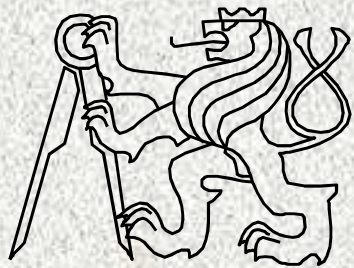


Java – základy



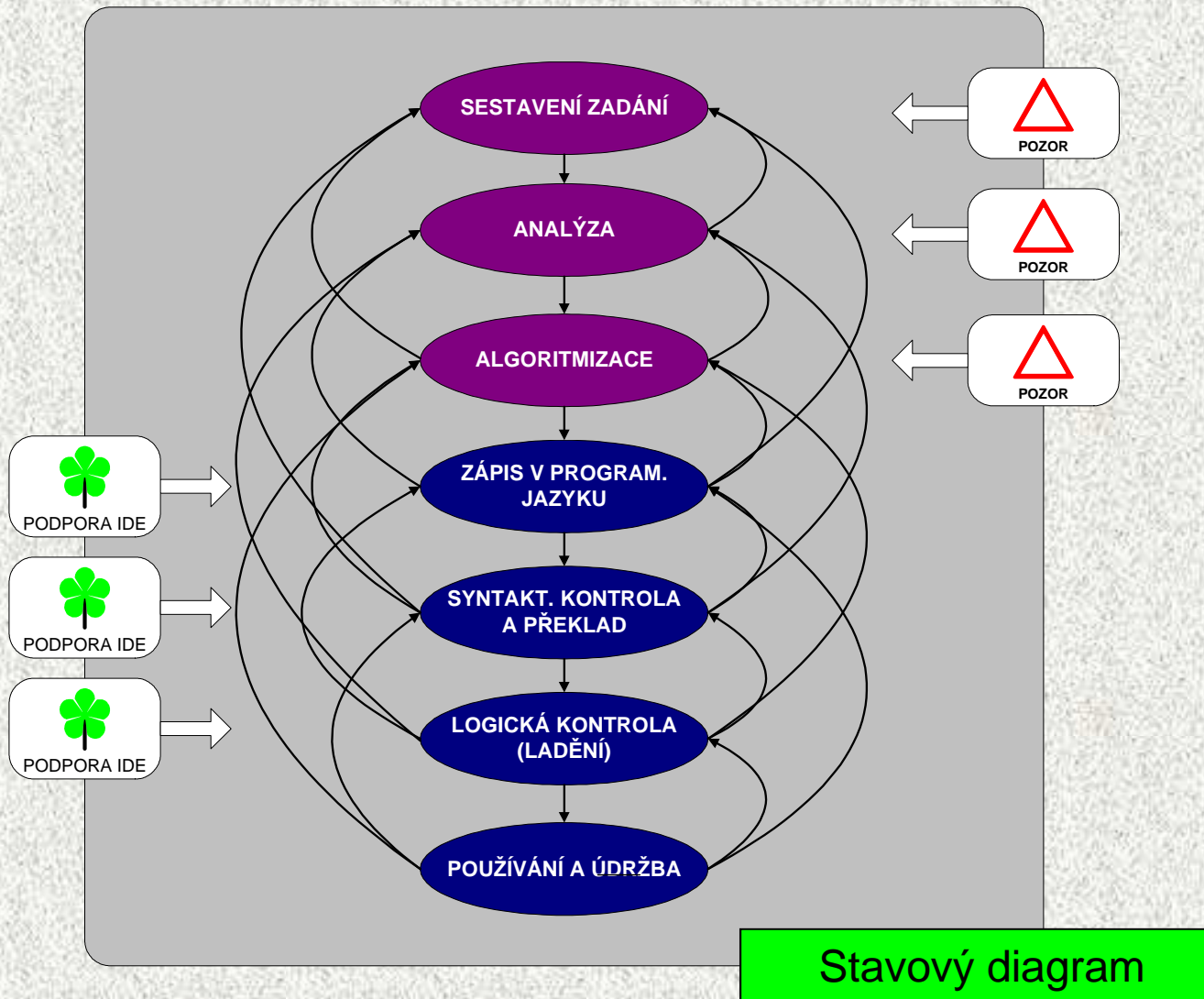
A0B36PR1-Programování 1
Fakulta elektrotechnická
České vysoké učení technické

Dva základní přístupy k imperativnímu programování

- Strukturované – procedurální
- Objektové
- V PR1 + PR2 byl zvolen postup od strukturovaného k objektovému
 - Těžiště PR1 je ve strukturovaném přístupu, PR2 v objektovém
 - Jazyk C nemá objektové možnosti
 - Princip strukturovaného přístupu:

Program = Data + Algoritmus

Řešení problému pomocí počítače



Reprezentace dat v počítači

- V počítači není u každé datové položky určeno, jaký typ dat uchovává
- V jazyce Java musíme deklarovat s jakými údaji (typy dat) budeme pracovat!!
- Překladač jazyka Java tuto deklaraci hlídá!!

Příklad:

$(0100\ 0001)_2 = (41)_{16}$ nebo $(65)_{10}$ nebo znak A

Reprezentace dat v počítači

- **bit** (**b**inary **d**igit – dvojková číslice; angl. bit – drobek, kousek)
 - základní a nejmenší jednotky informace
 - nabývá hodnot 0 nebo 1
- **byte** (též bajt, česky - slabika) – uspořádaná osmice bitů

0 1 1 0 0 1 1 1

- Násobky
- 1 kB = 1 000 bajtů (dekadický kilobajt, malé „k“)
- 1 KB = 1 024 bajtů = 2^{10} (binární kilobajt, velké „K“)
- 1 MB = 1 048 576 bajtů = 2^{20}
- 1 GB = 1 073 741 824 bajtů = 2^{30}

Reprezentace celých čísel

- číselné soustavy - polyadické

$$X_d = \sum_{i=-n}^{i=m} a_i z^i, \quad a - \text{čísllice soustavy, } z - \text{základ soustavy}$$

- Desítková soustava

$$\begin{aligned} 138,24 &= 1 \cdot 10^2 + 3 \cdot 10^1 + 8 \cdot 10^0 + 2 \cdot 10^{-1} + 4 \cdot 10^{-2} \\ &= 1 \cdot 100 + 3 \cdot 10 + 8 \cdot 1 + 2 \cdot 0,1 + 4 \cdot 0,01 \end{aligned}$$

- Dvojková soustava

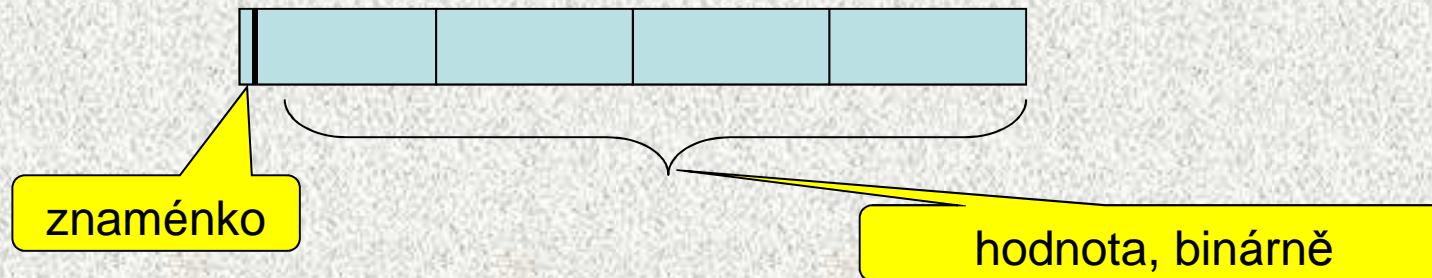
$$\begin{aligned} 11010,01_b &= 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = \\ &= 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 + 0 \cdot 1/2 + 1 \cdot 1/4 = 26,25 \end{aligned}$$

- Šestnáctková soustava (hexadecimal numeral system)

- $a = „0“, „1“, „2“, \dots „9“, „A“, „B“, \dots „F“; z = 16$
- Př: $07D_h = 0 \cdot 16^2 + 7 \cdot 16^1 + D \cdot 16^0 = 0 + 112 + 13 = 125$

Celá čísla v Javě – typ int

int je reprezentováno 32 bity



největší číslo $0111\dots111 = 2^{31} - 1 = 2147483647$

Pro zobrazování záporných čísel – doplňkový kód

nejmenší číslo v doplňkovém kódu $1000\dots000 = -2^{31} = -2147483648$

Reprezentace záporných celých čísel

- Doplnkový kód – $D(x)$

- předpokládejme reprezentaci čísel pomocí 8 bitů, lze reprezentovat $2^8 = 256$ čísel = r (rozsah)

$$D(x) = \begin{cases} x, & \text{pro } 0 \leq x < r/2 \\ r+x, & \text{pro } 0 > x \geq -r/2 \end{cases}$$

*Pozor na důsledky počítání
(v doplnkovém kódu) pro `int`:*
 $2\ 000\ 000\ 000 + 1\ 000\ 000\ 000 =$
 $-1\ 294\ 967\ 296$

desítkově

doplnkový kód

0 – 127

00000000 – 01111111

128

nelze zobrazit na 8 bitů v d.k.

-128

$D(-128) = -128 + 256 = 128 = 10000000$

-1

$D(-1) = -1 + 256 = 255 = 11111111$

-4

$D(-4) = -4 + 256 = 252 = 11111100$

Nepřesnost v zobrazení čísel

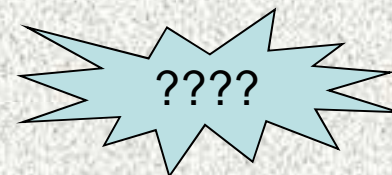
Nepřesnosti způsobují

1. Iracionální čísla (e , π , $\sqrt{2}$,.....)
2. Čísla, jež mají v dané soustavě periodický rozvoj (1/3 ve dvojkové či dekadické soustavě, 1/10 ve dvojkové)
3. Čísla, která mají příliš dlouhý zápis

Implementace

`double` (na 64 bity)

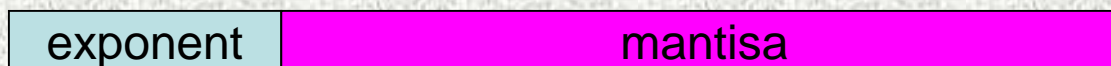
- 1bit znaménko – dvě možnosti, +,-
- 11 bitů exponent – 2048 možností
- 52 bitů – 4 503 599 627 370 496 možností, tedy asi 4,5 biliardy
 - není možné v typu `double` přesně uložit čísla se zápisem delším než 52 bitů!
 - čím větší exponent tím větší „mezery“ mezi sousedními aproximacemi!!!



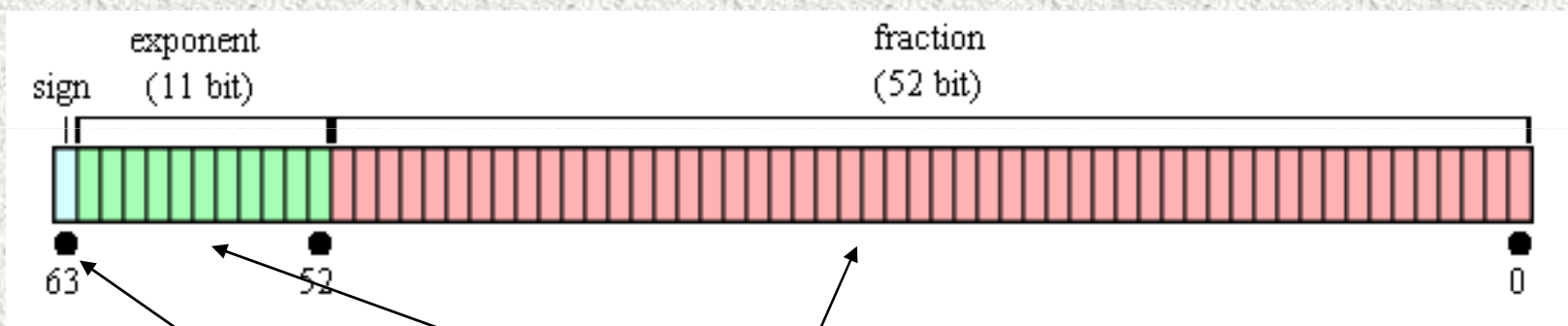
Necelá čísla v Javě – typ double

Reálná čísla X se zobrazují ve tvaru:

$$X = \text{mantisa} * \text{základ}^{\text{exponent}}$$



double je reprezentován 64 bity, norma IEEE 754



znaménkový bit (s), exponent, mantisa

Nejmenší čísla v normalizovaném tvaru

$$\pm 2^{-1022} \approx \pm 2.2250738585072020 \times 10^{-308}$$

Největší čísla

$$\pm (1 - (1/2)^{53}) 2^{1024} \approx \pm 1.7976931348623157 \times 10^{308}$$

Model reprezentace reálných čísel

Reálná čísla se zobrazují jako aproximace daných rozsahem paměťového místa

Reálná čísla se zobrazují ve tvaru:

$$X = \text{mantisa} * \text{základ}^{\text{exponent}} = m * z^{\text{exponent}}$$

Mantisa musí být normalizována:

$$0,1 \leq m < 1, \text{ důvod: jednoznačnost zobrazení}$$

Model:

- délka mantisy 3 pozice + znaménko
- délka exponentu 2 pozice + znaménko

Příklad

$$X = 77.5 = +0.775 * z^{+02}$$

+|775|+|02

1|775|1|02

- zakódujeme znaménko “+” = 0, “-” = 1
- z je pro názornost 10

Model reprezentace reálných čísel

Maximální zobrazitelné kladné číslo

Minimální zobrazitelné kladné číslo

Maximální zobrazitelné záporné číslo (v absolutní hodnotě)

Minimální zobrazitelné záporné číslo (v absolutní hodnotě)

Nula

+|999|+|99

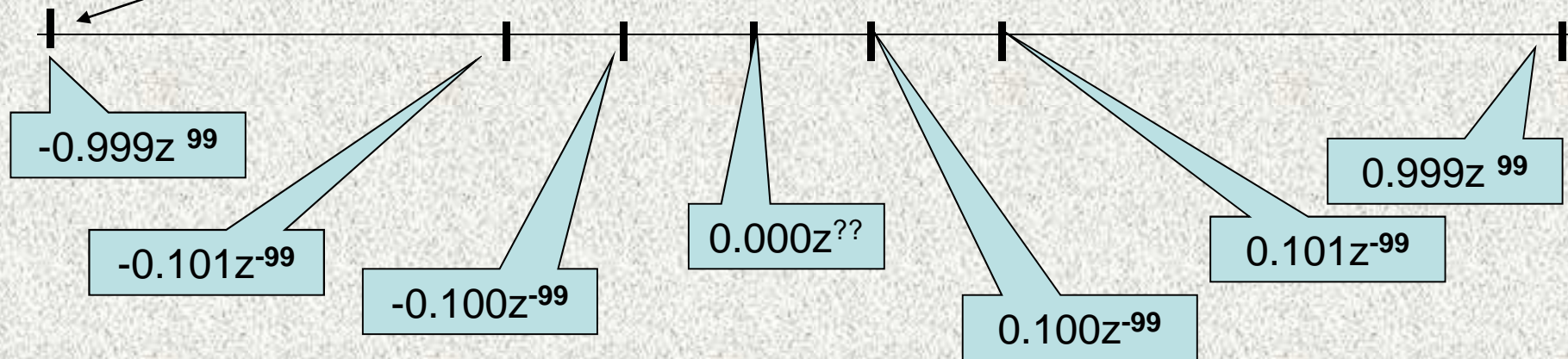
+|100|-|99

-|999|+|99

-|100|-|99

0|000|???

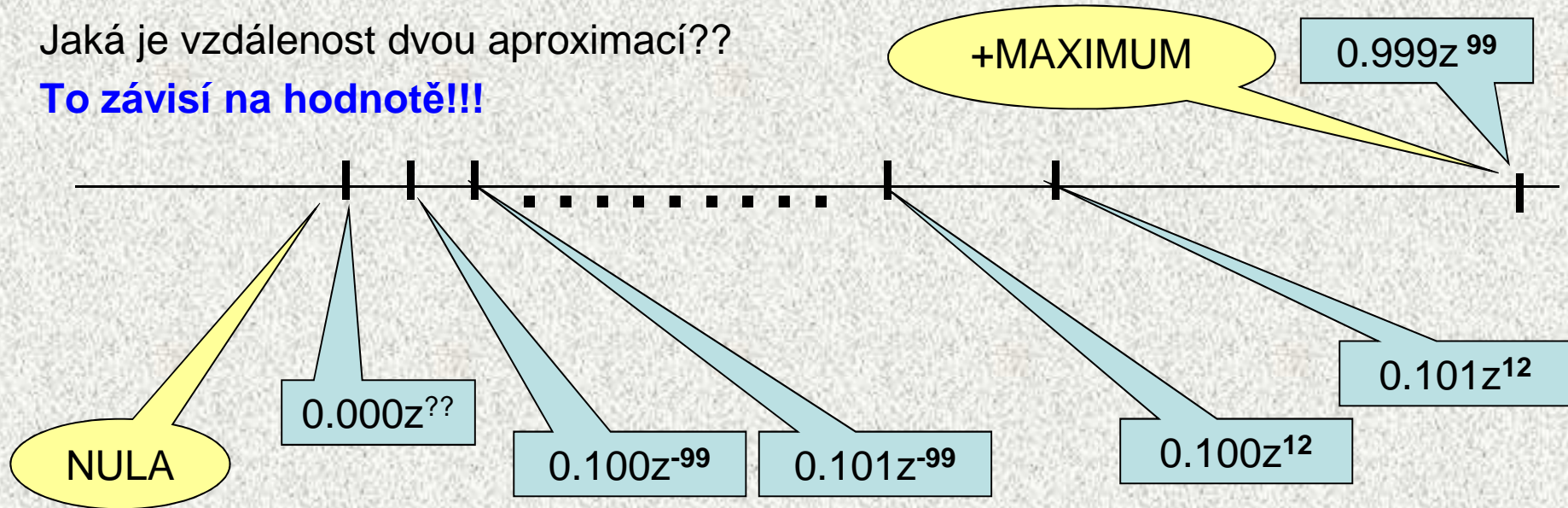
Jak je to se zobrazitelnými hodnotami „okolo nuly“,



Model reprezentace reálných čísel

Jaká je vzdálenost dvou aproximací??

To závisí na hodnotě!!!

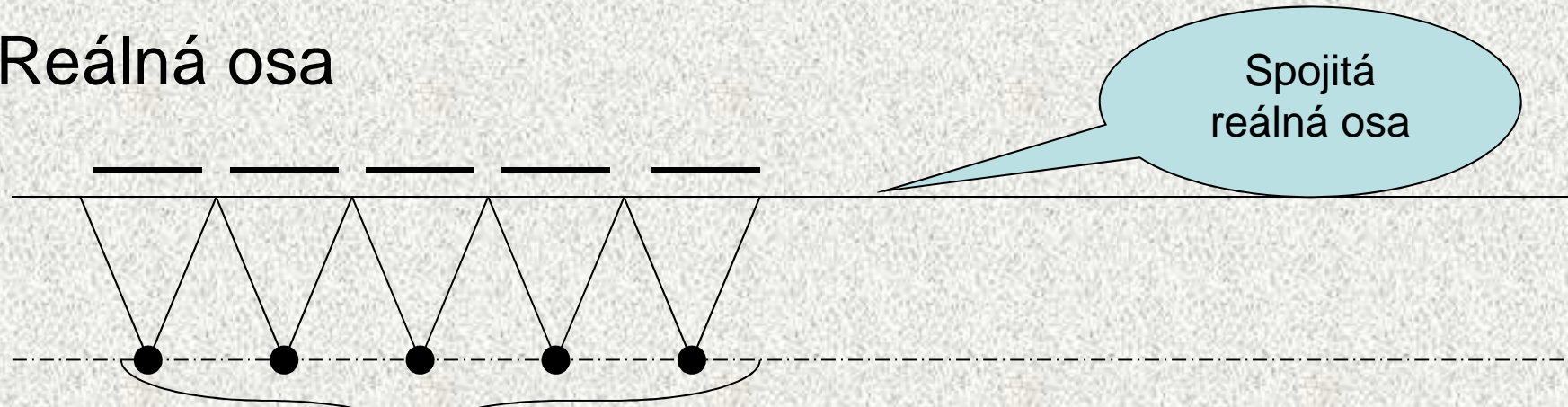


Aproximace reálných čísel: nejsou na číselné ose rovnoměrně rozložené!!



Reprezentace reálných čísel

Reálná osa



Zobrazení aproximací (pro jeden exponent!!)

Reálná čísla se zobrazují jako aproximace daných rozsahem paměťového místa

Diskrétní
aproximace v
paměti počítače

Reprezentace znaků

Java používá typ `char`, znaky jsou kódovány 16 bitovým kódem Unicode (UTF-16, podmnožina UCS-2).

- Odstraňuje problémy češtiny
- Programy v Javě provádí konverzi znaků mezi vnitřní reprezentaci v programech a OS, většinou automaticky podle informací OS (Locale)
- Poznámky
 - Různé OS různá kódování
 - sedmibitové ASCII (American Standard Code for Information Interchange), osmibitová Win1250, ISO 8859-2, kód s proměnnou délkou znaku - UTF8(UCS Transformation Format), .
 - Java – UCS2
 - Universal Character Set - univerzální znaková sada definována normou ISO 10646

Reprezentace znaků, část reprezentace

Dec	Hex	Znak	Význam	Dec	Hex	Znak	Dec	Hex	Znak	Dec	Hex	Znak
0	00	NUL		32	20	SPC	64	40	@	96	60	`
1	01	␣	Start of Header (SOH)	33	21	!	65	41	A	97	61	a
2	02	␣	Start of Text (STX)	34	22	"	66	42	B	98	62	b
3	03	␣	End of Text (ETX)	35	23	#	67	43	C	99	63	c
4	04	␣	End of Transmission (EOT)	36	24	\$	68	44	D	100	64	d
5	05	␣	Enquiry (ENQ)	37	25	%	69	45	E	101	65	e
6	06	␣	Acknowledge (ACK)	38	26	&	70	46	F	102	66	f
7	07	␣	Bell (BEL)	39	27	'	71	47	G	103	67	g
8	08	␣	Backspace (BS)	40	28	(72	48	H	104	68	h
9	09	␣	Horizontal Tab (HT)	41	29)	73	49	I	105	69	i
10	0a	␣	Line Feed (LF)	42	2a	*	74	4a	J	106	6a	j
11	0b	VT	Vertical Tab	43	2b	+	75	4b	K	107	6b	k
12	0c	FF	Form Feed	44	2c	,	76	4c	L	108	6c	l
13	0d	CR	Carriage Return	45	2d	-	77	4d	M	109	6d	m
14	0e	SO	Shift Out	46	2e	.	78	4e	N	110	6e	n

Primitivní či základní datové typy

Typ	Bitů	Rozsah	Obal.třída
Celočíselný typ			
byte	8	-128 ... 127	Byte
short	16	-32768 ... 32767	Short
int	32	-2147483648 ... 2147483647	Integer
long	64	-9223372036854775808 ... 9223372036854775807	Long
Reálný typ, IEEE 754 (NaN, infinity)			
float	32	$2^{-149} \dots (2-2^{-23}) \cdot 2^{127}$	Float
double	64	$2^{-1074} \dots (2-2^{-52}) \cdot 2^{1023}$	Double
Znaky, UCS2			
char	16	'\u0000' to '\uffff' 0 ... 65535	Character
Logický typ			
boolean	1/8	true false	Boolean
Pomocný prázdný typ			
void			

Výrazy

- Výraz předepisuje výpočet hodnoty určitého typu
- Příklad výrazu: $\{14.6 + \sin(3.14)\} * a_{100}$
- Výraz může obsahovat:
 - proměnné
 - konstanty
 - binární operátory
 - volání funkcí
 - unární operátory
 - závorky
- Pořadí operací předepsaných výrazem je dáno:
 - prioritou operátorů
 - asociativitou operátorů
- Příklad:

výraz

$x + y * z$

$x + y + z$

pořadí operací

$x + (y * z)$

$x + y) + z$

zdůvodnění

* má vyšší prioritu než

+ je asociativní zleva

Vyhodnocení výrazů, příklady

$$124+4*8 = 124 + (4*8) = 124 + 32 = 156$$

// násobení má vyšší prioritu

$$36/2*9 = (36/2)*9 = 18*9 = 162$$

// násobení i dělení mají stejnou prioritu, asociativita je L - zleva

Asociativita a priorita operací

- Binární operace na množině S **asociativní**, jestliže platí
 - $(x * y) * z = x * (y * z)$, pro každé x, y a z v S .
- U neasociativních operací je tedy třeba buď důsledně závorkovat, nebo se dohodnout na implicitním pořadí provádění operací – pak se někdy mluví o operacích *asociativních zleva* či *asociativních zprava*.
- **Priorita binárních operací** vyjadřuje pořadí, v jakém se provádějí binární operace
- Příklady:
 - Odčítání levě asociativní,
 - výraz $10 - 5 - 3$ se chápe jako $(10 - 5) - 3$,
 - Umocňování je asociativní zprava

$$2^{3^4} = 2^{(3^4)}$$

Aritmetické operátory

- Pro operandy typu `int` a `double` budeme používat tyto aritmetické operátory (seřazeno sestupně podle priority):
 - unární `-` (změna znaménka)
 - binární `*`, `/`, `%` (násobení, dělení a zbytek po dělení)
 - binární `+` a `-` (sčítání a odčítání)
- Jsou-li oba operandy stejného typu, výsledek aritmetické operace je téhož typu
- Jsou-li operandy různého typu, operand typu `int` se implicitní konverzí převede na hodnotu typu `double` a výsledkem operace je hodnota typu `double`
- Výsledkem dělení operandů typu `int` je celá část podílu
např. $7/3$ je 2 $-7/3$ je -2
-
- Pro zbytek po dělení platí: $x \% y = x - (x / y) * y$
např: $7\%3$ je 1; $-7\%3$ je -1; $7\%-3$ je 1; $-7\%-3$ je -1
- speciální inkrementační a dekrementační operátory:
`++x`, `x++`, `--x`, `x--`

Relační operátory

- Hodnoty všech jednoduchých typů jsou uspořádané a lze je porovnávat relačními operátory
- Budeme používat tyto relační operátory (priorita je menší než priorita aritmetických operátorů):
 - `>`, `<`, `>=`, `<=` (větší než, menší než, větší nebo rovno, menší nebo rovno)
 - `==`, `!=` (rovná se, nerovná se)
- Výsledek relační operace je typ `boolean` (`true`, když relace označená operátorem platí, `false` v opačném případě)
- Jestliže při porovnávání číselných hodnot jsou operandy různého typu, operand typu `int` se implicitní konverzí převede na hodnotu typu `double`
- Relační operátory mají menší prioritu, než aritmetické operátory
- Příklady relačních výrazů (relací):

```
int i=10; double x=12.3; boolean b;  
System.out.println(i==10); // vypíše true  
System.out.println(i+1==10); // vypíše false  
b = i>x; // proměnné b se přiřadí false
```

Logické operátory

- Logické operátory jsou definovány pro hodnoty typu `boolean`
 - unární `!` (negace)
 - binární `&&` resp. `&` (konjunkce, logický součin)
 - binární `||` resp. `|` (disjunkce, nevýhradní logický součet, OR)
 - Binární `^` (disjunkce, výhradní logický součet, XOR)

x	y	!x	x && y	x y	x ^ y
false	false	true	false	false	false
false	true	true	false	true	true
true	false	false	false	true	true
true	true	false	true	true	false

- Negace má stejnou prioritu, jako změna znaménka, logický součin má nižší prioritu než relační operátory
- Operace `&&` a `||` se vyhodnocují zkráceným způsobem, tj. druhý operand se nevyhodnocuje, jestliže lze výsledek určit již z prvního operandu
- Příklady:

```
int n = 10; boolean b1 = false, b2 = true;
System.out.println(1<=n && n<=20); // vypíše se true
System.out.println(b1 || !b2); // vypíše se false
if (y != 0 && x/y < z) // zkrácené vyhodnocení
```


Operátory a jejich priorita

priorita	operátor	typ operandu	asociativita	operace
1	++	aritmetický	P	pre/post inkrementace
	--	aritmetický	P	pre/post dekrementace
	-	aritmetický	P	unární plus/minus
	~	celočíselný	P	bitová inverze
	!	logický	P	logická negace
	(typ)	libovolný	P	přetypování
2	*, /, %	aritmetický	L	násobení, dělení, zbytek
3	-	aritmetický	L	odečítání
	+	aritmetický, řetězový	L	sčítání, zřetězení

Operátory a jejich priorita II

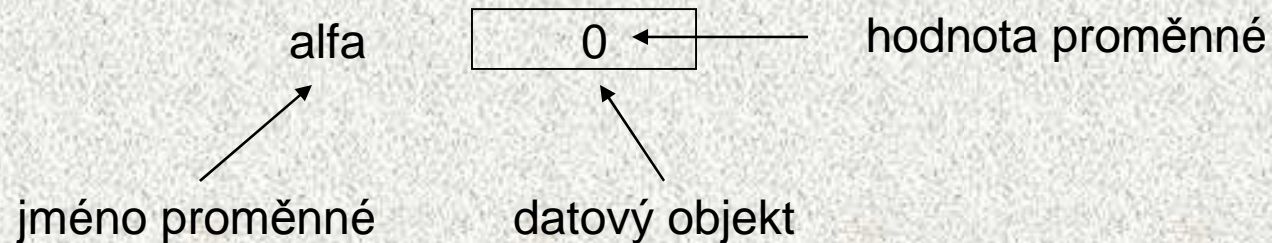
priorita	operátor	typ operandu	asociativita	operace
4	<<	celočíselný	L	posun vlevo
	>>	celočíselný	L	posun vpravo
	>>>	celočíselný	L	posun vpravo s doplňováním nuly
	<,>, <=, >=	aritmetický	L	porovnání
	instanceof	objekt	L	test třídy
	==, !=	primitivní	L	rovno, nerovno
7	&	celočíselný nebo logický	L	bitové nebo logické AND
8	^	-"-	L	bitové nebo logické XOR
9		-"-	L	bitové nebo logické OR

Operátory a jejich priorita III

priorita	operátor	typ operandu	asociativita	operace
10	&&	logický	L	logické AND vyhodnocované zkráceně
11		logický	L	logické OR vyhodnocované zkráceně
12	? :	logický	P	podmíněný operátor
13	=, -=, *=, /=, % =, ==, & =, ^ =, =	libovolný	P	přiřazení

Proměnné a přiřazení

- Proměnná je datový objekt, který je označen jménem a je v něm uložena hodnota nějakého typu, která se může měnit



- V jazyku Java zavedeme výše uvedenou proměnnou deklarácí
`int alfa = 0;`



- Hodnotu proměnné lze změnit přiřazovacím příkazem

```
alfa = 37;
```

Deklarace proměnných

- každá proměnná má definovaný **typ**
- Java zná 8 primitivních datových typů
 - ostatní typy jsou referenční – objekty, pole, řetězce,..)

Příklad

`int a;`

- definuje proměnnou typu int
- lze do ní přiřadit pouze hodnoty typu int a hodnoty užší (tedy hodnoty, které se dají rozšířit na int, tedy byte, short a char)
- lze s ní provádět operace definované pro int

Jak spolu souvisí typ int a typ „celá čísla“??

Deklarace proměnných

Proměnné se zavádějí deklaracemi

```
int i;           // deklarace proměnné i typu int
double x;       // deklarace proměnné x typu double
```

- Proměnná deklarovaná uvnitř funkce (lokální proměnná) nemá deklarací definovanou hodnotu
- Použití proměnné s nedefinovanou hodnotou v jazyku Java je chyba při překladu

```
int x, y;
x = y + 2; // chyba při překladu, není známa hodnota y
```

- Deklaraci proměnné lze doplnit o inicializaci proměnné:

```
int x = 10; // deklarovaná proměnná má hodnotu 10
```

- Deklarací lze zavést několik proměnných stejného typu:

```
int x, z;
```

Přiřazovací příkaz

```
int a, b = 10;
```

- hodnota a není definována, hodnota b je 10

```
a = b;
```

- a je 10, b je 10

```
b = 12 + 5;
```

- a je 10, b je 17

```
a = a+3;
```

- vezmi hodnotu z a (10), zvětši ji o 3 a výsledek ulož zpět do a, a je 13
- Pozor, proč lze psát:

```
y = x = x + 6;
```

- Přiřazovací příkaz je výrazem - vyhodnotí se jako

```
y = (x = (x + 6));
```

Proč jazyk Java?

- jde o vyšší, obecně použitelný programovací jazyk s **vysokým stupněm zabezpečení**
- je **objektově orientovaný**, umožňuje však i klasické **procedurální programování**
- vytvořené programy jsou **zcela portabilní** (program vytvořený pod MS Windows bez problémů funguje pod Unixem a naopak)
- syntaxe výrazů a příkazů **vychází z jazyka C**; přechod z Javy na C nebo C++ je tedy jednodušší, než odjinud
- základní implementaci (JDK – Java Development Kit) firmy Sun lze pro prostředí Windows i Unix stáhnout ze stránek firmy Sun:
<http://java.sun.com>
- My používáme vývojové prostředí NetBeans 6.8 resp 6.9 resp. 7.0 resp. 7.1.2 ☺, fy. Sun Microsystem. <http://www.netbeans.org/>

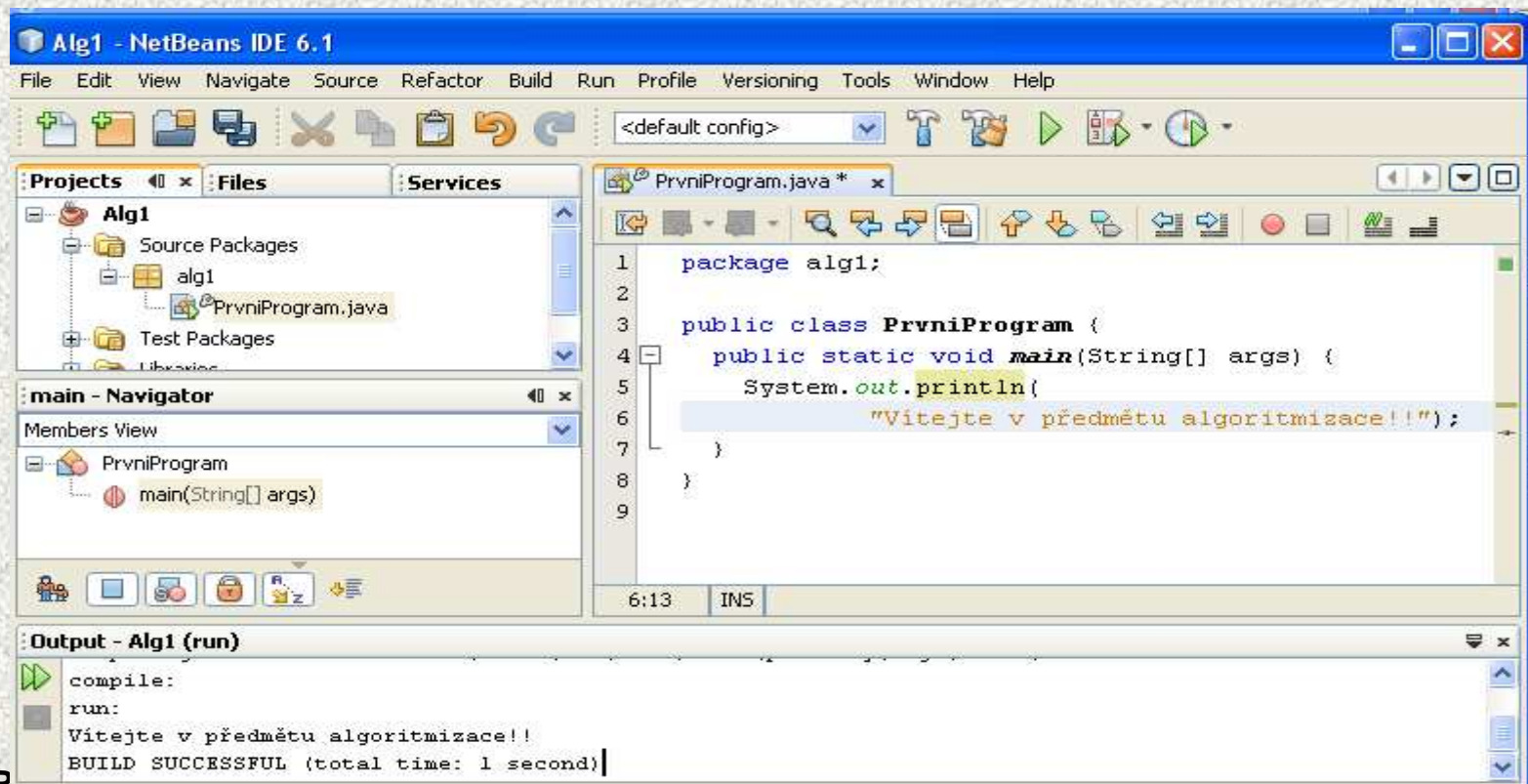
IDE - vývojová prostředí

IDE – integrované vývojové prostředí

- NetBeans – opensource -zdarma, původně Xelfi, vyvíjeno v Praze firmou SUN Microsystems
- Eclipse - opensource - zdarma, fa IBM
- IDEA - komerční (30tí denní zkušební verze zdarma)
- JBuilder - základní verze zdarma pro nekomerční využití, fa Borland
- JDeveloper - vývojové prostředí firmy Oracle, freeware
- BlueJ – bylo volně šiřitelné multiplatformní vývojové prostředí

Vývojový systém

- Programy v jazyku Java budeme vytvářet pomocí **vývojového prostředí NetBeans**, který přípravu programu, jeho překlad a provedení zjednodušuje
- Se systémem NetBeans se seznámíte na 2. cvičení v počítačové učebně



Úvod do jazyka Java

Jazyk Java je implementován interpretačním způsobem

- program je tvořen jedním nebo několika **zdrojovými soubory** s příponou **.java**:

`Program.java`

- zdrojové soubory se přeloží **překladačem(*) javac** do **vnitřní formy** (byte code, bajt-kód) s příponou **.class**:

`Program.java > javac > Program.class`

- **interpretaci** vnitřní formy **provede program java** (JVM – Java Virtual Machine v balíčku JRE Java Runtime Environment) a provede výpočet:

`Program.class > java > „výpočet“`

(*) v terminologii firmy Sun to je kompilátor

Poznámka:

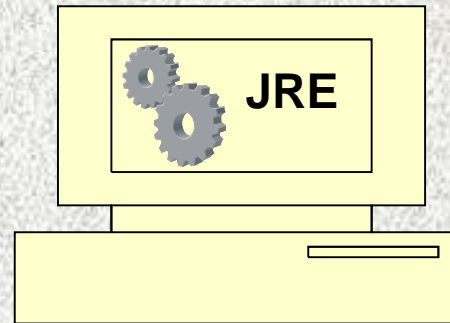
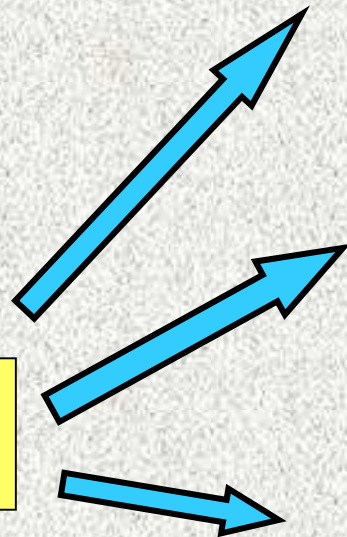
- program obvykle **využívá řadu knihoven**, které je třeba mít k dispozici jak při překladu, tak při interpretaci!!!

Interpretační metoda - jazyk Java

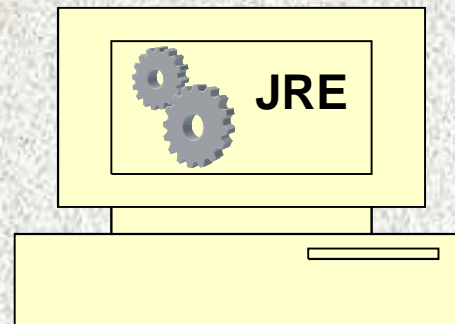
Zdrojový kód
v jazyku Java
soubor *.java*



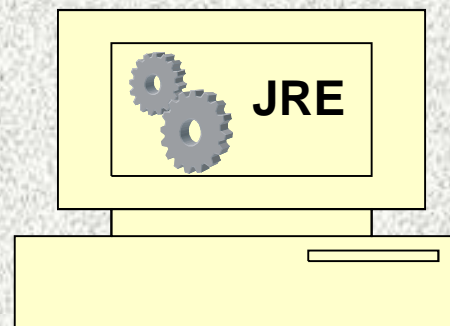
Bytecode
soubor *.class*



OS MS
Windows



OS GNU
Linux



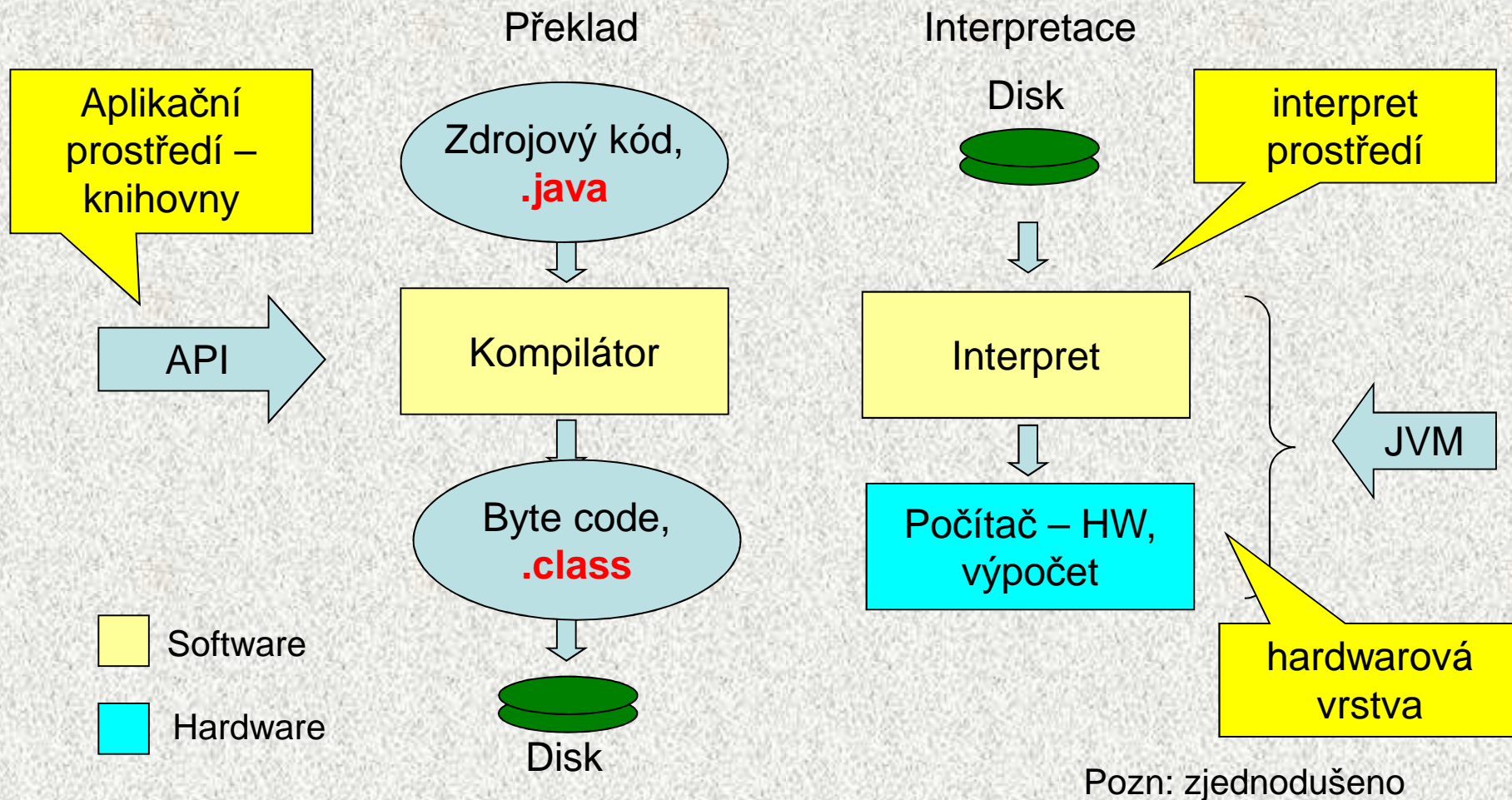
OS
Solaris

První program v jazyku Java

- Příklad programu, který vypíše daný text na obrazovku:

```
public class PrvniProgram {  
    public static void main(String[] args) {  
        System.out.println("Nazdar, toto je prvni program");  
    }  
}
```
- Po překladu a spuštění se na obrazovku vypíše
 - Nazdar, toto je prvni program
- Nejjednodušší zdrojový program - jeden soubor
 - deklarace veřejné třídy (`public class`),
 - hlavní funkce `main` (veřejná statická metoda, `public static method`)
- Soubor musí mít jméno shodné se jménem veřejné třídy a příponu `.java`
- Hlavička funkce `main ()`:
 - klíčová slova `public static void` (`void` - procedura, nic nevrací)
 - `(String[] args)` specifikace parametrů zadané při spuštění
- Konvence: jména tříd se píší s prvním velkým písmenem

Java Platforma (JRE) = Java Core API + JVM



Vývoj programů v Javě

- JRE - běhové prostředí, JRE = JVM +API
pro běh programů
 - JVM – Java Virtual Machine – virtuální stroj
 - API – Application Programming Interface - knihovny
- JDK - Software Development Kit pro Javu
 - sada základních nástrojů pro vývoj programů v Javě
 - JRE, překladač javac, debugger, javadoc, nástroje pro vytváření jar archivů, mnohé další
 - ke stažení na <http://java.sun.com>

JAVA – struktura programu

```
public class Main {
```

```
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n;  
        System.out.println("Zadej počet ");  
        n = sc.nextInt();  
        while (n > 0) {  
            System.out.println("Vypis " + n);  
            n = zmensi(n);  
        }  
    }
```

```
    static int zmensi(int n) {  
        n=n-1;  
        return n;  
    }
```


Datové typy

- Při návrhu algoritmů a psaní programů ve vyšších programovacích jazycích abstrahujeme od binární podoby paměti počítače
- S daty pracujeme jako s hodnotami různých datových typů, které jsou uloženy v datových objektech
- Datový typ (zkráceně jen typ) specifikuje:
 - množinu hodnot (+ způsob reprezentace)
 - množinu operací, které lze s hodnotami daného typu provádět
- Příklad typu: celočíselný typ `int` v jazyku Java:
 - množinou hodnot jsou celá čísla z intervalu `-2147483648 .. 2147483647`
 - množinu operací tvoří
 - aritmetické operace `+`, `-`, `*`, `/`, jejichž výsledkem je hodnota typu `int`
 - relační operace `==`, `!=`, `>`, `>=`, `<`, `<=`, jejichž výsledkem je hodnota typu `boolean`
 - a další
- Typ `int` je jednoduchý typ, jehož hodnoty jsou atomické (z hlediska operací dále nedělitelné)



graficky

Zpracování programu v jazyku JAVA

Program.java

```
public class Program {  
    public static void main(String[] args) {  
        System.out.println("Nazdar,toto je první program");  
    }  
}
```

Spuštění překladače do byte-code:

```
javac Program.java
```

vznikne:

```
Program.class
```

spuštění interpretru:

```
java Program
```

Nazdar, toto je první program

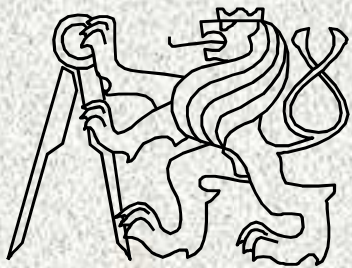
Jména a konvence

- **balíček** - `package` - jen malá písmena, i několik jmen oddělených tečkou, např: `prog1`, `java.util`
- **třída, abstraktní třída, jejich konstruktory, rozhraní** - `class`, `abstract class`, `interface`
 - jméno začínající velkým písmenem, např: `String`, `MyFirstClass`, `Serializable`, `Comparable`
 - Výjimky by měly mít sufix `Exception`, např: `MySpecialException`
- **metoda** - sloveso začínající malým písmenem, další slovo začíná velkým písmenem, např: `setBorder`, `isEmpty`, `getNumber`
- **proměnné** - začínající vždy malým písmenem další slovo začíná velkým písmenem, např: `diskriminant`, `totalCount`
- **konstanta** - jen velká písmena, jednotlivá slova oddělena podtržítkem, např. `MAX_COUNT`, `RED`

Ukázka vývojového prostředí

toto je nejlepší ukázat v NetBeans

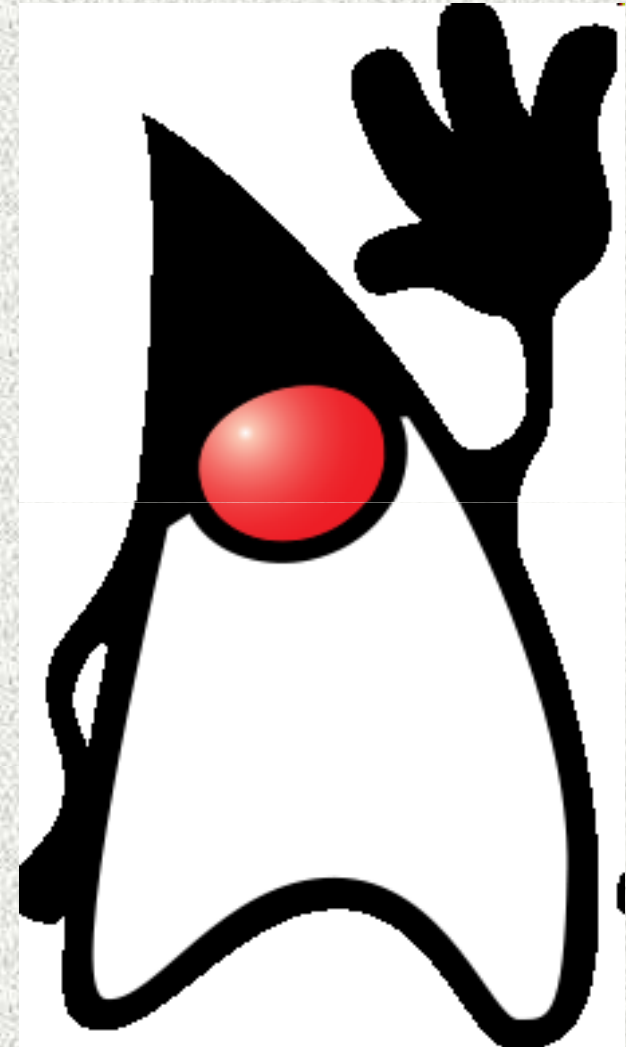
Java – základy konec



A0B36PR1-Programování 1
Fakulta elektrotechnická
České vysoké učení technické

Java - historie

- v roce 1990 vznikl Green Team vedený Jamesem Goslingem – jazyk Oak (dub)
 - původně navržen pro vestavěné systémy
- Java představena firmou Sun Microsystems 23. května 1995
 - navržen pro vývoj aplikací na webu
- 1996 vydán první Java Development Kit 1.0 (pro aplety)
 - důležité využití
- 2008 - JDK 6 Update 13 s JavaFX SDK
- 2009 – JDK , 1.6.0_14;
- **2011 – JDK 7 !!!**
- **2012 – JDK 7.1 !!!**



Duke, maskot Javy

Java - edice

- každá edice je určena pro specifické účely, pro specifická zařízení
 - Java Card - smart (chytrý) karty - platební a kreditní karty
 - J2ME Java 2 Micro Edition – midlety, Kilobyte VM, mobilní zařízení, PDA, set-top boxy, vyžaduje 160 kb ROM a 32 kB RAM (CLDC, MIDP)
 - J2SE standard edition – stolní počítače
 - J2EE enterprise edition – podnikové aplikace