Algoritmizace a programování

Řídicí struktury, standardní metody

- Problematika načítání pomocí `Scanner`
- Některé poznámky k příkazům
- Psaní kódu programu
- Metody třídy `Math`
- Obalové třídy primitivních datových typů

Verze pro akademický rok 2012/2013

Terminálový vstup

- Problematika načítání z terminálového vstupu
- Po potvrzení zadávaných hodnot stiskem <ENTER> je vstupní textový řetězec uložen do vstupního bufferu
- Při načítání jsou postupně zpracovávány znaky ze vstupního bufferu

- Metody `nextInt()`, `nextFloat()`, `next()` ...
  - Ignorují všechny počáteční „bílé znaky“
  - Čtení textového řetězce do dalšího prvního „bílého znaku“
  - Konverze textového řetězce na příslušnou hodnotu

- Metoda `nextLine()`
  - Čte textový řetězec do konce řádku

Přiřazení

- Hodnotu proměnné lze použít ve výrazu pouze v případě, že jí předtím byla přiřazena hodnota

- Přiřazení hodnoty
  - Operace přiřazení
  - Načtení hodnoty (z proudu – z terminálového vstupu, ze souboru atp.)

- Použití předchozí hodnoty proměnné je skryto i v následujících výrazech
  - Inkrementace `i++` tj. v podstatě `i = i + 1`
  - Přiřazení `s *= x` `s = s * x`

Aritmetika celých čísel

- Celočíselný typ – interval celočíselných hodnot zobrazitelných daným typem
- Každá hodnota příslušného typu – v paměti počítače zobrazena přesně
- Porovnání hodnot
- Aritmetické operace – může docházet k přetečení
  - Aritmetické operace – *sčítání, odčítání, násobení, dělení, modulo* – pokud je jeden operand typu `long`, výsledek typ `long`, jinak je výsledek typu `int`

Aritmetika reálných čísel

- Reálný typ – interval reálných hodnot zobrazitelných daným typem
- Hodnota příslušného reálného typu – v paměti počítače nemusí být zobrazena přesně – garantovaný počet platných číslic
- *Mantisa, exponent* – omezení mantisy – zaokrouhlování, chyby
- Vytvoření hodnoty reálného typu – konverze do formátu definovaného standardem
- Problém práce s reálnými čísly je nepřesnost
- Aritmetické operace
- Nelze se spoléhat na to, že výsledkem dvou reálných výrazů představujících stejnou hodnotu budou dvě totožná čísla
- Porovnání hodnot
  - Např. testování rovnosti – testujeme, zda daná čísla jsou „skoro stejná“ – s určitou přesností
  - Test `if (x == y) {…}`
  - Můžeme nahradit testem `if (Math.abs(x - y) < skoroNula) {…}`

Logické příkazy

- Příkazy jazyka Java mohou být libovolně vnořené
- Je třeba dávat pozor na vnořené logické příkazy v případě střídání úplných a neúplných podmínek (`if-else` a `if` příkazy)
  - Větev `else` se vždy přiřazuje k nejbližšímu možnému `if` – skutečná interpretace by potom mohla jiná než jak zamýšlel programátor
  - Interpretaci shodnou se záměrem programátora vynutíme v případě potřeby použitím bloku příkazů v jednotlivých větvích logických příkazů
  - Je dobrým zvykem používat bloky příkazů ve všech větvích (nebo na místě příkazu těla cyklu) a to i v případě, že na daném místě zapisujeme pouze jeden jediný příkaz

```
if (<podminka1>)
    if (<podminka2>) <prikaz1>;
else <prikaz2>;
```

```
if (<podminka1>){
    if (<podminka2>) <prikaz1>;
} else <prikaz2>;
```

Příkazy cyklu

- Vzájemné porovnání příkazů cyklu
  - Všechny cykly jazyka Java jsou ukončované podmínkou
  - Tělo cyklu `do-while` se provede vždy alespoň jednou, podmínka je testována po provedení kódu těla cyklu
  - Příkaz `for` – vzhledem ke své inicializační a inkrementační části umožňuje v některých případech kompaktnější zápis
  - Cykly jsou navzájem zaměnitelné – tedy libovolný příkaz cyklu můžeme přepsat pomocí libovolného jiného příkazu cyklu
- Při konstrukci cyklů je nutné se vyvarovat vytvoření nežádoucího nekonečného cyklu

Psaní kódu

- Pojmenování – identifikátory – proměnných, tříd, metod, aplikace a podobně volíme tak, aby byl program dobře čitelný
- Při zápisu identifikátorů dodržovat nepsaná pravidla pro použití malých a velkých písmen v identifikátorech
- Deklarace proměnných používaných v algoritmu umístit na začátek metody – oddělit prázdným řádkem od vlastního kódu
- Deklarace pomocných proměnných, jejichž platnost je omezena pouze na jediný blok (například řídicí proměnné zavedené pro zajištění příslušného počtu opakování cyklu `for`) deklarujeme zpravidla na začátku příslušného bloku (inicializační část cyklu `for`)

- Dodržovat nepsaná pravidla formátování kódu – je vhodné využít automatické formátování nabízené používaným vývojovým prostředím (plovoucí menu po stisku pravého tlačítka myši v okně editoru – volba „format“)

Přehled některých tříd

- Jazyk Java – objektový
  - Třída – základní stavební prvek programu v jazyce Java
  - Třída ~ obecný typ – chceme-li použít nástroje, které jsou v dané třídě k dispozici je nutné deklarovat proměnnou příslušného typu třídy a vytvořit instanci třídy – pro tuto instanci potom máme k dispozici vše, co třída obsahuje
  - Třída ~ knihovna funkcí – použití funkcí (prostředků) bez vytváření instance dané třídy

- Programovací jazyky mají k dispozici zpravidla knihovny standardních funkcí
  - Jazyk Java neumožňuje vytváření knihoven, nicméně můžeme vytvářet třídy, jejichž nástroje (metody, konstanty, proměnné) lze používat i bez vytváření instance příslušné třídy – *knihovní třída*
  - Jednou z *knihovních tříd* je i třída `Math`, která obsahuje řadu metod pro výpočet matematických funkcí

Přehled konstant a metod třídy `Math`

E – konstanta přirozeného základu logaritmu, `double`

PI – konstanta Ludolfova čísla s přesností `double`

`abs` – absolutní hodnota čísla

`max` – větší ze dvou hodnot

`min` – menší ze dvou hodnot

`sin` – goniometrická funkce sin – parametr v radiánech

`cos` – goniometrická funkce cos

`tan` – goniometrická funkce tg

`asin` – inverzní funkce k funkci sin – výsledek v radiánech

`acos`

`atan`

`sinh` – hyperbolický sinus

`cosh`

`tanh`

`toDegrees` – převod radiánů na stupně

`toRadians` – převod stupňů na radiány

`pow` – obecná mocnina čísla

`sqrt` – druhá odmocnina čísla

`exp` – exponenciélní funkce

`log` – logaritmická funkce – o základu e

`log10` – dekadický logaritmus

`random` – generování náhodného čísla v rozsahu 0 – 1

`ceil` – celá část reálného čísla, která je menší než parametr (parametr i výsledek `double`)

`floor` – celá část reálného čísla, která je větší než parametr

`rint` – celé číslo nejbližší předanému argumentu

`round` – zaokrouhlená hodnota reálného čísla – výsledek je typu `long`

`signum` – vrací 0.0, -1.0, 1.0 dle znaménka argumentu

`getExponent` – vrací exponent reálného čísla

Použití metod třídy `Math`

- Metody vrací hodnotu definovaného typu svým jménem – volání metod použijeme obecně ve výrazech
- Každá z uvedených metod (kromě metody `random`) má parametry – pořadí a typ parametrů musí odpovídat deklaraci – podrobnější popis v nápovědě
- Některé z uvedených metod jsou k dispozici ve více verzích – například metoda `abs` – tyto verze se liší typem argumentu, typem vracené hodnoty – jedná se o tzv. *přetížené metody* – při volání je potom podle typu skutečného argumentu rozhodnuto, která verze metody bude použita
- Bližší popis metod v nápovědě jazyka Java

```
float r, obvod;
r = (float) Math.random()*100; // náhodné realne v intervalu <0, 100)
obvod = (float) (2 * Math.PI * r);
int n = (int) (Math.random() * 100); // náhodné celé od 0 do 99
double alfa = 60;       // uhel ve stupnich
double sinAlfa = Math.sin(Math.toRadians(alfa));
sinAlfa = Math.sin(alfa * Math.PI/180);
```

Obalové třídy primitivních datových typů

- Java objektový jazyk
- Primitivní datové typy – `float`, `double`, `byte`, `short`, `int`, `long`, `char`, `boolean` – proměnné primitivních datových typů nejsou objekty
- Ke každému primitivnímu datovému typu má jazyk Java *obalovou třídu* (*wrapper class*, obalující/obalovou třída)

Typ	Obalová třída
`float`	**`Float`**
`double`	**`Double`**
`byte`	**`Byte`**
`short`	**`Short`**
`int`	**`Integer`**
`long`	**`Long`**
`char`	**`Character`**
`boolean`	**`Boolean`**

- *Obalové třídy* obsahují konstanty, metody, umožňují konverzi hodnoty příslušného typu na objekt

Třídy `Float` a `Double`

- Vybrané konstanty

`MIN_VALUE`
`MAX_VALUE`
`MIN_EXPONENT`
`MAX_EXPONENT`
`NEGATIVE_INFINITY` – hodnota, která je výsledkem dělení záporného čísla nulou
`POSITIVE_INFINITY`
`NaN` – tato hodnota je výsledkem operace dělení nuly nulou
`SIZE` – velikost hodnoty typu v bitech

- Použití

```
float min = Float.MAX_VALUE;
```

- Vybrané statické metody typu `Double` (`Float` obdobně)

`compare(d1, d2)` – výsledek typu `int`
`isInfinite(d)` – výsledek typu `boolean`
`isNaN(d)` – výsledek typu `boolean`

`parseDouble(s)` – výsledek typu `double`
`toString(d)` – výsledek typu `String`

`valueOf(s)` – výsledek typu `Double`
`valueOf(d)` – výsledek typu `Double`

- Použití

```
v = sum / n;
if (Float.isInfinite(v)) …;
else …;
```

Třídy `Byte`, `Short`, `Integer` a `Long`

- Vybrané konstanty

`MIN_VALUE`
`MAX_VALUE`

`SIZE` – velikost hodnoty typu v bitech

- Použití

```
int max = Integer.MIN_VALUE;
```

- Některé statické metody

`parseInt(s)` – výsledek typu `int`
`toString(d)` – výsledek typu `String`

`valueOf(s)` – výsledek typu obalové třídy
`valueOf(d)` – výsledek typu obalové třídy

- Použití – před jménem metody musí být uveden identifikátor odpovídající třídy oddělený od identifikátoru metody tečkou

```
int n = Integer.parseInt(“126”);
```

Třída `Character`

- Třída Character obsahuje řadu konstat
- Vybrané konstanty

`MIN_VALUE`
`MAX_VALUE`
`SIZE` – velikost hodnoty typu v bitech

- Použití konstant a metod
  - před jménem statické metody třídy nebo konstanty musí být uveden identifikátor odpovídající třídy oddělený od identifikátoru metody resp. konstanty tečkou
  - před jménem instanční metody (konstanty, instanční proměnné) musí být uveden identifikátor proměnné odpovídajícího objektového typu oddělený od identifikátoru metody tečkou

- Některé metody

`valueOf(c)` – výsledek typu obalové třídy

```
isLetter(c)
isLetter(i)
isLowerCase(c)
isUpperCase(c)
isSpaceChar(c)
isWhitespace(c)
toUpperCase(c)
toLowerCase(c)
```

Třída Boolean

- Vybrané konstanty

`FALSE` – konstanta deklarovaná ve třídě `Boolean` typu `Boolean`

`TRUE` – konstanta typu `Boolean`

Poznámka:

`false, true` – toto jsou rezervovaná slova jazyka Java představující konstantní hodnoty primitivního typu `boolean`

- Některé metody

`parseBoolean(s)` – výsledek typu `boolean`

`toString(b)` – výsledek typu `String`

`valueOf(s)` – výsledek typu `Boolean`

`valueOf(b)` – výsledek typu `Boolean`

Chyby v programu

- Chyby syntaktické
  - Syntaktický analyzátor
  - V době kompilace
  - Netbeans – zvýraznění v editoru – v době psaní kódu je prováděna syntaktická analýza
- Chyby běhové
  - Dělení nulou v oboru celých čísel
  - Odmocnina ze záporných čísel (aplikace funkcí na hodnoty mimo definiční obor)
  - Převod textových řetězců na čísla
  - Chyba – přerušení vykonávání kódu na daném místě – generování výjimky – výjimku lze v programu ošetřit
  - Pokud zůstane výjimka neošetřena, program vypíše standardní chybové hlášení a skončí
  - V případě některých byť chybných operací k chybě nedochází – přetečení rozsahu typu
- Chyby sémantické
  - Program lze přeložit, spustit,
  - Pro zadaná vstupní data (popřípadě pro některé kombinace zadaných vstupních dat) poskytuje chybné výsledky