4. Pole

Jednorozměrné pole

Pole je datová struktura, která nám umožňuje pracovat s větším množstvím hodnot stejného typu. Pole si lze představit jako řadu hodnot. Pole má jen jeden identifikátor (jméno), pro práci s jednotlivými položkami používáme indexy. Následuje příklad pole se jmény dnů v týdnu.

pondělí	úterý	středa	čtvrtek	pátek	sobota	neděle	← **hodnota**
0	1	2	3	4	5	6	← **index**

tabulka 7 Ukázka pole

Pole v Javě je zvláštní druh objektu a tudíž podobně jako u objektů se rozlišuje deklarace pole od vytvoření instance pole (alokace pole). Jednorozměrné pole lze deklarovat takto (dvě varianty pro totéž):

typ **[ ]** *jmenoPole*

typ jmenoPole **[ ]**

kde *typ* určuje datový typ položek pole (např. int nebo String).

Rozsah pole při deklaraci neuvádíme, deklarací pouze vytváříme proměnnou (odkaz) na pole. Několik příkladů deklarace pole následuje:

```
int [ ] prvocisla
String [ ] dnyvTydnu
String[] args
```

Poslední příklad je převzat z deklarace metody main – toto pole odkazuje na jednotlivé parametry z příkazové řádky při spuštění programu.

Pole alokujeme pomocí příkazu new takto:

```
jmenoPole = new typ [pocetPoložek];
prvocisla = new int [5];
dnyvTydnu = new String [7];
```

Obě části lze spojit dohromady, takže pokud chceme vytvořit pole prvních 5 prvočísel, zapíšeme to takto:

```
int prvocisla [] = new int [5];
```

Na jednotlivé prvky pole se pak odkazujeme indexy 0 až n –1, kde n je počet prvků pole. Na první položku námi definovaného pole se tedy odkážeme *prvocisla [0]* a na poslední *prvocisla [4]*. Pokud se pokusíme použít index 5 nebo jiný mimo interval 0..4 bude ohlášena chyba. Java striktně kontroluje překročení mezí pole.

Po vytvoření jsou v jednotlivých položkách pole jejich inicializační hodnoty, tj. např. pole celých čísel obsahuje ve všech položkách nuly.

Lze vytvořit i pole s již naplněnými hodnotami, pro pole dnyvTydnu by pak deklarace a inicializace vypadala takto:

```
String dnyvTydnu[] = {"pondělí", "úterý", "středa", "čtvrtek",
"pátek", "sobota","neděle"};
```

V tomto případě se pole neinicializuje pomoci příkazu new. Práce s tímto polem je naprosto stejná jako s polem vytvořeným pomocí **new** – prvky inicializovaného pole nejsou konstanty (tj. lze je měnit).

Každé pole má definovanou konstantu length, ve které je uložen počet prvků pole. Lze se na ni odkázat jménem pole, tečkou a jménem length. Tedy pro naše příklady:

```
System.out.println("Pole dnyvTydnu ma " +
       dnyvTydnu.length + " prvku");
for (int i = 0; i < dnyvTydnu.length; i++) {
   System.out.print(dnyvTydnu[i] + " ");
};
```

Vícerozměrná pole

V Javě lze vytvářet i vícerozměrná pole, deklarace vypadají takto:

```
int poleDvojrozmerne [] [];
```

Inicializovat toto pole lze několika způsoby. Na následujícím řádku

```
int poleDvojrozmerne [] [] = new int [2] [3];
```

vznikne pole se dvěma řádky a třemi sloupci, položky jsou naplněny nulami. Při inicializaci lze zadat počáteční hodnoty. Následuje příklad vytvoření pole se dvěma řádky a třemi sloupci, položky budou naplněny zadanými hodnotami:

```
int poleDvojrozmerne [ ] [ ] = {
   { 1, 2, 3, },
   { 4, 5, 6, },
}
```

V Javě je možná i postupná inicializace jednotlivých rozměrů pole:

```
int poleDvojrozmerne [ ] [ ] = new int [2] [ ];
```

Takto vznikne pole, u něhož ještě není určen počet sloupců. Toto umožňuje vytvářet poněkud nezvyklá pole, která budou mít v každém řádku jiný počet prvků[13].

```
poleDvojrozmerne [0] = new int [3];
poleDvojrozmerne [1] = new int [5];
```

Takto vzniklé pole má v prvním řádku tři a ve druhém pět prvků. Všechny prvky mají hodnotu nula. Pro zjištění počtu prvků lze opět použít proměnnou length. Počet řádků našeho dvojrozměrného pole zjistíme takto poleDvojrozmerne.length, počet prvku prvků prvního řádku poleDvojrozmerne [0].length a pro další řádky analogicky.

Lze vytvářet i pole tří a více rozměrná. Při deklaraci a inicializaci platí stená pravidla jako u dvojrozměrných. V případě, že pole vytváříme po částech, nesmíme přeskakovat rozměry.

[13] V Javě se vytvářejí pole polí, ne klasická dvourozměrná pole známá např. z Pascalu.

Lze tedy napsat

```
int poleTroj [] [] [] = new int [5] [5] [ ];
```

a poslední rozměr určit později, nelze však napsat

```
int poleTroj [] [] [] = new int [5] [ ] [5];
```

Pole je referenční typ, takže pokud pole použijeme jako parametr metody, je předáno odkazem.

Parametry vstupní řádky

Řekli jsme si, že pokud má třída poskytnout možnost spuštění programu, musí obsahovat metodu main definovanou jako public static void main (String [] args). Jako parametr metody je tedy použito jednorozměrné pole řetězců. Znamená to, že při spuštění programu můžeme za příkaz java a jméno třídy uvést libovolný počet parametrů oddělených mezerou, které budou uloženy do pole args. Pole args má samozřejmě také proměnnou length, která udává kolik parametrů uživatel zadal. Pokud nejsou zadány žádné parametry, má proměnná args.length hodnotu nula. V případě, že jeden parametr má být text s mezerami, je možné zadat ho v uvozovkách např. "Dobrý den".
Následující příklad ukazuje, jak z příkazové řádky načíst desetinné číslo představující poloměr kruhu a na konzoli pak vypsat jeho obvod a obsah. Parametr je načten jako String, je tedy nutný převod na číslo aby byl možný výpočet. Zatím neumíme ošetřit případ, že uživatel na příkazové řádce zadá chybný parametr (tj. např. java ObsahKruhu ahoj).

```
public class ObsahKruhu {

public static void main (String [] args) {
     if (args.length == 0) {
             //osetrime moznost ze uzivatel nezadal parametr
        System.out.println("Nebyl zadan parametr");
        System.exit (0);        //ukonci cely program
     };
     double polomer = Double.valueOf(args[0]).doubleValue();
      //prevedeme parametr ze Stringu na cislo typu double
     double obvod = 2*Math.PI*polomer;
                          //Math.PI je konstanta tridy Math
     double obsah = Math.PI*polomer*polomer;
     System.out.println("Kruh s polomerem "+polomer+" ma obsah "
                 +obsah+" a obvod "+ obvod);
}
}
```

Souhrnný příklad

Použití polí je ukázáno v následujícím programu, ve kterém se nadeklaruje pole o deseti prvcích typu int a naplní se náhodnými čísly z intervalu 0 až 100. Poté se pole vypíše, setřídí a znovu vypíše.

```
import java.util.*;
public class Pole {

      public static void main(String[] args){

         int poleCisel [] = new int [10];
         Random nahodneCislo = new Random ();
         for (int i = 0; i<10; i++){
           poleCisel[i] =Math.abs(nahodneCislo.nextInt() % 100);
         };
         System.out.println("Nesetridene pole");
         vypisPole(poleCisel);
         trideni(poleCisel);
         System.out.println("Setridene pole");
         vypisPole(poleCisel);
      }

      static void vypisPole (int pole []){
         for (int i = 0; i <pole.length; i++){
           System.out.print(" " + pole [i]);
         };
         System.out.println();
      }

      static void trideni (int pole []){
         boolean vymena = true;
         while (vymena){
           vymena = false;
           for (int j = 0; j< (pole.length -1); j++){
             if (pole[j] > pole [j+1]){
               int pomocna = pole[j];
               pole[j]= pole[j+1];
               pole[j+1] = pomocna;
               vymena = true;
             }
           }
         }
      }

}//konec tridy Pole
```