Obsah 2. přednášky:

- Číselné soustavy
- Kódování
- Datové typy a Java
- Deklarace proměnné
- Výraz, přiřazení, příkaz
- Operátory
- Konverze typu
- Základní matematické funkce
- Terminálový formátovaný vstup a výstup

Tato tematika je zpracována v

Záznamy přednášek: str. 14 - 44

Problém:

Je dán počet testovaných případů *t* ($1 \leq t \leq 100$) a pro každý takový případ je na samostatném řádku zadáno celé číslo *n* ($-1000 \leq n \leq 1000$).
Pro všechna zadaná *n* spočtěte a na samostatném řádku vypište výsledek následujícího úkolu:
Vynásobte číslo *n* 567mi, poté vydělte výsledek 9, přičtěte 7492, pak vynásobte 235, výsledek vydělte 47 a odečtěte 498. Jaká číslice je ve vypočtené hodnotě na řádu desítek?

Úvod k řešení bude uveden na konci této přednášky.

Číselné soustavy

Př. **desítková soustava**

$$5321 = 5 \cdot 10^3 + 3 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0$$

Zápis celého čísla

$$N = a_{m-1} Z^{m-1} + a_{m-2} Z^{m-2} + ... + a_1 Z^1 + a_0 Z^0$$

Zápis desetinné části

$$N = a_{-1} Z^{-1} + a_{-2} Z^{-2} + ...a_{-n} Z^{-n}$$

kde: **Z** … základ, **m** … počet řádových míst , $\mathbf{a_i}$ … koeficient
n … počet desetinných míst

Kapacita soustavy $K = Z^m$

Největší hodnota soustavy $N_{MAX} = Z^m - 1$

Významné soustavy

Základ	Soustava	Hodnoty
10	desítková (dekadická)	0,1,2,3,4,5,6,7,8,9
2	dvojková (binární),	0,1
8	osmičková (oktalová)	0,1,2,3,4,5,6,7
16	šestnáctková (hexadecimální)	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Šestnáctková soustava:

($\mathbf{A}$)$_{16}$=(10)$_{10}$ ($\mathbf{D}$)$_{16}$=(13)$_{10}$
($\mathbf{B}$)$_{16}$=(11)$_{10}$ ($\mathbf{E}$)$_{16}$=(14)$_{10}$
($\mathbf{C}$)$_{16}$=(12)$_{10}$ ($\mathbf{F}$)$_{16}$=(15)$_{10}$

Převody

- převádíme odděleně celou a desetinnou část
- převod dekadického čísla 25,625 = $x_{(2)}$ (k=2)

x_c	x_c/k	x_c mod k	x_d	x_d*k
25	12	1	0,625	1,250
12	6	0	0,250	0,5
6	3	0	0,5	1,0
3	1	1		
1	0	1		

$25{,}625_{(10)} = 11001{,}101_{(2)}$

Převody mezi soustavami o $z = 2^N$

$11001_{(2)} = 31_{(8)}$ **$11001_{(2)} = 19_{(16)}$**

Kódování

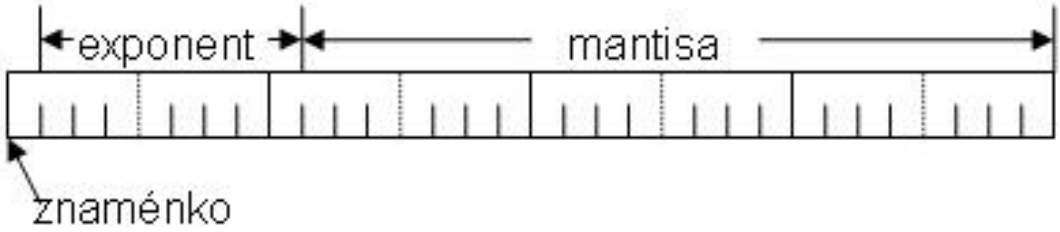
- podrobně bude předneseno na konci semestru

Kódy pro celá čísla

– přímý
– inverzní
– doplňkový
– s posunutou nulou

Kódy pro reálná čísla

- jednoduchá a dvojnásobná přesnost



Kódy pro znaky ASCII

- číslice vs. číslo, znak vs. hodnota ASCII kódu !

Kódy pro logické hodnoty

Datové typy a Java

datový typ = množina hodnot +množina operací

primitivní datový typ – číslo/znak/logická hodnota

Java

Číselné typy – celá čísla

Typ	Rozsah
byte	-128..127
short	-32768..32767
int	-21474836448..2147483647
long	-9223372036854775808..9223372036854775807

Číselné typy – reálná čísla

Typ	Rozsah
double	4.9E-324..1.797693134862315 7E308
float	1.4E-45 .. 3.4028235E38

Číselné konstanty

Double.POSITIVE_INFINITY (+∞)
Double.NEGATIVE_INFINITY(-∞)
Double.NaN (Not a Number)
Integer.MAX_VALUE (největší hodnota)
Double.MIN_VALUE (nejmenší hodnota)

Znakové typy a konstanty

char
znaková konstanta je uzavřena v apostrofech 'A', '1', '%'

Řetězcové konstanty

String je v Javě třída, proto
zatím budeme řetězce užívat pouze pro výstup, např.
"tento retezec" a "tento" + "retezec" (pozor na typ uvozovek!)

Logický typ a konstanty

boolean

true (logická 1), false (logická 0)

&& logický součin, || logický součet, ! negace

A	B	A && B	A || B	!A
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

Poznámka – datový typ výčet (Záznamy str.66) – pro zájemce

Deklarace proměnné

Deklarace

- stanovení symbolického jména a datového typu
- deklarace s inicializací (stanovení počáteční hodnoty)
- pojmenovaná konstanta (hodnota se nemění, `final`)

Proměnná

= symbolicky pojmenovaný datový objekt v operační paměti

Identifikátor - jméno proměnné, konstanty, metody, třídy...

Pravidla pro zápis

- neomezená délka
- začíná písmenem, podtržítkem (_) nebo dolarem ($)
- nesmí obsahovat operátory
- nelze použít rezervovaná slova

Konvence v Javě – nutno dodržovat

třídy – začínají velkým písmenem (další slovo velkým)
metody a proměnné – začínají malým písmenem
konstanty – všechna písmena velká, podtržítka
balíky – pouze malá písmena

Výraz, přiřazení, příkaz

Výraz

- předepisuje postup při výpočtu hodnoty určitého typu

skládá se z operandů (proměnné, konstanty, volání metod ...) a operátorů (+,-,*,/, ...)

Přiřazení a příkaz

- nastavení hodnoty **=** (není to rovnice)

proměnná **=** výraz; - toto celé je příkaz

Příklady:

```
final char MEZERA = ' ';
int pocet = 0;
pocet = pocet +3;
```

```
// vypocet plochy
int polomer = 1;
double plocha = polomer*polomer*3.14;
System.out.println("P("+polomer+")="+plocha);
// vypocet dalsi plochy
polomer = 2;
plocha = polomer*polomer*3.14;
System.out.println("P("+polomer+")="+plocha);
```

```
P(1) =3.14
```

```
P(2) =12.56
```

Operátory

Operace s operandy určují operátory

- unární, binární, ternární (1)
- aritmetické, relační, přiřazovací, bitové

Vyhodnocení výrazu zleva doprava dle priorit

Priority možno upravit použitím závorek

Priorita operátorů

⇩ ++, --
⇩ *, /, %
⇩ +, -
⇩ <, <=, >, =>
⇩ ==, !=
⇩ &&
⇩ ||
⇘ =, +=, -=, *=, /=, %=

Poznámka:

/ .... celočíselné vs. reálné

Příklad: vzdálenost dvou bodů v rovině

$$v = \sqrt{(x_1 - x_2)^2 \cdot (y_1 - y_2)^2}$$

```
int x1 = 0, y1 = 0;
int x2 = 1, y2 = 1;
double v = Math.sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
System.out.println("v = " + v);
```

v = 1,41...

Konverze typu

proměnné lze přiřadit jen hodnotu stejného typu

jinak **přetypování**

- implicitní (automaticky)
- explicitní (nutno napsat), jen pro danou operaci, typ proměnné se trvale nezmění!

```
double r,p;
r = 3;      // implicitni - rozsireni typu
int i = 1;
p = i;      // implicitni
p = (double) i; // lepe explicitne, je to umysl
i = (int) r;  // explicitni – ztrata rozsahu
i = (int) p + r;     // chybne!!!
i = (int) (p + r);  // spravne
```

Základní matematické funkce

Standardní funkce (Třída Math)

- mocnina
- druhá odmocnina
- goniometrické funkce
- logaritmus
- exponenciální fce

Konstanty - π, e ... viz Java Core API.

Příklady použití standardních funkcí:

```
double pi = Math.PI;  //3.141592653
double e = Math.E;   // Math.exp(1);
int a =3, b = 4;
double prepona = Math.sqrt(Math.pow(a,2)+b*b); // 5
double v =
Math.pow(Math.sin(1),2)+Math.pow(Math.cos(1),2); // 1
double logaritmusDes = Math.log10(100);   // 2
double logaritmusPriroz = Math.log(Math.E); // 1.71...
double nahodneCisloX = Math.random(); // 0.0 ≤X≤ 1.0
```

Příklad:

```
// minimum ze 3 cisel
int a = 1, b = 2, c = 3 ;
int min = Math.min (Math.min (a, b),c);
System.out.println("min = " + min);
```

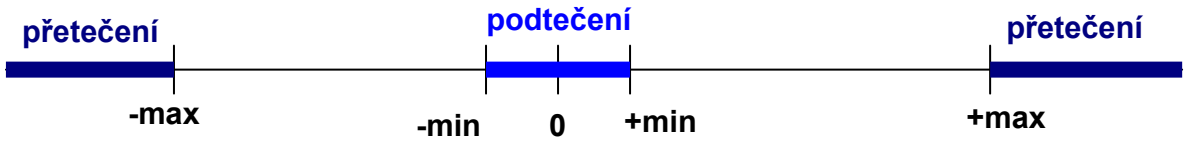
min = 1

Důležitá poznámka

Problémy při provádění aritmetických operací

1. Přetečení a podtečení čísla

Je-li hodnota exponentu čísla
větší než maximální zobrazitelná dochází k **přetečení**
menší než minimální zobrazitelná dochází k **podtečení**



2. Porovnání dvou reálných čísel

double x, y; **x == y !!** **nikdy takto!!!**

- upravit prostřednictvím <=, >=
- **if** (x == y) nahradit **if** (Math.abs(x - y) < ε)
  kde: ε absolutní chyba porovnání

3. Problém zaokrouhlení (důvod: zobrazení čísel)

Nevhodným řazením operací násobení a dělení může dojít vlivem zaokrouhlení k výrazným chybám.

```
System.out.println(1/Math.sqrt(3) * Math.sqrt(78)); // ne!
System.out.println(Math.sqrt(78) / Math.sqrt(3));    // ok
// 5.099019513592786
// 5.099019513592785
```

Generátor náhodných čísel

(pseudo)náhodná čísla(celá, reálná)–třída Random

- nutný import balíčku java.util
- inicializace = vytvoření objektu
  (obdobně jako u DrawingTool)

Příklad:

```
import java.util.*;
public class NahodnaCisla {
  public static void main (String[]args) {
    Random r = new Random();
    // Random r = new Random(1);
    // stale stejna nahodna posloupnost

    double realne = r.nextDouble();  //nema parametr
    int pocetCisel = 10; // 0,1,2,3,4,5,6,7,8,9
    int cele = r.nextInt(pocetCisel);  //muze mit parametr
    System.out.println("" + realne + " : " + cele);
    long velkeCele = r.nextLong();  //nema parametr
    System.out.println("" + velkeCele);
  }
}
```

Poznámka:

r je proměnná - reference třídy Random

nextInt() - metoda vrací náhodné celé číslo typu **int**

nextInt(celeCislo) vrací náhodné celé číslo v rozsahu ⟨0, celeCislo-1⟩

nextDouble() je metoda třídy Random vracející náhodné reálné číslo ⟨0.0, 1.0⟩

Terminálový formátovaný Vst/Výst

terminálový

- velmi primitivní V/V (klávesnice, obrazovka - txt režim)

formátovaný

– čísla jsou na výstupu automaticky převedena na řetězec číslic v desítkové soustavě

Klasický výstup

System.out.print(parametr), System.out.println(),

System.out.println(parametr)

print(parametr) – metoda třídy System, pro výstup

parametr – primitivní typ nebo řetězec

funkce - pro každý primitivní typ existuje implicitní konverze na řetězec

```
int i=1;
System.out.print ("i = " +i);
System.out.println('A');        // i = 1A
System.out.println(Math.E);
System.out.println(5);
System.out.println(5.25);
System.out.println("Hello world\n");  // "\n" odradkovani

System.out.println(5 + i + 'A');
// ALE POZOR: 71=5+1+65 (65=kod pismena velke A)
System.out.println("" + 5 + i + 'A'); // 51A  ok!

System.out.println("index = " + i + 2);
// ALE jeste jednou POZOR: chci soucet, ale vypis = 12
System.out.println("index = " + (i+2)); //  3 ok!
```

Lepší řešení výstupu (od JDK 1.5)

Použití metody - System.out.format(parametr)

pozor: parametry jsou odděleny , (čárkou) nikoliv +

Používat jen, je-li to nutné!

Více informací:

Záznamy přednášek (str. 36 – 38) + java.util.Formatter

Příklad (program + výstup):

```
import java.util.Formatter;

public class FormatovaniVystupu {
  public static void main(String [] args) {

    System.out.format("nova radka%n");

    int i = -1234;
    System.out.format("i = %d%n", i);
    System.out.format("i = %7d%n", i);
    System.out.format("i = %-7d%n", i);
    System.out.format("i = %+7d%n", i);
    System.out.format("i = % 7d%n", i);
    System.out.format("i = %07d%n", i);
    System.out.format("i = %, 7d%n", i);

    char c = 'a';
    System.out.format("c = %c%n", c);
    System.out.format("c = %3c%n", c);
    System.out.format("c = %C%n", c);
    System.out.format("c = %c%n", c);
    System.out.format("c = %c%n", 65);
```

... pokračování na dalším slidu

```
    double d = 1234.567;
    System.out.format("d = %f%n", d);
    System.out.format("d = %g%n", d);
    System.out.format("d = %e%n", d);
    System.out.format("d = %10.1f%n", d);
    System.out.format("d = %-10.1f%n", d);

    int j = 30;
    System.out.format("j = %o%n", j);
    System.out.format("j = %x%n", j);
    System.out.format("j = %X%n", j);
    System.out.format("j = %3X%n", j);
    System.out.format("j = %#x%n", j);

    String s = "ahoj lidi";
    System.out.format("s = |%s|%n", s);
    System.out.format("s = |%S|%n", s);
    System.out.format("s = |%11s|%n", s);
    System.out.format("s = |%-11s|%n", s);
    System.out.format("s = |%.3s|%n", s);
    System.out.format("s = |%11.3s|%n", s);

    System.out.format("Pivo \"lezak\" ma 12%%%n");
    System.out.format("Znak 'backslash' je \\%n");

    j = 30;
    System.out.format("%d = %o = %X%n", j, j, j);
    System.out.format("%1$d = %1$o = %1$X%n", j);
    System.out.format("%d = %<o = %<X%n", j);

    String zformatovanyRetezec =
                                  String.format ("%6d", 123);
    System.out.println("konec");
  }
}
```

```
nova radka
i = -1234
i =   -1234
i = -1234
i =   -1234
i =   -1234
i = -001234
i =  -1 234
c = a
c =   a
c = A
c = a
c = A
d = 1234,567000
d = 1234.57
d = 1.234567e+03
d =     1234,6
d = 1234,6
j = 36
j = 1e
j = 1E
j =  1E
j = 0x1e
s = |ahoj lidi|
s = |AHOJ LIDI|
s = |  ahoj lidi|
s = |ahoj lidi  |
s = |aho|
s = |       aho|
Pivo "lezak" ma 12%
Znak 'backslash' je \
30 = 36 = 1E
30 = 36 = 1E
30 = 36 = 1E
konec
```

Formátovaný vstup

od. JDK 1.5 - jednoduché řešení třída Scanner, použitelné i pro soubory (bude uvedeno později)

postupujeme obdobně jako u třídy Random

- import balíčku java.util
- inicializace, vytvoření objektu (**new**)
- nutno řešit lokalizaci (. vs. , v reálném čísle)
- používá metody:

nextInt(), nextDouble(), next(), nextLine()

Příklad:

Načti dvě čísla (reálné, celé) a spočti jejich průměr.

```
import java.util.*;

public class VstupScanner {
    public static void main(String [] args) {

        Scanner sc = new Scanner(System.in);
        sc.useLocale(Locale.US); // lokalizace !

        System.out.print("Zadej 1. cislo realne: ");
        double cislo1 = sc.nextDouble();
        System.out.println("Zadano bylo: " + cislo1);

        System.out.print("Zadej 2. cislo cele: ");
        int cislo2 = sc.nextInt();
        System.out.println("Zadano bylo: " + cislo2);

        double prumer = (cislo1 + cislo2)/2;
        System.out.println("Prumer = " + prumer);
    }
}
```

Příklad (problém vyprázdnění vstupu)

```
import java.util.*;
public class VyprazdneniVstupu {
    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        sc.useLocale(Locale.US);
        System.out.print("Zadej cele cislo: ");
        int cislo = sc.nextInt();
        System.out.println("Zadano bylo: " + cislo);
        System.out.print("Zadej realne cislo: ");
        double realne = sc.nextDouble();
        System.out.println("Zadano bylo: " + realne);
        sc.nextLine();
   // docteni radky <cr> a <lf> - vyprazdneni vstupu
        System.out.print("Zadej vetu: ");
        String veta = sc.nextLine();
        System.out.println("Zadano bylo: " + veta);
        System.out.print("Zadej znak: ");
        char znak = sc.next().charAt(0);
        System.out.println("Zadano bylo: " + znak);
        System.out.print("Zadej slovo: ");
        String slovo = sc.next();
        System.out.println("Zadano bylo: " + slovo);
    }
}
```

Výstup 1. programu:

```
Zadej 1. cislo realne: 12.3
Zadano bylo: 12.3
Zadej 2. cislo cele: 12
Zadano bylo: 12
Prumer = 12.15
```

Výstup 2. programu:

```
Zadej cele cislo: 12
Zadano bylo: 12
Zadej realne cislo: 12.2
Zadano bylo: 12.2
Zadej vetu: Ahoj, jak se mas?
Zadano bylo: Ahoj, jak se mas?
Zadej znak: w
Zadano bylo: w
Zadej slovo: lepidlo
Zadano bylo: lepidlo
```

Metody nextInt(), nextDouble() a next() jsou tzv. „žravé“ – přečtou a zlikvidují všechny „bílé“ znaky (mezera, znaky konce řádky, tabulátor)

Řešení

před použitím metody nextLine() vyprázdnit vstup příkazem sc.nextLine()

Pozor – ale:
metoda nextLine() není žravá, ale za to zlikviduje znaky konce řádky, proto se za ní nesmí použít příkaz sc.nextLine()!

Vyprázdnění vstupu bývá častou chybou!

Správné použití třídy Scanner

```
public class PouzitiScanneru{
    static private  Scanner sc = new Scanner(System.in);
    public static void main (String [] args) {
        int a = sc.nextInt();
    }
}
```

Takto budeme Scanner používat při vícenásobném volání (např. v main() a v nějaké další metodě) – více bude vysvětleno později.

Vstup s použitím argumentů (příkazová řádka)

```
public class VstupArgumenty {
 //  cisla zadana jako parametry v prikazove radce
  public static void main(String[] args) {
    int a = Integer.parseInt(args[0]);
    System.out.print(" a = " + a);
    int b = Integer.parseInt(args[1]);
    System.out.print(" b = " + b);
    int suma = a + b;
    System.out.println("suma = " + suma);
  }
}
```

Na příkazové řádce spustíme: java VstupArgumenty 5 7

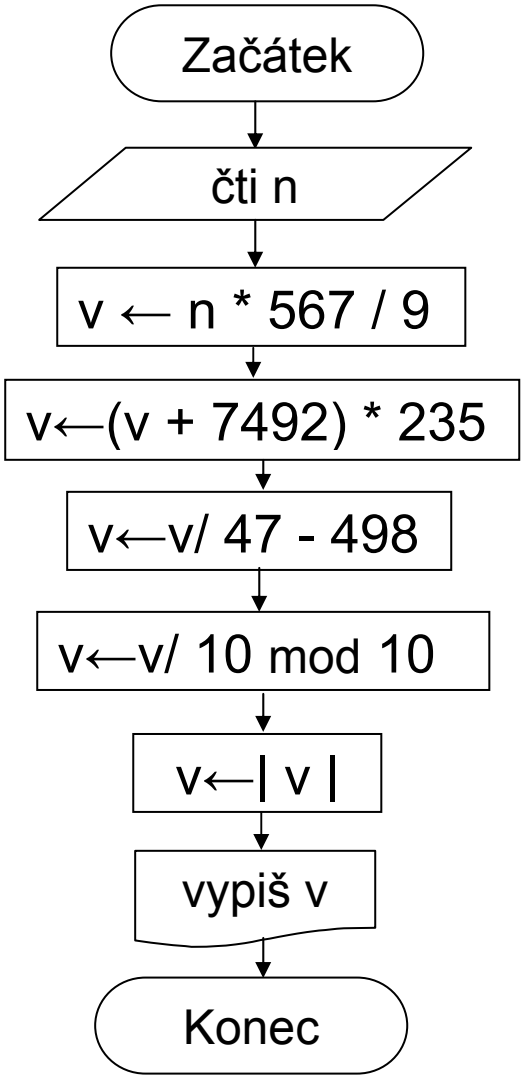
metoda parseInt() provádí převod řetězce na celé číslo (bude vysvětleno později - se třídou String)

Jak na problém?

Použijeme znalosti z dnešní přednášky.

Budeme řešit pouze:

Pro zadané *n* spočtěte a na samostatném řádku vypište výsledek následujícího úkolu:
Vynásobte číslo *n* 567-mi, poté vydělte výsledek 9, přičtěte 7492, pak vynásobte 235, výsledek vydělte 47 a odečtěte 498. Jaká číslice je ve vypočtené hodnotě na řádu desítek?




```
načti n
v ← n * 567 / 9
v ← (v + 7492) * 235
v ← v/ 47 – 498
v ← v/10 mod 10
v ← | v |
vypiš v
```

```
int n = sc.nextInt();
int v = (n * 567 / 9 + 7492) * 235 / 47 – 498;
v = v/10 % 10;
v = Math.abs(v);
System.out.println(v);
```

Pokračování na konci další přednášky ...