

# JAVA

Úvod

# Úvodní informace

- Petr Hnětynka
  - [hnetynka@d3s.mff.cuni.cz](mailto:hnetynka@d3s.mff.cuni.cz)
- <https://d3s.mff.cuni.cz/teaching/nprg013/>
- **2/2 Zk/Z**
- zkouška
  - písemný test
- zápočet
  - zápočtový test
  - zápočtový program
    - "rozumná" velikost
    - **téma do 8. 1. 2021**
      - emailem cvičícímu
  - domácí úkoly – 225 bodů (z 450)
  - ~~docházka~~
    - ~~víc než 3 absence – 315 bodů~~

# Úvodní informace

- Virtuální cvičení pro repetenty
  - a ty co nechtějí chodit na cvičení
- Seznam "zakázaných" témat na zápočeták
  - piškvorky
  - lodě
  - tetris
  - ...
  - započťáky/příklady na algoritmy, grafiku
  - ...
- téma vždy dohodnout s cvičícím
- cvičení ve Po 17:20 je v angličtině

# Literatura, odkazy

- „Homepage“
  - <https://www.oracle.com/java/>
- Java tutorial (pozor – jen pro Java 8)
  - <https://docs.oracle.com/javase/tutorial/tutorialLearningPaths.html>
- Java Language Specification
  - <http://docs.oracle.com/javase/specs/>

# Java

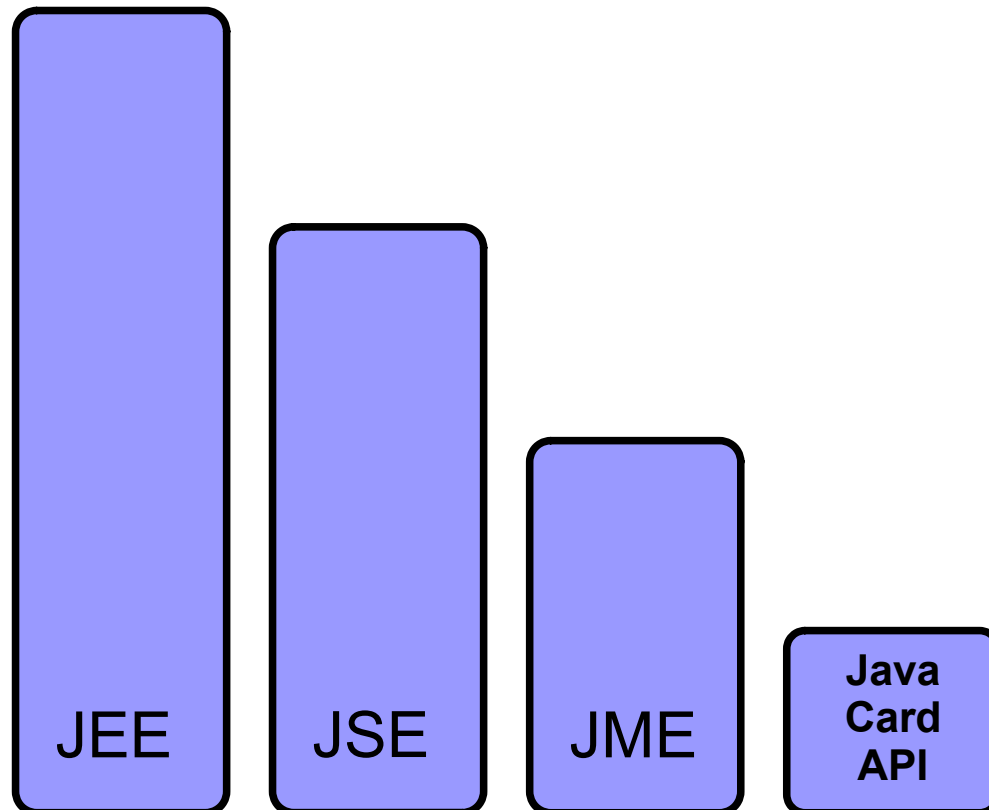
- objektově orientovaný
  - vše (téměř) je objekt
- interpretovaný
  - zdrojový kód (.java) – překlad do tzv. *bytecode*
  - bytecode (.class) – interpretován v tzv. *virtual machine*
    - just-in-time compilation
      - překlad bytecode do nativního kódu před/během vykonávání programu
- nezávislý na architektuře
  - programy běží ve *virtual machine*
- od Java 9
  - ahead-of-time compilation

# Historie

- 1.0 (1996)
- 1.1 (1997)
  - Vnitřní třídy
- Java 2 platform (2000)
  - 1.2, 1.3 – změny pouze v knihovnách
- 1.4 (2002)
  - Assert
- 5.0 (2004)
  - velké změny v jazyce – generické typy, anotace,...
- 6 (2006) – změny v knihovnách
- 7 (2011) – změny (malé) v jazyce
- 8 (2014) – velké změny v jazyce – lambda typy,...
- 9 (2017) – změny v jazyce – moduly
- 10 (2018) – změny v jazyce – odvození typu lok. prom (var)
- 11 (2018) – změny v knihovnách (redukce std knihovny)
  - long-term support
- 12 (2019) – rozšířený switch příkaz (ale pouze jako „preview“)
- 13 (2019) – další úprava switch, textové bloky (stále „preview“)
- 14 (2020) – switch (preview: plus records, textové bloky, instanceoff pattern matching)
- 15 (2020) – textové bloky (preview: records, instanceoff pattern matching, sealed classes)

# Java platform

- JSE – standard edition
- JEE – enterprise edition
- JME – micro edition



# Získání Javy

- <https://www.oracle.com/java/technologies/javase-downloads.html>
  - JDK
    - překladač, virtual machine, debugger, ...
      - Windows, Linux, Solaris
  - JRE
    - bez nástrojů pro vývoj (tj. bez překladače,...)
      - Windows, Linux, Solaris
  - dokumentace
- IDE
  - Netbeans – <http://www.netbeans.org/>
  - Eclipse – <http://www.eclipse.org/>
  - IntelliJ IDEA – <https://www.jetbrains.com/idea/>
- Ant – obdoba programu **make**
  - <http://ant.apache.org/>
- Maven – „like Ant on Steroids“
  - <http://maven.apache.org/>
- Gradle – podobné jako Maven

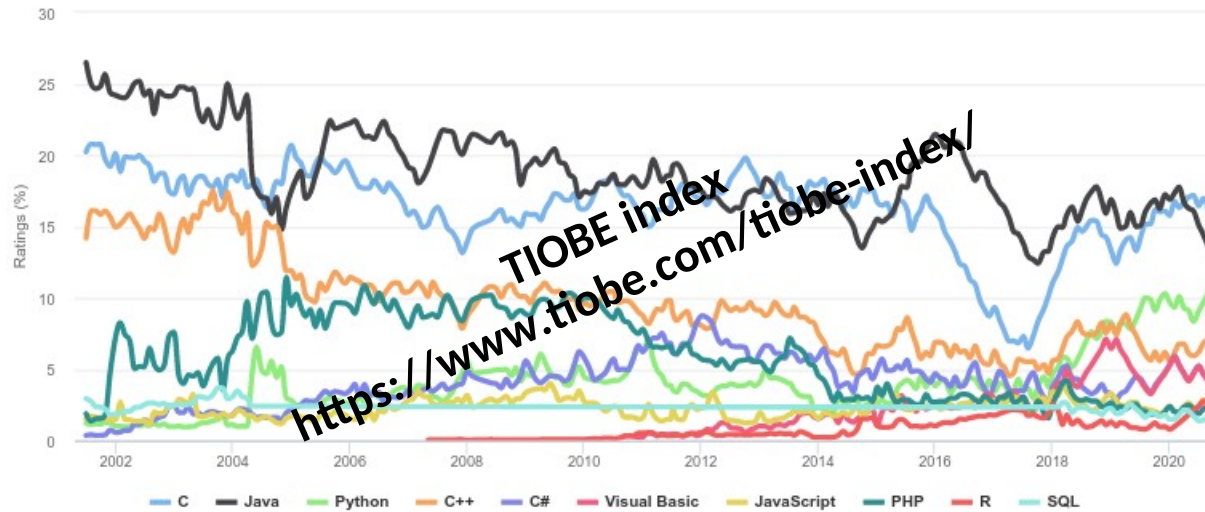


# Přibližný průběh přednášky

- Jazyk
  - třídy, primitivní typy, programové konstrukce,...
- Základní nástroje
- Základní třídy
  - vlákna, kolekce, vstup/výstup...

# Popularita

TIOBE Programming Community Index  
Source: www.tiobe.com



Rank	Language	Type	Score
1	Python	🌐 📄 📧	100.0
2	Java	🌐 📄 📧	95.3
3	C	📄 📧 📧	94.6
4	C++	📄 📧 📧	87.0
5	JavaScript	🌐	79.5
6	R	📄 📧 📧	78.6
7	Arduino	📄 📧 📧	77.2
8	Go	📄 📧 📧	73.1
9	Swift	📄 📧 📧	70.5
10	Matlab	📄 📧 📧	68.4

Worldwide, Sept 2020 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	31.56 %	+2.9 %
2		Java	16.4 %	-3.1 %
3		Javascript	8.38 %	+0.3 %
4		C#	6.5 %	-0.8 %
5		PHP	5.85 %	-0.5 %
6		C/C++	5.8 %	+0.0 %
7		R	4.08 %	+0.0 %
8		Objective-C	2.79 %	+0.2 %
9		Swift	2.45 %	-0.1 %
10		TypeScript	1.92 %	+0.1 %

<https://spectrum.ieee.org/at-work/tech-careers/top-programming-language-2020>  
 IEEE Spectrum  
 29.9.2020

Popularity Index  
<http://pypl.github.io/>

# JAVA

## Jazyk

# Komentáře

- Komentář

```
/* komentar */
```

```
// komentar do konce radku
```

- "dokumentační" komentáře (javadoc)

```
/** komentar */
```

# Objekty

- Vše je objekt
- Objekt – instance třídy (class) nebo pole
  - nová instance pomocí operátoru **new**
- Vše je definováno ve třídách
  - tj. žádné funkce mimo třídy (jako jsou třeba v C++)
- Manipulace s objekty – reference
  - Neexistují ukazatele

```
String s;
```

```
String s = new String("ahoj");
```

# Reference

```
StringBuilder s1 =  
    new StringBuilder("ahoj");  
StringBuilder s2 = s1;  
  
s1.append(" svete");  
  
System.out.println(s2);  
    // vytiskne "ahoj svete"
```

# Primitivní typy

- Výjimka – ne zcela vše je objekt
  - proměnné nejsou reference
  - pevná velikost, pouze se znaménkem

```
int a = 10;
```

Typ	Velikost	Min	Max	Wrapper
boolean	-	-	-	Boolean
char	16-bit	Unicode 0	Unicode $2^{16}-1$	Character
byte	8-bit	-128	+127	Byte
short	16-bit	$-2^{15}$	$+2^{15}-1$	Short
int	32-bit	$-2^{31}$	$+2^{31}-1$	Integer
long	64-bit	$-2^{63}$	$+2^{63}-1$	Long
float	32-bit	IEEE754	IEEE754	Float
double	64-bit	IEEE754	IEEE754	Double

# Primitivní typy – proměnné

```
int i1 = 42;
```

```
int i2 = i1;
```

```
i1 += 1;
```

```
System.out.println(i2);
```

```
// prints out 42
```



# Primitivní typy

- Vnitřní reprezentace celočíselných typů
  - „signed two's-complement integers“
  - př. typ **byte**
    - 0 ~ 00000000
    - 127 ~ 01111111
    - -1 ~ 11111111
    - -128 ~ 10000000
- Typy s pohyblivou řádovou čárkou
  - umožňují reprezentovat hodnotu NaN (not-a-number)
    - jakékoliv porovnání dvou NaN je vždy **false**

# Autoboxing, autounboxing

- od Java 5
- automatická konverze mezi primitivními typy a wrapper typy

```
int a = 5;  
Integer b = a;    // autoboxing  
int c = b;        // autounboxing
```

# Pole

- kontrola mezí
- definice polí  

```
int[] iArray;  
int i2Array[];
```
- vícerozměrná pole  

```
int[][] iiArray;
```
- vytvoření pole – pouze dynamicky  

```
iArray = new int [10];
```
- délka pole  

```
iArray.length
```

# Rušení objektů

- garbage collector

# Definice třídy

```
class MyClass {  
    /* telo tridy */  
}
```

- tělo třídy
  - atributy
  - metody
  - vnitřní/vnořné třídy

# Třída: Atributy

```
class MyClass {  
    int i;  
    float f;  
    boolean b;  
    String s;  
}  
  
...  
MyClass m = new MyClass();  
m.i = 5;  
m.f = 3.7;  
m.b = true;  
m.s = new String();
```

# Třída: Atributy

- Implicitní hodnoty
  - boolean – false
  - ostatní primitivní typy – 0
  - reference – null
- Pozor
  - lokální proměnné nejsou inicializovány
  - chyba při překladu

# Třída: Metody

```
navratovyTyp jmenoMetody ( parametry ) {  
    telo metody;  
}
```

```
class MyClass {  
    int pow2(int a) {  
        return a*a;  
    }  
  
    void nothing() {}  
}
```



# Třída: Metody

- volání metody

`objekt.jmenoMetody(parametry)`

```
MyClass m = new MyClass();  
int a = m.pow2(5);
```

- Předávání parametrů **hodnotou**

```
class Foo {  
    void plusOne(int a) {  
        a = a + 1;  
    }  
    void use() {  
        int a = 5;  
        plusOne(a);  
        System.out.println(a); // 5  
    }  
}
```

```
class Bar {  
    void appendA(StringBuilder sb) {  
        sb.append("A");  
    }  
    void use() {  
        StringBuilder sb =  
            new StringBuilder("A");  
        appendA(sb);  
        System.out.println(sb); // AA  
    }  
}
```

# enum

```
enum Planet {  
    MERCURY, VENUS, EARTH, MARS,  
    JUPITER, SATURN, URANUS, NEPTUNE,  
    PLUTO };
```

...

```
public Planet p1 = MARS;
```

# Balíky (packages)

- oddělené prostory viditelnosti jmen tříd
- balík (package)
  - množina tříd logicky patřících k sobě
  - obdoba v C#, C++ namespace
- každá třída patří do právě jednoho balíku
  - explicitně uvedený
  - implicitní neuvedený
- příslušnost k balíku  
`package jmenoBaliku;`

# Balíky

- hierarchická jména
  - "obrácená" internetová adresa tvůrce
  - `cz.cuni.mff.java.example01`
  - `org.w3c.dom`
- plný název třídy
  - `jmenoBaliku.JmenoTridy`
- třídy z vlastního balíku – "krátké" jméno
- třídy z jiného balíku – plné jméno
- zjednodušení používání – `import`

```
import jmenoBaliku.JmenoTridy;  
import jmenoBaliku.*;
```

- balík `java.lang` – vždy nainportován

# Klíčové slovo `static`

- `static` atributy a metody
  - nejsou svázány s konkrétní instancí (objektem)
  - někdy "*class data*", "*class methods*"

```
class MyClass {  
    static int i;  
}
```

```
class MyClass2 {  
    static void incr() {  
        MyClass.i++;  
    }  
}
```

# static import

- import statických položek
- používání bez jména třídy

```
import static java.lang.Math.PI;  
import static java.lang.Math.tan;  
...  
tan (PI/4) ;
```

# Viditelnost lokálních prom.

```
{  
    int x=10;  
    // dosazitelne je x  
    {  
        int y=11;  
        // dosazitelne je x i y  
    }  
    // dosazitelne je pouze x  
}
```

```
{  
    int x = 1;  
    {  
        int x = 2;    // chyba pri prekladu  
    }  
}}
```

# Třídy a soubory

- každá veřejná (`public`) třída – jeden soubor
- stejné jméno jako třída + `.java`
- balíky ~ adresáře

```
package jmenoBaliku;
```

```
import ....;
```

```
import ....;
```

```
public class JmenoTridy {
```

```
    ....
```

```
}
```

- neveřejné třídy (bez `public`)
  - viditelné pouze ze svého balíku



# Program

```
package cz.cuni.mff.java.example01;

public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

- uložit do
  - adresář ../cz/cuni/mff/java/example01
  - soubor Hello.java

# Program

- překlad
  - `javac Hello.java`
  - vznikne `Hello.class`
- spuštění
  - `java cz.cuni.mff.java.example01.Hello`
- nastavení **CLASSPATH**
  - seznam adresářů, kde se hledají třídy
    - proměnná prostředí `CLASSPATH`
    - parametry `-cp`, `-classpath`
  - např.
    - `/home/petr/java/cz/cuni/mff/java/example01/Hello.class`
    - `java -cp /home/petr/java cz.cuni.mff.java.example01.Hello`

# Spouštění „zdrojáků“

- od Java 11
- `java HelloWorld.java`

# Moduly – od Java 9

- modul
  - pojmenovaná kolekce tříd (a dalších elementů) (sada balíčků)
  - deklaruje
    - na kterých modulech závisí
    - které balíčky exportuje (tj. které balíčky jsou viditelné z jiných modulů)
  - mění se viditelnost (dosažitelnost) tříd
- module-info.java

```
module com.foo.bar {
    requires com.foo.baz;
    exports com.foo.bar.alpha;
    exports com.foo.bar.beta;
```

# Moduly – od Java 9

- MODULEPATH
  - obdoba CLASSPATH
- moduly lze „ignorovat“
  - bez určeného modulu => třída je v ***nepojmenovaném*** modulu
    - závisí na všech modulech
    - exportuje všechny balíčky
  - zejména kvůli zpětné kompatibilitě

# Operátory: přiřazení

- Přiřazení

```
int i;  
int[] array;
```

```
i = 4;  
array[4] = 5;  
4 = i; // spatne
```

- Primitivní typy

- kopírování hodnoty

- Objekty

- kopírování reference na objekt
- ne kopírování objektu!

# Operátory: aritmetika

- unární  
+ -
- binární  
+ - \* / %
- "zkratky" pro přiřazení  
+= -= \*= /= %=
- inkrementace a dekrementace  
- prefix i postfix  
i-- i++ --i ++i
- přetečení a podtečení jsou „tichá“  
- negeneruje se žádná výjimka

# Operátory: porovnání

- generují **boolean** výsledek

== != lze porovnat všechny typy

< > <= >= pouze primitivní mimo **boolean**

- test – co vypíše?

```
Integer i1 = new Integer(1);  
Integer i2 = new Integer(1);  
if (i1 == i2)  
    System.out.println("ANO");  
else  
    System.out.println("NE");
```

tyto konstruktory  
jsou deprecated



# Operátory: logické

- generují **boolean** výsledek
- lze použít jen na **boolean**

&&    ||    !

- zkrácené vyhodnocování

# Operátory: bitové

- lze použít na **short, int, long, char** a **boolean**

&      |      ^      ~

- zkratky

&=      |=      ^=

- není zkrácené vyhodnocování
- typ **boolean**
  - považován za 1-bit hodnotu
  - nelze na něj použít ~

# Operátory: posunutí

- lze použít na **short, int, long, char**
  - posun vlevo <<
    - doplňuje nuly do dolních bitů
  - posun vpravo >>
    - pokud je číslo kladné, doplňuje nuly
    - pokud je číslo záporné, doplňuje jedničky
  - posun vpravo >>>
    - vždy doplňuje nuly
- **char, byte, short**
  - vždy nejdřív převedeny na **int**
  - výsledek je **int**
- **long**
  - výsledek je **long**

# Operátory: různé

- Ternární operátor

```
int a;
```

```
a = a > 0 ? a : 0;
```

- Operátor **čárka**

- pouze v **for** cyklu

- Operátor **+** na **String**

- spojuje Stringy

- pokud je ve výrazu aspoň jeden String a jen operátory **+**, tak se vše ostatní ve výrazu převede na String a spojí

- Přetypování

```
int i = 1;
```

```
long x = (long) i;
```

- **Není sizeof operátor**

- není potřeba

# Operátory: priorita

Typ operátoru	Operátory
unární	+ - ++ --
aritmetika a posun	* / % + - << >>
porovnávání	> < >= <= == !=
logické a bitové	&&    &   ^
ternární	? :
přiřazení	= (zkratky typu +=)

- Při stejné prioritě se výrazy vyhodnocují zleva

# if - else

**if** (boolean-vyraz)

    prikaz

**else**

    prikaz

- **else** větev lze vynechat
- příkaz
  - jeden příkaz
  - složený příkaz { . . . . . }

# while, do - while

**while** (boolean-vyraz)  
prikaz

**do**  
prikaz

**while** (boolean-vyraz) ;

- provádějí se, dokud podmínka platí

# for

```
for (inicializace; boolean-vyraz; krok)  
    prikaz
```

- v inicializace a v krok lze použít operátor **čárka**

```
for (int i=1,j=1; i<5; i++, j=i*10) {  
    ....  
}
```



# for

```
int[] arr = new int [10];  
  
for (int i:arr) {  
    ...  
}
```

- pole
- objekty s *iterátorem*

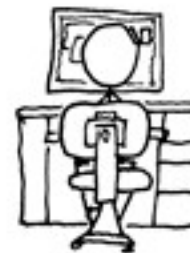
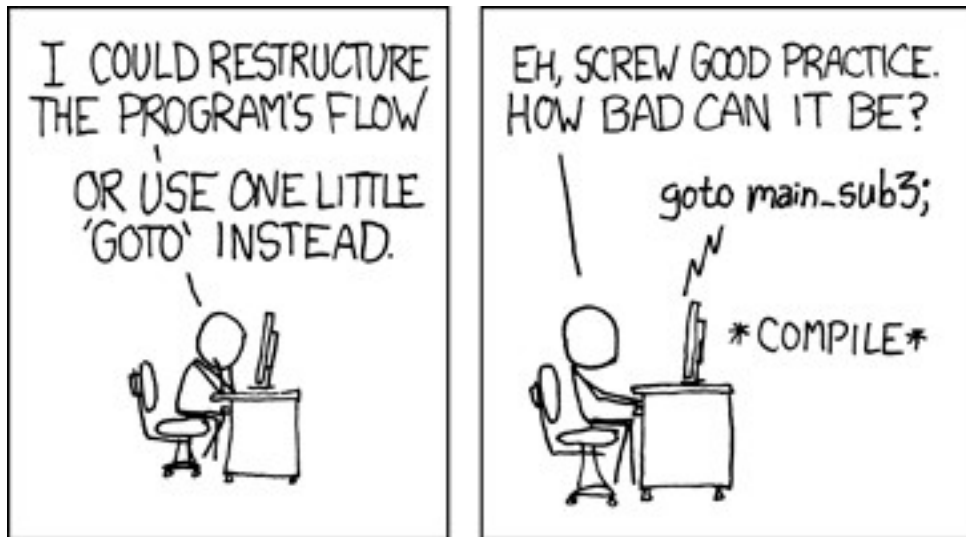
# break, continue

- **break**
  - ukončí provádění cyklu
- **continue**
  - ukončí aktuální provádění cyklu a začne novou smyčku
- návěští (labels) – význam pouze před cyklem

```
label: vnejsi-cyklus {  
    vnitri-cyklus {  
        break;  
        continue;  
        continue label;  
        break label;  
    }  
}
```

# goto

- **goto**
  - je rezervované slovo
  - není použito!



<http://xkcd.com/292/>

# switch

```
int a;  
...  
switch (a) {  
    case 1:  
    case 2: System.out.println("1, 2");  
        break;  
    case 3: System.out.println("3");  
        break;  
    default: System.out.println("3..");  
}
```

- od Java 7 lze v **switch** použít i typ **String**
- od Java 14 rozšíření **switch** příkazu
  - bude později



Verze prezentace J01.cz.2020.01

Tato prezentace podléhá licenci [Creative Commons Uved'te autora-Neužívejte komerčně 4.0 Mezinárodní License](https://creativecommons.org/licenses/by-nc/4.0/).