LIST:** <https://list.fmph.uniba.sk>
- **MS-Teams:** wbfuuuv
- **gitHub:** <https://github.com/Programovanie4>

## **Konzultačné hodiny:**

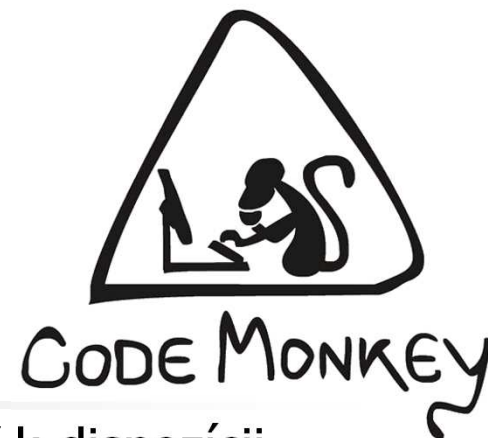
- MS Teams call - **kedykoľvek po e-dohode s vyučujúcim** 😊 😊 😊
- **Kontakt [všetci cvičiaci a ja]:** [prog4java@lists.dai.fmph.uniba.sk](mailto:prog4java@lists.dai.fmph.uniba.sk)

A	114-...
B	100-113
C	86-99
D	72-85
E	68-71
Fx	...-67

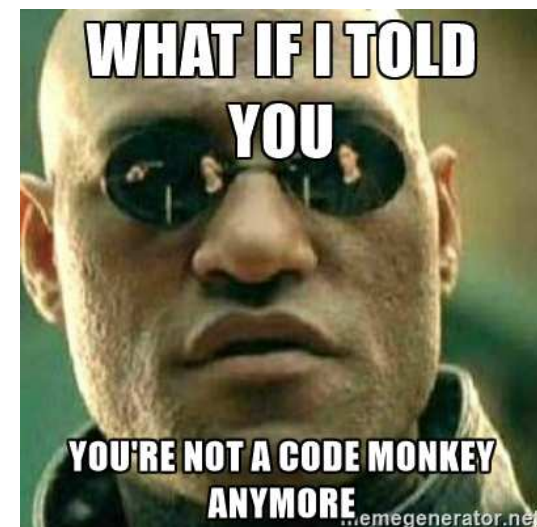
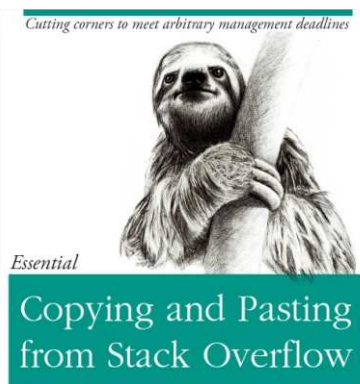
# Hodnotenie

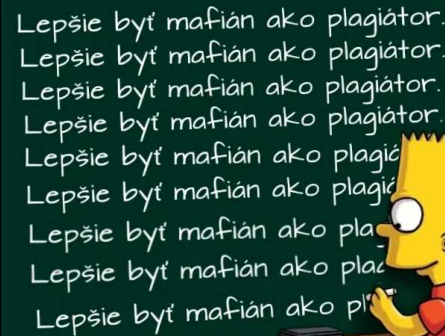
- **DÚ**(12x3), **cvičenia**(10x1.5), **quadt**(2x15), **midt**(25), **projekt**(15), **skúška**(30)
- **midterm** je online test v jedinečnom termíne **18.4. 18:10**, nedá sa opakovať,
- **dva quadtermy** sú online testy *počas riadnych cvičení*, tiež sa nedajú opakovať,
- **cvičenia sú bodované**, riešenia cvičení môžete odovzdávať do piatku polnoci,
- **skúška** sa hodnotí len, ak študent má z nej aspoň {20-cvičenia bodov},
- cvičenia končia povinnou **domácou úlohou**, ktorej elektronické odovzdanie sa očakáva do termínu cca 10 dní (nasledujúca nedeľa polnoc),
- semestrálny projekt je **nutná** podmienka ku skúške (musí byť uznaný cvičiacim pred termínom skúšky), témy projektov budú zverejnené v apríli, 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 28.2. 14:30, záujemci sa prihlasujú e-mailom,
- **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  **$\geq 100$  bodov**, automaticky dostáva hodnotenie **A** bez skúšky (projekt musí mať aj tak)
- ak študent nazbiera počas semestra  **$< 50$  bodov**, automaticke hodnotenie **Fx**.

# Filozófia kurzu



- programátor pri práci potrebuje internet, budete ho mať k dispozícii
- riešenie akejkolvek úlohy **musí byť vaše riešenie**
- ak riešenie, časť, nejakej úlohy nájdete všeobecne dostupné na internete:
  - takúto úlohu chápeme jako nešťastne zadanú, ale občas sa to „podarí“...
  - a použijete kód, **musíte** uviesť http-link na zdroj,
  - inak sa to vníma ako opisovanie, autora nepenalizujeme ☺
- Pravidlo „zdravého sedliackeho rozumu“ sa používa v akýchkoľvek sporných prípadoch nepokrytých pravidlami; ak zlyhá, rieši štúdijné; ale nestáva sa to  
Príklad: Ak Jožo začne vešať svoje riešenia na web, iný kolega ich nemôže použiť jako svoje riešenia, ani ak uvedie presný link na Jožove riešenie



An analog clock with a red frame and a white face. The hour hand points to 12 and the minute hand points to 3, indicating the time is 3:00.

```
def vypis_rastuce(n):
    def generuj(i):
        for j in range(n):
            if j >= zoz[i-1]:
                zoz[i] = j
            if i == n-1:
                print(*zoz)
            else:
                generuj(i + 1)
    zoz = [0] * n
    generuj(0)
##1
##vypis_rastuce(2)
##vypis_rastuce(3)

def vsetky(mnozina):
    def generuj(slovo):
        for i in mnozina:
            slovo.append(i)
            if len(slovo) == len(mnozina):
                print(*slovo)
            else:
                generuj(slovo)
            slovo.pop()
    slovo = []
    generuj(slovo)
##2
##vsetky({'x', 'y'})
##vsetky(set('ahoj'))

def vsetky_len_raz(mnozina):
    def generuj(slovo):
        for i in mnozina:
            slovo.append(i)
            if len(slovo) == len(mnozina):
                if len(set(slovo)) == len(mnozina):
                    print(*slovo)
            else:
                generuj(slovo)
            slovo.pop()
    slovo = []
    generuj(slovo)
##3
##vsetky_len_raz({'a', 'k', 'm'})

def vsetky_zoznam(mnozina):
    def generuj(slovo):
        for i in mnozina:
            slovo.append(i)
            if len(slovo) == len(mnozina):
                z.append(''.join(slovo))
```

# Prompt Engineering

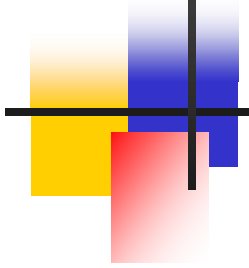


- bezpochyby nová vznikajúca profesia
- s zatiaľ nejasnou budúcnosťou
- preto vzniká prirodzená otázka - chcem byť
  - programátor
  - prompt inžinier
- je asi veľmi naivné si myslieť, že tohto kurzu sa to nedotkne
- osobne stále dúfame, že naše zadania budú lepšie/zaujímavejšie/hlbšie ako súčasné schopnosti chat GPT-3, ale ... jeden nikto nevie (experiment)
- záver: použitie chat-GPT-like riešenia deklarujte v riešení
  - zaujíma nás to, už len pre informáciu, koľkí chcete vyštudovať prompt engineering

## Dôležité upozornenia:

- typicky Quadterm neposkytuje časový komfort si chatovať s AI
- midterm je na papieroch, bez internetu
- akékoľvek odovzdané riešenie podlieha pravidlu „Podanie vysvetlenia“, t.j. študent-autor musí riešeniu dostatočne rozumieť, nech je zdroj akýkoľvek. Inak sa riešenie nehodnotí.





When CS students couldn't find proper code from StackOverflow



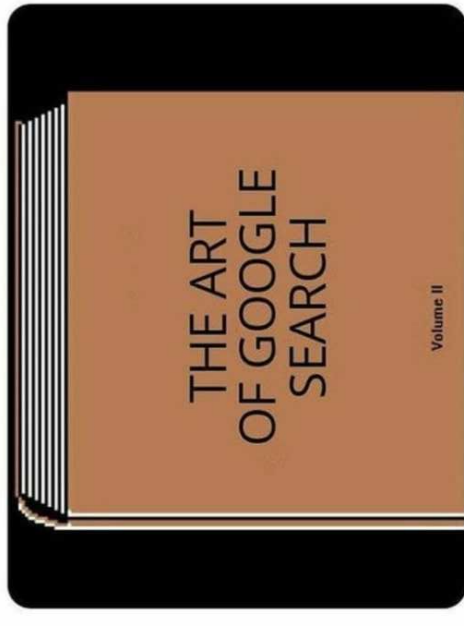
"When you remember it was your own decision to do **Programming**"



#SQUIDGAME



Can you recommend a good book about programming for beginners?



# Testy

- prvá polovica kurzu používa automatické testy s automatickým bodovaním
- zrejme sa objaví syndróm *Works on my machine*
- v histórii sa to už stalo



**See No Evil**





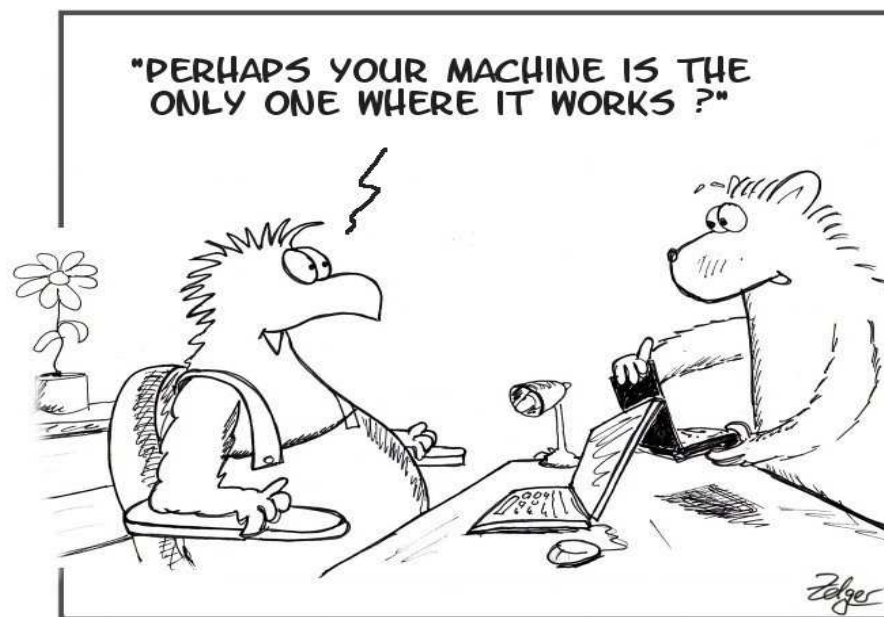
# Works on my machine

Often developers and testers are using their own machines for developing and testing software. The local environment can look different, have different tools installed, even different libraries

It's not so strange to hear someone say  
**"but it works on my computer".**

## **Základné pravidlo:**

- L.I.S.T. aj každé zadanie či test môže mať chybu,
- chyby, na ktoré nás upozorníte, oceňujeme bodom,
- ale nakoniec vás boduje L.I.S.T. nie váš domáci komptuter



It works on my machine

# Zmysel pre študentský humor a kreativitu - 2022

---

**Keď si ledva dokončil prvé Java  
cvičenia a už sú ďalšie prémie na liste**



# Cieľ kurzu



Java is C++ without the guns, knives,  
and clubs

— James Gosling —

AZ QUOTES

- 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 (JavaFx)
- a v neposlednej rade, aj **zaprogramovať si ...**

## Cieľom kurzu nie je:

- úplné programátorské základy (veď už máte za sebou 3 semestre)
- 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
  - Pokročilé programovanie v JavaEE, <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

"If I were to pick a language  
to use today other than Java,  
it would be **Scala**"

James Gosling



dnes bude:

- trochu málo z histórie jazyka Java
- základné (numerické) dátové typy
- syntax (niektorých) príkazov
- polia

Cvičenie:

- získať prvé body za cvičenie, pretlačiť riešenie cez automatické testy
- 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, ...

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



<https://www.youtube.com/watch?v=TJ6XHroNewc>

James Gosling  
Unix  
Emacs  
>15r.SUN  
Oracle  
Google  
...  
Liquid  
Robotics

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*





# OOP



The best way to predict the future is to invent it.

(Alan Kay)

- 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

<https://www.youtube.com/watch?v=9KivesLMncs>



# OOP historia

(Uncle Bob Martin)



## Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.  
Through this work we have come to value:

Individuals and interactions over processes and tools  
Working software over comprehensive documentation  
Customer collaboration over contract negotiation  
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.



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

(primitívne)

---

neexistuje neznamienková verzia *unsigned*

Všetky začínajú malým písmenom (primitívne)

- **byte**

java.lang.**Byte** [8 bitov]

-128 .. 127    rozsah Byte.MIN\_VALUE .. Byte.MAX\_VALUE

- **short**

java.lang.**Short** [16 bitov]

$-2^{15}$  ..  $2^{15}-1$     rozsah Short.MIN\_VALUE .. Short.MAX\_VALUE

- **int**

java.lang.**Integer** [32 bitov]

$-2^{31}$  ..  $2^{31}-1$     rozsah Integer.MIN\_VALUE..Integer.MAX\_VALUE

- **long**

java.lang.**Long** [64 bitov]

rozsah Long.MIN\_VALUE .. Long.MAX\_VALUE



# Základné typy

---

*Znaky (Unicode, 16 bitov)*

- **char**

java.lang.**Character**

*Reálne čísla*

- **float**

java.lang.**Float**

- **double**

java.lang.**Double**

*Logické hodnoty*

- **boolean**

java.lang.**Boolean**

*Ret'azce*

- **String**

java.lang.**String**





# 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 retazec 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)

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.  $-(2^{15})$  to  $(2^{15}) - 1$
- C. 0 to 32767
- D. 0 to 65535

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

# Komentáre

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

    public static void main(String[] args) {
        double ucet;
        int pocetPiv = 5;
        ucet = pocetPiv * 1.90;
        System.out.println("Platis = " + ucet);

        ucet = pocetPiv * /* 1.90 */ 2.50; /* 1.90 je za desinku */
        System.out.println("Platis = " + ucet);

    }
}
```

Platis = 9.5  
Platis = 12.5

# Komentáre pre dokumentáciu

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

```
/**  
 * príklad s dvomi funkciami (resp. procedurami s vystupnou hodnotou)  
 * @author PB  
 */  
public class Gulicka2 {  
    /**  
     * definícia 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 hlavný program  
     * @param args - argumenty príkazového riadku, ktoré zatiaľ nevyužívame  
     */  
    public static void main(String[] args) {  
        double x=0.0, y=5.0, fi=0.56;  
        for (int t=0; t<10; t++) { // definícia premennej cyklu t priamo v cykle  
            x = posunX(x, fi);      // volanie funkcie s dvomi argumentami  
            y = posunY(y,fi);      // a priradenie výslednej hodnoty do premennej  
        }  
    }  
}
```

## Method Summary

static void	<a href="#">main</a> (java.lang.String[] args) toto je hlavný program
static double	<a href="#">posunX</a> (double x, double fi) definícia funkcie posunX
static double	<a href="#">posunY</a> (double y, double fi) definícia funkcie posunY

## Method Detail

### posunX

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

definícia funkcie posunX

#### Parameters:

x - suradnica gulicky  
fi - sklon sikmej plochy

#### Returns:

vrati novu X-ovu suradnicu gulicky





# javadoc – generátor dokumentácie

Ako písať dokumentáciu

- <http://www.oracle.com/technetwork/articles/java/index-137868.html>
- Kde nájsť dokumentáciu k JDK SE

Najbežnejšie tagy

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

Komentáre môžete HTML – naformátovať:

```
/**
 * príklad programu, ktorý číta celé číslo z konzoly do premennej N,
 * na ktorú potom vypíše prvých <code>N</code> fibonacciových čísel.
 * <br>
 * Fib. čísla sú dane vzťahom
 * <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 používa triedu Input ako pomocku na získanie čísla
 * @author PB
 * @version 1.1
 */
```



# 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) {  
        var int i = 4;  
        var 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

- 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

podmienený 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; // zober 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



# Switch expression

---

```
var day = ...;
int numLetters;
switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        numLetters = 6;
        break;
    case TUESDAY:
        numLetters = 7;
        break;
    case THURSDAY:
    case SATURDAY:
        numLetters = 8;
        break;
    case WEDNESDAY:
        numLetters = 9;
        break;
    default:
        throw new IllegalStateException("Wat: " + day);
}
```

```
var numLetters = switch(day) {
    case MONDAY, FRIDAY, SUNDAY -> 6;
    case TUESDAY                 -> 7;
    case THURSDAY, SATURDAY      -> 8;
    case WEDNESDAY               -> 9;
    default -> throw new IllegalStateException("Wat: " + day);
};
```

# 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' Considered Harmful?," CACM, 1987



[Goto in Java](#)

# Skrátená forma, ostatné operátory

`+=`      `a += b;`  
`-=`      `a -= b;`  
`*=`      `a *= b;`  
`/=`      `a /= b;`  
`%=`      `a %= b;`  
... a mnoho ďalších

`// a = a+b`  
`// a = a-b`  
`// a = a*b`  
`// a = a/b`  
`// a = a%b`

`// delenie alebo div`  
`// modulo`

`==` rovný  
`!=` nerovný  
`&&` log.súčin(boolovské and)  
`||` log.súčet(boolovské or)  
`!` log.negácia(boolovské not)  
`~` bitová negácia  
`&` bitové and  
`|` bitové or  
`^` bitové xor  
`<<` shift left (`<< n` je násobenie  $2^n$ )  
`>>` shift right (`>> n` je delenie  $2^n$ )  
  
`>>>` unsigned right shift

`// a == 0`  
`// (a != 0) == false`  
`// (a >= 0) && (a <= 0)`  
`// (a + a == a) || (a * a == a)`  
`// !(a!=0)`  
`// (~a) == -1`  
`// a & (~a)`  
`// a | (~a)`  
`// a ^ (~a)`  
`// (a+1) << 2`  
`// (a+1) >> 1`  
`// (a-1) >> 4`  
`// (a-1) >>> 4`



# Priority

---

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

Príklady:

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





# Hádanka

---

Čo počíta funkcia quiz ?

Príklady zlých odpovedí:

- n-té prvočíslo
- $n^2$
- $2^n$

```
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
```

# Polia jednorozmerné

- *typ[]* – je typ 1-rozmerného poľa
- **new** *typ[size]* – vytvorenie/alokácia
- *pole.length* – dĺžka poľa
- *pole[i]* – indexovanie poľa
- polia majú VŽDY indexy 0..N-1

```
public class Jednoduche {
```

```
    public static void main(String[] args) {  
        final int MAX = 20;
```

```
        // int[] poleInt;
```

```
        // poleInt = new int[MAX];
```

```
        int[] poleInt = new int[MAX];
```

```
        var poleInt = new int[MAX];
```

```
        for (int i = 0; i < poleInt.length; i++) {
```

```
            poleInt[i] = i + 1;
```

```
            System.out.print(poleInt[i] + " ");
```

```
        } // for
```

```
    } // main
```

```
} // class
```

// konštanta – veľkosť poľa

// definícia poľa

// vytvorenie poľa

// definícia poľa s vytvorením

// definícia poľa s vytvorením

// i < MAX

// inicializácia poľa

typ elementu poľa



# Dobré rady

(kuchárka začiatčníka)

Napriek tomu, že nasledujúce rady sú kus za okrajom samozrejmosti, dovoľujem si ich uviesť (pre vaše dobro).

- ak je len trochu možné, vytvorte/alokujte pole ZÁROVEŇ s jeho deklaráciou. Predpokladá to, že v mieste deklarácie poľa poznáte jeho veľkosť. Ušetríte si chyby, keď píšete do nevytvoreného poľa.  
inak: deklarácia ***int[] prvocisla*** žiadne pole nevytvorí. Jediné, čo urobí, že existuje null-referencia/smerník ***prvocisla***, ktorý by chcel ukazovať na pole.
- ak to je možné, inicializujte pole hneď, ako ho deklarujete. Bonusom je, že sa vám aj automaticky vytvorí, príklad  

```
int[] prvocisla = { 2, 3, 5, 7, 11, 13, 19 };           // má dĺžku 7, indexy 0..6  
var prvocisla = new int[]{2, 3, 5, 7, 11, 13, 19};
```
- pole dĺžky N nikdy nebude mať iné indexy ako 0..(N-1).  
ešte inak: `pole[pole.length]` vždy skončí s **ArrayIndexOutOfBoundsException**.
- najprirodzenejší cyklus pre pole je `for(int i=0; i<pole.length; i++) ...`  
ešte inak: **pascalistický zlozvyk** `for(int i=1; i<=pole.length; i++)` je kandidátom na **ArrayIndexOutOfBoundsException**



# Polia v Java vs. C++

(porovnanie pre C++ programátora)

- v C++ po deklarácii poľa `int P[100]` sa vám pole automaticky naalokuje
- v Java toto `int[] P` je deklarácia a toto `P = new int[100]` alokácia
- v Java aj C++ pole inicializujete podobne `int P[] = { 1, 2, 3, 4 },`  
`int[] P = { 1, 2, 3, 4 }`
- v Java sa vytvorené pole inicializuje hodnotami
  - 0 pre číselné typy,
  - `'\u0000'` pre char,
  - `false` pre boolean,
  - `null` iné
- v Java sa nedá indexovať za hranice poľa, kontroluje hranice
- pole je referenčný typ v Java aj C++
- `pole1 = pole2;` je priradením referencií nie kopírovanie polí
- ak potrebujeme kopírovať poľa:
  - C++: `void*memcpy(void *dest, void *source, size_t num)`
  - Java: `System.arraycopy(src, srcPos, dst, dstPos, count)`
  - `dest = Arrays.copyOf(src, count)`

# Polia dvojrozmerné

- *typ*[][] – je typ 2-rozmerného poľa,
- *pole*[i,j] píšeme ako *pole*[i][j],
- *new typ*[M][N] vytvorí pole MxN

- java nemá klasické viacrozmerné polia (matice),
- viacrozmerné polia môžu byť „zubaté“ (jagged)

```
public class Dvojite {
```

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

```
        var int[][] a = new int[4][];
```

// hlavné pole

```
        for (var int i = 0; i < a.length; i++) {
```

```
            a[i] = new int[i + 1];
```

// podpole

```
            for (var int j = 0; j < a[i].length; j++) {
```

```
                a[i][j] = i * 10 + j;
```

```
                System.out.print(a[i][j] + " ");
```

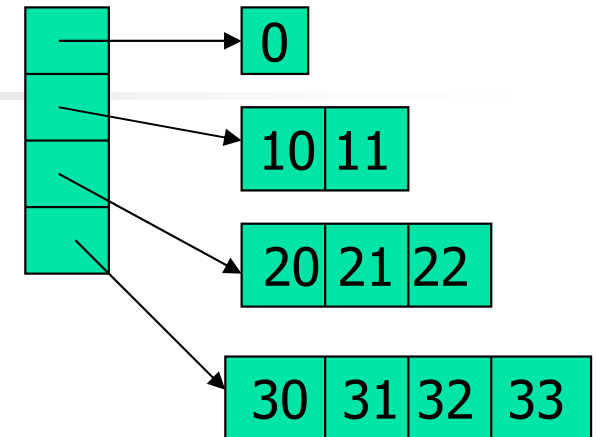
```
            } // for
```

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

```
        } // for
```

```
    } // main
```

```
} // class
```

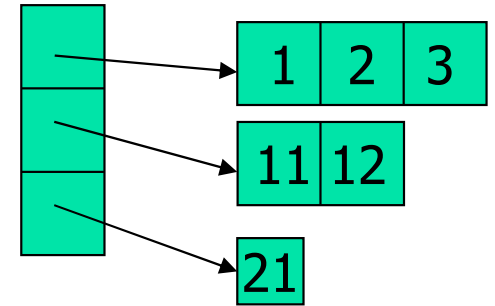


```
0
10 11
20 21 22
30 31 32 33
```



# Inicializácia poľa

(jagged array – štrbavé pole)



- inicializácia dvojrozmerného poľa

```
int[][] a = { {1, 2, 3},  
              {11, 12},  
              {21} };
```

```
a[0][0] = 1; a[0][1] = 2; a[0][2] = 3; a[0].length == 3  
a[1][0] = 11; a[1][1] = 12; a[1].length == 2  
a[2][0] = 21; a[2].length == 1  
a.length == 3
```

- vytvorenie 3-rozmernej matice matice 5x5x5

```
int[][][] d = new int [5][5][5];           // definícia s vytvorením
```

- vytvorenie 2-rozmernej matice "matice" 5x5, ktorej prvky sa vytvoria neskôr

```
int [][][] e = new int[5][5][];  
e[0][1] = new int [8];           ... ok
```

- nesprávne vytvorenie

```
int[][][] f = new int[5][][5]  
f[0]?[1] = new int[8]
```

```
.... Chyba - nemôžem vytvoriť "maticu", ktorej  
.... druhý rozmer nepoznám ale tretí poznám
```

pascalistu poznáš podľa  
ArrayIndexOutOfBoundsException: N,  
kde N je dĺžka jeho poľa

# Polia a cykly

```
final static int MAX = 100;  
public static void main(String[] args) {  
    char[] poleChar = new char[MAX];
```

- `for (int i = 0; i < poleChar.length; i++) { . . . }` // for-to-do
- `for (int i = MAX-1; i >= 0; i--) { . . . }` // for-downto-do
- `int j=MAX;` // while  
`while (j-- > 0) { . . . }`
- `int i=0;` // do-while  
`do { . . .`  
`} while (++i < MAX);`
- `for (char ch:poleChar) System.out.println(ch);` // for-each  
`for (char ch:string.toCharArray()) System.out.println(ch);` // for-each

*for (typPrvkuPola prvokPola:pole) tu vidím prvokPola, neviem jeho index*  
// prechádza postupne prvky poľa bez toho, aby sme vedeli ich index

# Triedy java.util.Arrays, java.lang.System

užitočné statické metódy na prácu s poľami

```
import java.util.Arrays;                                // používam triedu z balíka java.util

int[] a = new int[10];                                  // pole primitívneho typu int
Arrays.fill(a, -1);                                     // vyplň pole nulami, memset
System.arraycopy(a, 11, b, 3, 7);                       // kópia od a[11]->b[3] 7 prvkov
                                                         // memcpy

String[] s = {"janko", "marienka", "jozko", "mracik"};
String[] s_copy = new String[4];
System.arraycopy(s, 0, s_copy, 0, s.length);           // kópia poľa
Arrays.sort(s);                                          // triedenie poľa
for(String elem:s) System.out.print(elem+",");          // janko,jozko,marienka,mracik,
                                                         // binárne vyhľadávanie v utriedenom poli

System.out.println(Arrays.binarySearch(s, "sandokan")); // nenachádza sa: -5
System.out.println(Arrays.binarySearch(s, "marienka")); // nachádza sa: 2

Arrays.equals(s, s_copy);                               // porovnanie polí- false
```

# Kvíz pre C++ programátora

Čo spraví nasledujúci program

```
#include <stdio.h>
void main() {
    int a[][] = { { 1,2,3 }, { 11, 12 }, { 21 } }; }
> gcc test.c
test.c:4: error: array type has incomplete element type
```

a čo tento:

```
#include <stdio.h>
void main() {
    int a[][3] = { { 1,2,3 }, { 11, 12 }, { 21 } };
    printf("%d\n", sizeof(a[0])/sizeof(int));
    printf("%d\n", sizeof(a[1])/sizeof(int));
    printf("%d\n", sizeof(a[2])/sizeof(int));
> gcc test.c
> a.out
3
3
3
```

Poučenie:  
medzi poliami v C++ a Jave  
sú subtilné rozdiely



# Bubble sort

Buble sort je bezpochyby najobľúbenejší triediaci algoritmus medzi študentami.

•ale aj ten možno pokaziť, vid' Chyba1, Chyba2, Chyba3, ...

```
public class BubbleSort {  
  
    public static void main(String[] args) {  
        int[] a = {4,5,2,12,1,2,3};  
        for (int i = 0; i < a.length ; i++) {  
            for (int j = a.length-1; j>i ; j--) {  
                if (a[j-1] > a[j]) {  
                    int temp = a[j];  
                    a[j] = a[j-1];  
                    a[j-1] = temp;  
                } // if  
            } // for  
        } // for  
        for (int elem:a)  
            System.out.println(elem);  
    }  
}
```

// cyklus for-to-do  
// cyklus for-downto-do  
// cyklus for-each-element

1  
2  
2  
3  
4  
5  
12



# 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();  
        }  
    }  
}
```

```
      1  
    1 2 1  
  1 3 3 1  
1 4 6 4 1
```

//  $C(n,k+1) = C(n,k) * (n-k) / (k+1)$



# Záver

---

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

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,
- komentáre –  
pomôžu nielen vám, ale aj cvičiacim pri hodnotení vašich kódov