Масиви в мові JAVA

Головне, про що треба пам'ятати при використанні масиву : ідентифікатор масиву – це змінна ***reference-типу***. Тому створення масиву включає 2 кроки:

- ✔ визначення змінної-ідентифікатору масиву (***декларація*** масиву) та
- ✔ виділення пам'яті розміру, достатнього для збереження вказаної кількості елементів масиву (***створення*** масиву).

В результаті ідентифікатор масиву виявляється посиланням на цю область пам'яті.

Синтаксис визначення масиву наступний:

```
<тип_елементів> [] <ідент_масиву>;
<ідент_масиву> = new <тип_елементів> [кіл_елементів];
```

Перший рядок – це ***декларація*** (опис) масиву. Другий рядок – ***створення*** масиву.

Або (одним рядком):

```
<тип_елементів> [] <ідент_масиву> =
                  new <тип_елементів> [кіл_елементів];
```

Приклади:

```
int[] Arr;                  // декларація масиву
Arr = new int[10];          // створення масиву з 10 елементів
або
int[] Arr = new int[10];//декларація і створення масиву
```

Приклади:

```
int[] Arr1;          // декларація масиву
Arr1 = new int[10]; // створення масиву з 10
                    // елементів
або
int[] Arr1 = new int[10]; //декларація і
                          // створення масиву
Arr1 = new int[20];  // створення іншого масиву
                     //з тим самим ідентифікатором
int[] Arr2;      // декларація ще одного масиву
Arr2 = new int[30]; // його створення
      Arr1.length ? Arr2.length
Arr1 = Arr2; // тепер перший масив той самий,
             // що другий
      Arr1.length ? Arr2.length
```

Звертання до елементу масиву відбувається через його ідентифікатор та індекс елементу. Важливо пам'ятати, що в мові JAVA елементи масиву індексуються від 0. Таким чином, перший елемент масиву завжди має індекс 0, а останній – індекс, рівний **кіл_елементів-1**. При звертанні до елементу з неіснуючим індексом генерується переривання Exception.

Приклади:

```
int[] Arr = new int[10];
for (int i = 0; i < 10; i++)
System.out.println("Arr [“ + i +”] = “ + Arr[i]);
```

або (що більш правильно, оскільки кожний масив сам знає свій розмір через властивість **Length**):

```
for (int i = 0; i < Arr.Length; i++)
System.out.println("Arr [“ + i +”] = “ + Arr[i]);
```

Ініціалізація масивів

Масив може бути проініціалізований в момент створення деякими значеннями (автоматично елементи новоствореного масиву ініціалізуються нульовими значеннями), наприклад, таким чином:

```
int[] iArray = {1, 2, 3, 4, 5};
```

Зверніть увагу – операція створення масиву **new** тут пропущена, адже, якщо вказані початкові значення елементів, масив створюється компілятором автоматично, а розмір масиву визначається по кількості заданих початкових значень.

Можна визначити і проініціалізувати масив, явно задавши його розмір (хоча це й надмірна для компілятора інформація), проте в цьому випадку службове слово **new** є необхідним. Крім того, вказаний розмір масиву має співпадати з кількістю значень ініціалізації:

```
double[] dArray = new double[3];
```

Масиви в мові Java (продовження)

Мова JAVA дозволяє використання масивів розмірності, більшої за одиницю. Для вивчення багатовимірних масивів обмежимось розмірністю 2.

Синтаксис визначення двовимірного масиву наступний:

```
<тип_елементів> [][] <ідент_масиву>;
<ідент_масиву> = new <тип_елементів> [кіл_1][кіл_2];
```

У першому рядку – двовимірний масив ***декларується*** (описується), у другому рядку – ***створюється***. (У декларації масиву кома в квадратних дужках позиціонує дві розмірності масиву, кількість перша і друга вказує відповідно на кількість елементів у рядку та у стовпчику, якщо розглядати двовимірний масив як аналог матриці).

Або (одним рядком):

```
<тип_елементів> [][] <ідент_масиву> =
                    new <тип_елементів> [кіл_1][кіл_2];
```

Приклади:

```
int[][] Arr;          // декларація масиву
Arr = new int[3][4];      // створення масиву, здатного
          // вмістити матрицю з 3 рядків та 4 стовпчиків
або
int[][] Arr = new int[3][4];//декларація і створення масиву
```

Звертання до елементу двовимірного масиву відбувається через його ідентифікатор та ***два індекси*** елементу. Очевидно, що останній елемент матиме індекс, рівний **кіл_1-1, кіл_2-1**.

Приклади:

```
int[][] Arr = new int[3][4];
for (int i = 0; i < 3; i++)
    for (int j = 0; j < 4; j++)
 System.out.println("Arr = " + Arr[i][j]);
```

або, використовуючи стандартні методи класу **Array** для визначення кількості елементів по кожній вимірності масиву:

```
for (int i = 0; i < Arr.length(0) ; i++)
    for (int j = 0; j < Arr.length(1) ; j++)
 System.out.println("Arr = " + Arr[i][j]);
```

Ініціалізація двовимірних масивів

Двовимірний масив також можна ініціалізувати, наприклад, таким чином:

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

або з використанням операції **new** :

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

Таким чином створений масив розмірностей 2*3 та проініціалізований: його перший рядок значеннями 1, 2, 3, а другий – значеннями 4, 5, 6.

```
//  так теж можна
int[][] iArray = {
                 {1, 2, 3},
                 {4, 5, 6} };
```

Цикл **foreach** виявляється особливо зручним для перебору всіх елементів багатовимірних масивів, адже відпадає необхідність використовувати вкладені цикли.

Приклад:

```
int[][] iArray ={
                  {1, 2, 3},
                  {4, 5, 6} };
//виводимо кожний елемент масиву Arr
```

Проте у такий спосіб важко забезпечити “красивий вивід” масиву – всі елементи масиву виводитимуться просто з нового рядка.

Зверніть увагу цикл **foreach** перебирає елементи у такому порядку, що найшвидше зростає останній індекс – саме так послідовно розташовані у пам'яті елементи двовимірного масиву (якщо мати за аналог матрицю – то по рядках).

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

for (int[] i : arr) {
    for (int y : i) {
        System.out.print(y + " ");
    }
    System.out.println();
}
```

Лабораторна №3

1. Знайти найбільше і найменше число та його координати в матриці чисел.

{4, 5, 6, 7}
{8, 9, 8, 7}
{7, 5, 4 ,3}

2. Підрахувати кількість нулів в матриці, що складається з двійкових чисел.

{0, 1, 1, 0}
{1, 0, 1, 0}
{1, 1, 0 ,1}

3. Підрахувати кількість елементів, що дорівнюють заданому числу(з консолі, нариклад 7).

{4, 5, 6, 7}
{8, 9, 8, 7}
{7, 5, 4 ,3}

4. Створити матрицю, значення елементів якої дорівнюють сумі індексів цих елементів.