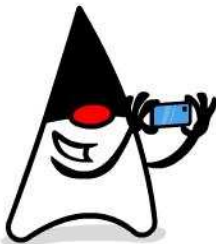
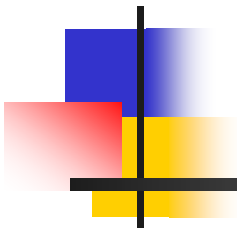


# 1-AIN-172:

## Programovanie (4)

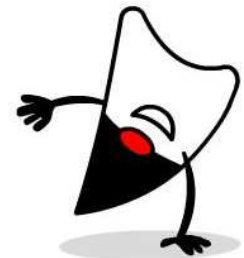
(alias JAVA pre C++ programátorov)



Peter Borovanský  
KAI, I-18

[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>

- Streda, 9:50, 2hod, F2    **Programovanie 4**

## **Cvičenia:**

- Štvrtok, 13:10, H6 (Peter Gergel' / Peter Borovanský)
- Štvrtok, 13:10, H3 (Juraj Holas / Peter Borovanský)
- domáce úlohy opravujú Lukáš Gajdošech a Jozef Kubík

**Používame systém LIST:** <https://list.fmph.uniba.sk>

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

**GitHub:** <https://github.com/Programovanie4>

## **Konzultačné hodiny:**

- Štvrtok 12:00-13:00
- **kedykoľvek po e-dohode s vyučujúcim** 😊 😊 😊
- <https://calendly.com/borovansky>

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



# Hodnotenie

- **DÚ**(12x3)+**quadtermy**(2x15)+**midterm**(25)+**projekt**(15)= $\Sigma$ 106, **skúška**(30),
- **midterm** je písomný test v jedinečnom termíne **16.4.18:00**, nedá sa opakovať,
- **dva quadtermy** sú testy pri počítačoch v terminálke počas riadnych cvičení,
- **cvičenia sú povinné**, akceptujú sa 3 absencie za semester, žiadne PN-ky...,
- **skúška** sa hodnotí len, ak študent má z nej aspoň 10 bodov,
- cvičenia končia povinnou **domácou úlohou**, ktorej elektronické odovzdanie sa očakáva do termínu cca 7-8 dní,
- 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 28.2. 13:10, záujemci sa prihlasujú e-mailom do 27.2.
- 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 **[quads+mid+projekt+DÚ]  $\geq$  95 bodov**, automaticky dostáva hodnotenie **A** bez skúšky (prémie nepočítame, nie je to bug)
- ak študent nazbiera počas semestra **[quads+mid+projekt+DÚ]  $<$  50 bodov**, automaticky dostáva hodnotenie **Fx**.

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

# Hodnotenie – Prípadi

- **DÚ**(12x3)+**quadtermy**(2x15)+**midterm**(25)+**projekt**(15)= $\Sigma$ 106, **skúška**(30),
- ak študent dosiahne za semester [**quads+mid+projekt+DÚ**]  $\geq$  **95 bodov**, automaticky dostáva hodnotenie **A** bez skúšky
- ak študent nazbiera počas semestra [**quads+mid+projekt+DÚ**]  $<$  **50** bodov, automaticky dostáva hodnotenie **Fx** okrem nasledujúcich prípadov: ...

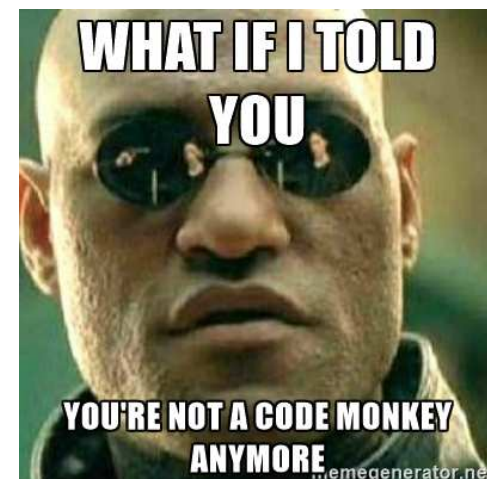
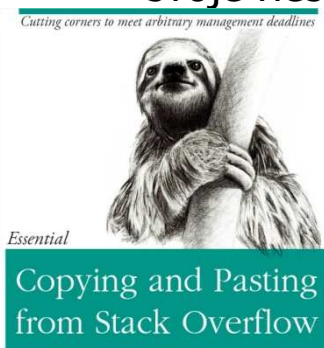
Quads/30	Mid/25	Projekt/15	DÚ/36	$\Sigma$	prémie	$\Sigma$	skúška
3+10	12	6	15	46	36 (!)	-	- 
13+14	22	14	30	93	16	109	- B
13+14	22	14	30	93	16	109	18 A
14+15	24	15	32	100	...		- A 
7+12	15	10	20	64	5	69	- E
7+12	15	10	20	64	5	69	13 D

# Práca počas kurzu



CODE MONKEY

- programátor pri práci potrebuje internet, budete ho mať k dispozícii
- „priateľ na telefóne“ je s **nulovou toleranciou**, aj keď sa to zle dokazuje...
- riešenie akejkoľvek úlohy musí byť vaše
- ak riešenie, časť, nejakej úlohy nájdete všeobecne dostupnú na internete:
  - takúto úlohu chápeme ako nešťastne zadanú, ale občas sa to „podariť“...
  - 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; nestáva sa to...  
napr. Ak Jožo začne vešať svoje riešenia na web, iný kolega ich nemôže použiť ako svoje riešenia, ani ak uvedie presný link na Jožove riešenie



# Testy

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





# Works on my machine

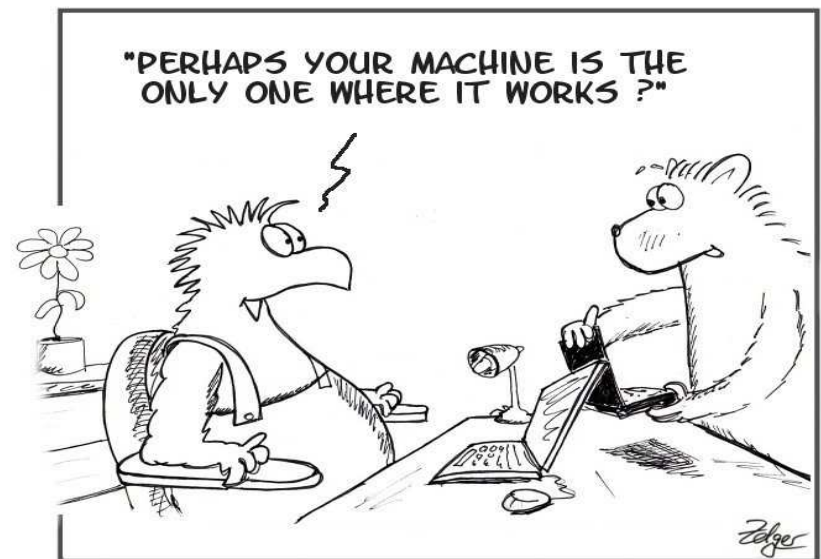
Often developers and testers are using their own machines for developing and testing software. The local environment can look different, have different tools installed, even different libraries

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



# Sylabus



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

— James Gosling —

AZ QUOTES

- [22.2.] Úvod do Javy (história a kontext)
- [28.2.] Komponenty jazyka (pre C++ programátora)
- [7.3.] Triedy a objekty (dedenie, ukrývanie, konštruktory a deštruktory)
- [14.3.] Triedy, objekty (pokračovanie)
- [21.3.] Parametrický polymorfizmus - na lineárnych dátových štruktúrach
- [28.3.] Java Collections
- [4.4.] Java I/O (výnimky a serializácia)
- [11.4.] Vlákna, konkurentné procesy, jednoduché simulácie v Java Fx
- [17.4., 18:00] Midterm [ 2008...2015, 2016, 2017 ]
- [18.4.] Vlákna - komunikácia, synchronizácia - pokračovanie
- [25.4.] Java Fx - základné komponenty, spracovanie udalostí
- [2.5.] JavaFx - pokračovanie
- [9.5.] JavaFx - záver
- [16.5.] Java Reflection Model, Java (1.8) Functional



# Budúcnosť

(eventuálna)



Programovanie (4) = Java úvod

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

4.semester

→ Vývoj mobilných aplikácií = Android/Java-Kotlin

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

5.semester

Programovacie paradigmy = Go, Haskell, Scala

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

5.semester

→ Funkcionálne programovanie = Haskell++

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

2.semester MAG

# Cieľ kurzu



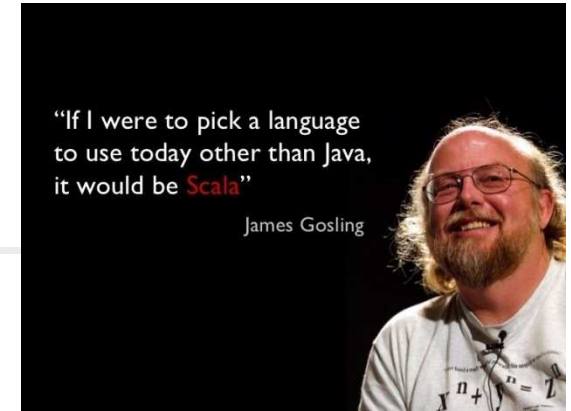
- oboznámiť sa s jazykom JAVA (syntaxou a sémantikou jednotlivých jazykových konštrukcií)
- ukázať špecifické princípy a vlastnosti jazyka JAVA (keďže o princípoch OOP ste počuli už na dvoch prednáškach, v iných kontextoch)
- byť schopný písať jednoduché aplikácie s GUI (JavaFx)
- a v neposlednej rade,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

## Cieľom kurzu nie je:

- úplné programátorské základy (ved' 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



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, ...

literatúra (vid' linky na stránke predmetu):

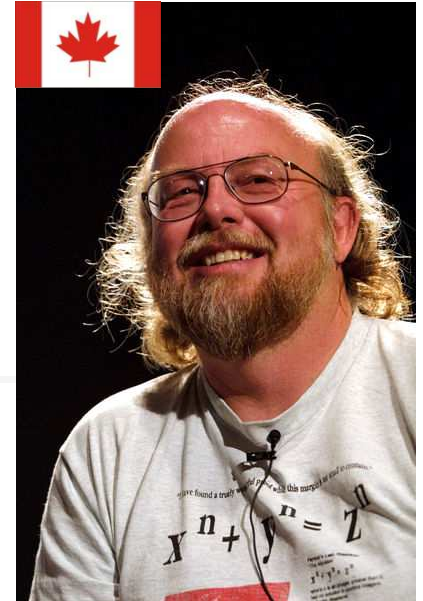
- Thinking in Java, 3rd Ed. - 2.kapitola Everything is an Object  
(<http://www.ibiblio.org/pub/docs/books/eckel/TIJ-3rd-edition4.0.zip>)
- Naučte se Javu – úvod (<http://interval.cz/clanky/naucte-se-javu-uvod/>)  
Naučte se Javu – dátové typy (<http://interval.cz/clanky/naucte-se-javu-datove-typy/>)

# 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*



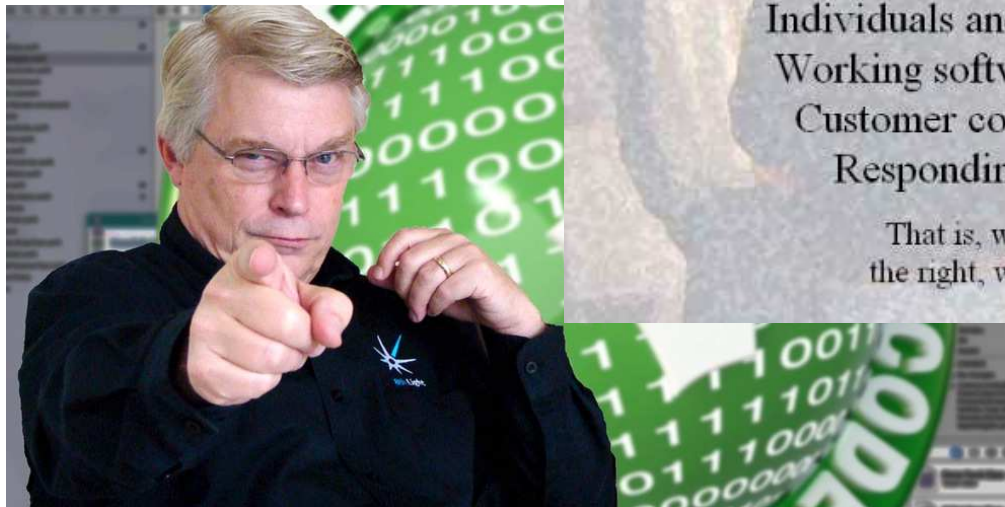
James Gosling  
Unix  
Emacs  
>15r.SUN  
Oracle  
Google

# OOP historia

(Uncle Bob Martin)



- <https://youtu.be/t86v3N4OshQ?t=496>



## 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.

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-8u121-windows-x64.exe](#)
  - edície: Standard Ed. (SE), Enterprise Ed. (EE), Micro Ed. (ME), ...
- Virtual Machine – interpreter byte kódu s knižnicami / Java Runtime Environment / java plugin do browsera
  - verzie: jre-8-OS, napr. [jre-8u121-windows-x64.exe](#)
- druhy programov, ktoré v prednáške :
  - spomenieme: aplikácie,
  - nespomenieme: applety, servlety, midlety, activity, ...
- prostredia pre vývoj - Java Integrated Environment
  - Eclipse (<http://www.eclipse.org/>)
  - IntelliJIdea (<http://www.jetbrains.com/idea/>)
  - NetBeans (<http://www.netbeans.org/>)
  - JBuilder (<http://www.embarcadero.com/products/jbuilder>)

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

V rámci prednášky/cvičení používame  
JDK 1.8 v prostredí Eclipse Protheo,  
resp. IntelliJ 2018.3



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
- sekvenčné a paralelné streamy
- Small Virtual Machine < 3MB

## **JDK 1.9 – túto budeme používať**

- Ahead-of-Time compilation (JIT)
- Jshell (read-eval-print-loop)
- drobnosti syntaxe

## **JDK 1.10 –**

## **JDK 1.11 – Long Term Support**







# Vývoj programátorských kultúr

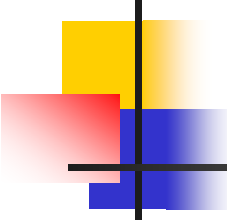
---

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.

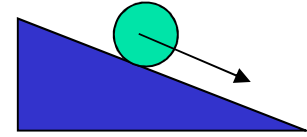
# Neštruktúrované programovanie

- 
- 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

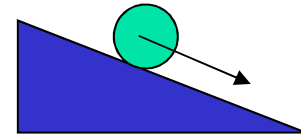


```
public class Gulicka1 {  
  
    public static void main(String[] args) {  
        double x=0.0, y=5.0, fi=0.56;  
        int t;  
        for (t=0; t<10; t++) {  
            x += Math.cos(fi);  
            y -= Math.sin(fi);  
        }  
    }  
}
```

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);  
    }  
}  
}
```

Súbor: [Gulicka2.java](#)



# Štruktúrované programovanie

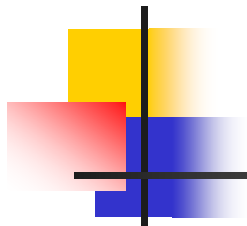
---

dáta:

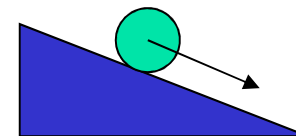
- entita = štruktúra
- štruktúra môže mať viac parametrov
- predstavuje dodefinovaný zložený typ
- štruktúra typu záznam (record/struct/class)
- varianty (case of, union) v jave nie sú
- skladanie štruktúr
- dynamika: statické aj dynamické štruktúry

riadenie:

- štruktúrované príkazy
- procedúry a funkcie s parametrami
- rekurzia



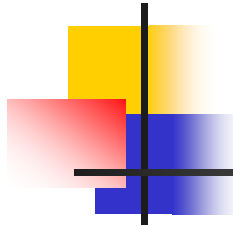
# Štruktúrované programovanie



```
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);  
    }  
}
```





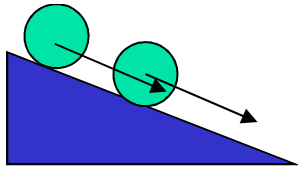
# OOP



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



# Objekt Gulička

```
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

- 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,...

Java 7

**Notácia s \_**

514\_000

0b1010 – binárne

0xFF\_FF

3.1415926535

\_8979323846

\_2643383279

\_5028841971

\_6939937510

\_5820974944

\_5923078164



# 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++  
nehľadajte #ifdef ... #endif



# Komentáre

---

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

    public static void main(String[] args) {
        double ucet;
        int pocetPiv = 5;
        ucet = pocetPiv * 1.3;                          // typický komentár
        System.out.println("Platis = " + ucet);

        ucet = pocetPiv * /* 1.3 */ 1.70; /* 1.3 je za desinku */
        System.out.println("Platis = " + ucet);

    }
}
```

Platis = 6.5  
Platis = 8.5

# 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	<a href="#">main</a> (java.lang.String[] args) toto je hlavny program
static double	<a href="#">posunX</a> (double x, double fi) definicia funkcie posunX
static double	<a href="#">posunY</a> (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://www.oracle.com/technetwork/articles/java/index-137868.html>
- Kde nájsť dokumentáciu k JDK SE 1.8

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 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

podmienенý výraz

// príklad: max = (i > j) ? i : j;

(booleovský výraz)?výraz1:výraz2



# Priradenie verzus porovnanie

---

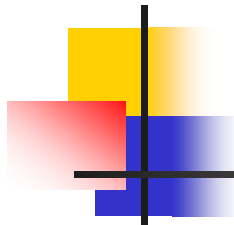
```
float f;                // definícia
f = 3.14;               // inicializácia/priradenie

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](#)

# 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](#)



# Operátory ++ a --

---

```
public class PlusPlus {
```

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

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

```
        i++;
```

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

i = 6

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

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

j = 7, i = 7

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

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

j = 7, i = 8

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

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

k = 8, j = 6  
// modulo

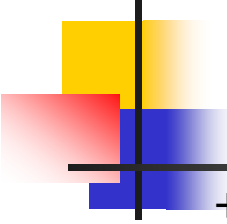
```
        i = j % 4;
```

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 <sup>n</sup> )	// (a+1) << 2
>> shift right (>> n je delenie 2 <sup>n</sup> )	// (a+1) >> 1
	// (a-1) >> 4
>>> unsigned right shift	// (a-1) >>> 4

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



# 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;           // 8-bitový vektor  
byte bit2 = 0x4;         // 416 = 0b1002  
stav |= bit2;            // nastav bit 2  
if ((stav & bit2) == bit2) ... // testuj bit 2  
stav &= ~bit2;           // zmaž bit 2
```

```
byte x = 5; x <<= 3;      // 4010 = (101)2 <<=3 = (101000)2  
int x = 256; x >>= 4;     // 1610 = (100000000)2 >>=4 = (10000)2  
int x = 16; x >>= 2;      // 410 = (10000)2 >>=2 = (100)2  
int x = -16; x >>= 2;     // 107374182010 = (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

Napíšte 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  
1 4 6 4 1



# Záver

---

Cieľom úvodnej prednášky s cvičeniami je aby ste vytvorili váš prvý program v jazyku JAVA, v prostredí Eclipse/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,
- vstup z konzoly s pomocnou barličkou (Input.java),
- komentáre –  
pomôžu nielen vám, ale aj cvičiacim pri hodnotení vašich kódov