20 Dvojrozmerné polia. Dynamické polia.

20.1 Dvojrozmerné pole

Dvojrozmerné pole je pole, v ktorom je každý prvok poľom. Môžeme si ho predstaviť ako tabuľku so stĺpcami a riadkami.

Deklarujeme a vytvárame ho nasledovne:

```
typPola[][] menoPola = new typPola[pocetPrvkov][pocetPrvkov];
```

Prvý index predstavuje počet riadkov a druhý index počet stĺpcov. Všetky prvky sú však v pamäti uložené za sebou, preto niekedy programátori nahrádzajú dvojrozmerné pole jednorozmerným, musia si však „ustrážiť“ indexy.

Príklad:

```
int[][] dvojrozmernePole = new int[2][5];
```

0	0	0	0	0	0
1	0	0	0	0	0
	0	1	2	3	4

```
dvojrozmernePole[1][3] = 12;
```

0	0	0	0	0	0
1	0	0	0	12	0
	0	1	2	3	4

Pri práci s dvojrozmerným poľom používame cyklus v cykle – najčastejšie `for`. V nasledovnej aplikácii si ukážeme prácu s dvojrozmerným poľom.

20.2 Aplikácia DvojrozmernePole

Vytvoríme aplikáciu **DvojrozmernePole**. V programe deklarujeme a vytvoríme dvojrozmerné pole s rovnakým počtom riadkov a stĺpcov, napr. pole 8 x 8. Naplníme ho náhodnými hodnotami, dáme vypísať a nakoniec vypočítame súčet prvkov na hlavnej diagonále a súčet prvkov nad hlavnou diagonálou. Na všetky činnosti si vytvoríme vlastné metódy.

Java umožňuje vytvárať aj viacrozmerné polia, kde počet rozmerov určuje aj počet indexov poľa. Taktiež môžeme vytvárať aj nepravidelné polia, kde jednotlivé prvky sú polia s rôznym počtom prvkov.

20.3 Dynamické údajové štruktúry

Jednou z nevýhod pri práci s poľom je to, že akonáhle deklarujeme a vytvoríme pole s istým počtom prvkov, tento počet prvkov už počas behu programu meniť nemôžeme. Sú však situácie, kedy dopredu nevieme koľko prvkové pole budeme potrebovať.

Napríklad z databázovej tabuľky čítame hodnoty a niektoré vyhovujúce hodnoty chceme zapísať do poľa. Dopredu ale nevieme koľko hodnôt vyhovuje zadaným podmienkam, a tým pádom tiež nevieme koľko prvkové pole vlastne potrebujeme. Vytvoríme si napríklad 10000 prvkové pole a po prečítaní tabuľky zistíme, že v skutočnosti vyhovovalo 500 hodnôt – zbytočne sme v pamäti zabrali veľa miesta (pozor nemusí ísť o pole typu int, ale o pole rozsiahlejších objektov, ktoré môže zabrať polovicu pamäte RAM!). Alebo naopak, 10000 prvkové pole nám stačiť nemusí.

V takýchto prípadoch by nám vyhovovalo, keby sme mohli meniť počet prvkov počas behu programu. S poľami, s ktorými sme doteraz pracovali, to ale urobiť nevieme. V Jave však máme k dispozícii tzv. **dynamické údajové štruktúry** nazývané tiež aj **kontajnery**. Tieto dynamické údajové štruktúry môžu počas behu programu meniť svoju veľkosť a navyše poskytujú niektoré pokročilé operácie s uloženými údajmi.

Dynamické údajové štruktúry sú vlastne objekty a preto s nimi pracujeme objektovo. Jednotlivé triedy dynamických údajových štruktúr sú súčasťou balíka java.util.

20.4 Dynamické polia – dynamická štruktúra ArrayList

Jednou z dynamických údajových štruktúr je aj štruktúra, ktorá vyhovuje naším požiadavkám a nazýva sa **ArrayList**. Predstavuje vlastne dynamické pole objektov, v ktorom môžeme meniť počet prvkov počas behu programu. Zvykne sa nazývať aj **kolekcia**.

V balíku java.util máme k dispozícii triedu ArrayList, pomocou ktorej vytvoríme dynamické pole objektov.

Všeobecný tvar deklarácie a vytvorenia poľa

```
ArrayList<typObjektu> menoPola;              //deklarácia dynamického poľa
menoPola = new ArrayList<typObjektu>(); //vytvorenie poľa objektov
```

`ArrayList` – názov triedy

`typObjektu` – typ poľa; akýkoľvek typ objektu, aký možno v Jave vytvoriť. Pozor! Nie je možné použiť primitívne údajové typy

`menoPola` – identifikátor poľa; názvy tvoríme rovnako ako u primitívnych premenných

`new` – operátor, ktorého úlohou je nájsť vo voľnom úložisku blok pamäte a vrátiť adresu

`()` – môžu byť prázdne, alebo môžeme uviesť inicializačný počet prázdnych prvkov

Deklarovať a vytvoriť možno dynamické pole aj na jednom riadku. Bezprostredne po vytvorení je dynamické pole prázdne. Pre prácu s dynamickým poľom máme k dispozícii množstvo metód, pomocou ktorých môžeme do poľa vkladať hodnoty, vypisovať, prechádzať celým poľom, usporadúvať hodnoty v poli atď.

20.5 Základné metódy pre prácu s dynamickými poľami

Metóda add(Object o) – vloženie objektu do poľa

Metóda vloží objekt na koniec poľa.

```
ArrayList<Integer> poleCelychCisel= new ArrayList<Integer>();
poleCelychCisel.add(100);        //pole bude obsahovať jeden prvok
poleCelychCisel.add(200);        //pole bude obsahovať dva prvky
poleCelychCisel.add(300);        //pole bude obsahovať tri prvky
poleCelychCisel.add(400);        //pole bude obsahovať štyri prvky
```

Metóda add(int index,Object o) – vloženie objektu do poľa na určenú pozíciu

Metóda vloží objekt na pozíciu určenú indexom a ostatné prvky posunie

```
poleCelychCisel.add(2,1000);     //pole bude obsahovať päť prvkov s hodnotami
                                 //100 200 1000 300 400
```

Metóda get(int index) – vrátenie hodnoty prvku

Metóda vráti hodnotu prvku, ktorého pozícia je zadaná indexom

```
poleCelychCisel.get(3);          //vráti hodnotu 300
```

Metóda remove(int index) – odstránenie prvku zo zadanou pozíciou

Metóda odstráni prvok z poľa, ktorého pozícia je zadaná indexom

```
poleCelychCisel.remove(1);       //odstráni prvok s indexom 1; v poli zostanú štyri prvky
```

Metóda set(int index, Object o) – zmena hodnoty prvku zo zadanou pozíciou

Metóda zmení hodnotu prvku, ktorého pozícia je zadaná indexom

```
poleCelychCisel.set(2,900);      //prvok s indexom 2 bude mať teraz hodnotu 900
```

Metóda size() – počet prvkov poľa

Metóda vráti počet prvkov poľa

```
poleCelychCisel.size();          //vráti hodnotu 4
```

Metóda contains(Object o) – zistenie výskytu objektu v poli

Metóda zistí, či sa v poli nachádza prvok s uvedenou hodnotou

```
poleCelychCisel.contains(100);        //vráti true
poleCelychCisel.contains(155);        //vráti false
```

Metóda clear() – odstránenie všetkých prvkov z poľa

Metóda odstráni všetky prvky z poľa

```
poleCelychCisel.clear();
poleCelychCisel.size();               //vráti 0
```

Trieda ArrayList obsahuje ďalšie užitočné metódy. Aj v tomto prípade je vhodné siahnuť po dokumentácii, ktorá je k dispozícii na stránke:

http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

20.6 Aplikácia DynamickePole

Vytvoríme aplikáciu **DynamickePole**. V nej si precvičíme metódy uvedené v kapitole 20.5, ukážeme si použitie cyklov pri práci s dynamickým poľom a tiež vytvorenie a volanie vlastných metód.

20.7 Cvičenie

1. Vytvorte aplikáciu s názvom **MaxPodDiagonalou**. Program zistí a vypíše najväčšiu hodnotu pod hlavnou diagonálou v dvojrozmernom poli N x N celých čísel.
2. Vytvorte aplikáciu s názvom **PoleNahodnychHodnot**. Program vygeneruje do dynamického poľa celočíselné náhodné hodnoty z intervalu <a,b>. Generovanie bude prebiehať dovtedy, kým sa nevygeneruje hodnota **P**. Hodnota P je zadaná z klávesnice a je z intervalu <a,b>. Program potom nech vypíše celé pole a počet prvkov.
3. Vytvorte aplikáciu s názvom **DeliteleCisla.** Program do dynamického poľa zapíše všetky celočíselné delitele celého čísla **P**, zadaného z klávesnice.
4. Vytvorte aplikáciu s názvom **DynamickePoleRetazcov**. Z klávesnice budeme postupne zadávať ľubovoľné reťazce tak, že vždy zadáme dlhší reťazec ako predchádzajúci. Tie sa budú zapisovať do dynamického poľa. V momente, keď zadáme reťazec s rovnakou alebo menšou dĺžkou ako predchádzajúci reťazec, tak zadávanie reťazcov skončí – posledný reťazec sa už do poľa nezapíše. Potom nech program vypíše celé pole a aj počet reťazcov v ňom.

20.8 Otázky

1. Charakterizujte dvojrozmerné pole.
2. Uveďte deklaráciu a vytvorenie dvojrozmerného poľa.
3. Čo sú to dynamické údajové štruktúry?
4. Charakterizujte dynamické pole a uveďte názov triedy, ktorá takéto pole umožňuje vytvoriť.
5. Uveďte deklaráciu a vytvorenie dynamického poľa.
6. Uveďte aspoň tri metódy pre prácu s dynamickým poľom.