Introduction to Programming

Rania Baghernejad

Java – part 7

Arrays

مباحث

آرایه‌ها (Arrays)

آرایه یکی از ابتدایی‌ترین و در عین حال کاربردی‌ترین ساختارهای داده‌ای در زبان جاوا است. زمانی که بخواهیم چند مقدار از یک نوع خاص (مثلاً چند عدد صحیح) را به صورت منظم و در کنار هم ذخیره کنیم، از آرایه استفاده می‌کنیم.

تعریف آرایه و مقداردهی

در جاوا، برای تعریف یک آرایه ابتدا نوع داده‌ی آن را مشخص می‌کنیم و سپس از علامت [] استفاده می‌کنیم. پس از آن نام آرایه و تعداد خانه‌هایی که می‌خواهیم برای آن رزرو کنیم را مشخص می‌کنیم.

تعریف و مقداردهی جداگانه:

```
int[] numbers = new int[5];
numbers[0] = 10;
numbers[1] = 20;
numbers[2] = 30;
numbers[3] = 40;
numbers[4] = 50;
```

در این مثال:

- آرایه‌ای از ۵ عدد صحیح تعریف شده است.
- اندیس‌ها (index) از 0 تا 4 هستند.
- مقدار هر خانه با استفاده از numbers[index] = value; تنظیم شده است.

تعریف و مقداردهی هم‌زمان:

```
int[] numbers = {10, 20, 30, 40, 50};
```

در این روش، بدون نیاز به نوشتن new، مقادیر به صورت مستقیم در هنگام تعریف وارد می‌شوند.

دسترسی به اعضای آرایه

برای دسترسی به یک خانه‌ی خاص در آرایه، از شماره‌ی اندیس آن استفاده می‌کنیم. شماره‌ی اندیس همیشه از صفر شروع می‌شود.

```
System.out.println(numbers[0]);
System.out.println(numbers[4]);
```

- خط اول مقدار اولین خانه را چاپ می‌کند . (۱۰)
- خط دوم مقدار پنجمین خانه را چاپ می‌کند. (۵۰)

تغییر مقدار اعضای آرایه

برای تغییر مقدار یک خانه از آرایه، کافی‌ست به آن اندیس دسترسی پیدا کرده و مقدار جدید را تخصیص دهید:

```
numbers[2] = 99;
```

در اینجا، مقدار خانه‌ی سوم که قبلاً ۳۰ بود، اکنون ۹۹ خواهد شد.

طول آرایه(Array Length)

برای این که بدانیم یک آرایه چند عنصر دارد، می‌توانیم از ویژگی length. استفاده کنیم:

```
System.out.println(numbers.length);
```

این دستور عدد ۵ را برمی‌گرداند زیرا آرایه شامل ۵ عنصر است.

دقت کنید که برای آرایه‌ها از length. بدون پرانتز استفاده می‌شود.

پیمایش آرایه با حلقه

برای این که به تمام عناصر آرایه دسترسی داشته باشیم، از حلقه‌ها استفاده می‌کنیم. دو روش رایج عبارت‌اند از:

با استفاده از حلقه for معمولی:

```
for (int i = 0; i < numbers.length; i++) {
    System.out.println(numbers[i]);
}
```

در این روش:

- متغیر i از صفر شروع می‌شود.
- در هر دور از حلقه، مقدار یکی از عناصر آرایه چاپ می‌شود.
- حلقه تا زمانی که i کمتر از طول آرایه باشد ادامه می‌یابد.

با استفاده از حلقه enhanced for:

```
for (int num : numbers) {
    System.out.println(num);
}
```

در این روش:

- متغیر num در هر تکرار یکی از عناصر آرایه را در خود نگه می‌دارد.
- این روش ساده‌تر و خواناتر است، اما اگر نیاز به اندیس‌ها داشته باشید، بهتر است از روش اول استفاده کنید.

خطاهای رایج

اگر تلاش کنید به خانه‌ای خارج از محدوده‌ی آرایه دسترسی پیدا کنید، برنامه دچار خطا خواهد شد:

```
System.out.println(numbers[5]);
```

در این حالت، چون اندیس 5 وجود ندارد (آخرین اندیس 4 است)، برنامه خطای **ArrayIndexOutOfBoundsException** می‌دهد.

محدودیت‌های آرایه‌ها

- آرایه‌ها دارای طول ثابت هستند. یعنی پس از تعریف، نمی‌توان تعداد عناصر آن را تغییر داد.
- برای ذخیره داده‌هایی با طول متغیر، بهتر است از کلاس‌هایی مانند ArrayList استفاده شود (در فصل‌های بعد بررسی خواهد شد).
- همه‌ی عناصر آرایه باید از یک نوع داده‌ی مشخص باشند. مثلاً آرایه‌ی int نمی‌تواند شامل مقدار String باشد.

آرایه‌های چندبعدی در جاوا (Multidimensional Arrays)

چرا آرایه دوبعدی؟

در زندگی واقعی، ما اغلب با داده‌هایی سروکار داریم که به صورت **جدول** یا **ماتریس** ذخیره می‌شوند. برای مثال:

Student	Math	Physics	Chemistry
John	85	90	78
Alice	92	88	95
Mark	75	80	72

این ساختار مانند یک ماتریس ۳×۳ است که هر **سطر (row)** نمایانگر یک دانش‌آموز و هر **ستون (column)** نمایانگر یک درس است.

برای پیاده‌سازی چنین ساختاری در جاوا، از آرایه‌های دوبعدی استفاده می‌کنیم.

نحوه تعریف یک آرایه دوبعدی

```
int[][] matrix = new int[3][4];
```

- این آرایه مثل یک جدول ۳ ردیفه و ۴ ستونه است.
- در اصطلاح ریاضی، به آن یک **ماتریس ۴×۳ (۳ سطر و ۴ ستون)** گفته می‌شود.
- عدد اول (3) تعداد سطرها، و عدد دوم (4) تعداد ستون‌هاست.

تصویر ذهنی:

[0][0] [0][1] [0][2] [0][3]

[1][0] [1][1] [1][2] [1][3]

[2][0] [2][1] [2][2] [2][3]

مقداردهی اولیه(Initialization)

الف. مقداردهی خانه‌به‌خانه:

```
matrix[0][0] = 1;
matrix[0][1] = 2;
matrix[1][0] = 3;
matrix[2][3] = 10;
```

ب. مقداردهی در زمان تعریف:

```
int[][] matrix = {
    {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9, 10, 11, 12}
};
```

این همانند نوشتن ماتریس زیر در ریاضیات است:

1 2 3 4

5 6 7 8

9 10 11 12

هر {} نشان‌دهنده یک سطر است.

دسترسی به عناصر ماتریس

دسترسی به یک عنصر در موقعیت خاص، با استفاده از اندیس سطر و ستون انجام می‌شود:

```
System.out.println(matrix[1][2]); // prints 7
```

این یعنی: **از ردیف دوم، ستون سوم** را بخوان (یادآوری: شمارش از صفر آغاز می‌شود).

پیمایش کامل ماتریس

برای خواندن یا نوشتن تمام عناصر ماتریس، معمولاً از دو حلقه تو در تو استفاده می‌کنیم.

الف. استفاده از حلقه for معمولی:

```
for (int i = 0; i < matrix.length; i++) {
    for (int j = 0; j < matrix[i].length; j++) {
        System.out.print(matrix[i][j] + "\t");
    }
    System.out.println();
}
```

- i نمایانگر شماره سطر (row index) است.
- jنمایانگر شماره ستون (column index) است.
- \t برای ایجاد فاصله‌های جدولی استفاده شده است.

ب. استفاده از حلقه enhanced for:

```
for (int[] row : matrix) {
    for (int value : row) {
        System.out.print(value + "\t");
    }
    System.out.println();
}
```

در این روش:

- row یک آرایه‌ی یک‌بعدی است که هر بار به یکی از سطرهای ماتریس اشاره می‌کند.
- value مقدار هر عنصر در آن سطر است.

طول سطر و ستون

- matrix.length → تعداد سطرها (rows)
- matrix[0].length → تعداد ستون‌ها (columns) در سطر اول

مثلاً در ماتریس بالا:

```
System.out.println(matrix.length);        // 3
System.out.println(matrix[0].length);     // 4
```

آرایه نامتقارن (Jagged Array)

جاوا به شما اجازه می‌دهد آرایه‌هایی بسازید که سطرهایشان تعداد ستون یکسانی نداشته باشند:

```
int[][] jagged = new int[3][];
jagged[0] = new int[2];
jagged[1] = new int[4];
jagged[2] = new int[1];
```

در اینجا:

- سطر اول ۲ عنصر دارد.
- سطر دوم ۴ عنصر دارد.
- سطر سوم تنها ۱ عنصر دارد.

برای درک بهتر، می‌توانید آن را مانند جدولی فرض کنید که بعضی خانه‌های آن خالی‌اند یا مقادیرش کمتر است.

به عنوان مثال:

1 2 # #

5 6 7 8

9 # # #

مثال عملی: جمع عناصر ماتریس

```
int sum = 0;
for (int i = 0; i < matrix.length; i++) {
    for (int j = 0; j < matrix[i].length; j++) {
        sum += matrix[i][j];
    }
}
System.out.println("Sum = " + sum);
```

این کد تمام عناصر ماتریس را جمع می‌کند.

خطاهای رایج

اگر به اندیسی خارج از محدوده دسترسی پیدا کنید، مانند:

```
System.out.println(matrix[3][0]);
```

خطای زیر ایجاد می‌شود:

```
java.lang.ArrayIndexOutOfBoundsException
```

زیرا سطر با اندیس ۳ در آرایه‌ای با فقط ۳ سطر (اندیس‌های ۰ تا ۲) وجود ندارد.

آرایه دوبعدی به عنوان ورودی کاربران

بعداً با استفاده از کلاس Scanner می‌توان آرایه‌ای دوبعدی را از ورودی کاربران گرفت، برای مثال نمرات دانش‌آموزان یا عناصر یک ماتریس.

گرفتن آرایه دوبعدی (ماتریس) از کاربر

فرض کنیم یک ماتریس ۲×۳ می‌خواهیم:

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

System.out.println("Enter elements of a 2x3 matrix:");

for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 3; j++) {
        System.out.print("Element at [" + i + "][" + j + "]: ");
        matrix[i][j] = input.nextInt();
    }
}
```