Introduction to Programming

Rania Baghernejad

Java – part 8

Methods and Functions

مباحث

تعریف ۲

ساختار کلی متد ۲

return یعنی چه؟ ۴

Overloading یا متدهای همنام با ورودی‌های متفاوت ۴

متدها(Methods)

تعریف

در جاوا، متد شبیه به یک "ماشین حساب" است. ما ورودی‌هایی به آن می‌دهیم، و آن متد کار خاصی را انجام می‌دهد و گاهی نتیجه‌ای به ما برمی‌گرداند.

در ریاضیات داشتیم که هرگاه می‌نویسیم:

$f(x) = x + 2$

یعنی اگر به این تابع (function) یک x بدهیم، به ما $x + 2$ را برمی‌گرداند.

در برنامه‌نویسی هم متدها دقیقاً همین نقش را دارند. مثلاً اگر متدی به نام add(x, y) داشته باشیم، انتظار داریم دو عدد x و y را بگیرد و جمع آن‌ها را بدهد. (توجه داشته باشید که متدها تنها محدود به برگرداندن اعداد نمی‌شوند.)

چرا از متد استفاده می‌کنیم؟

اگر در برنامه‌ای بخواهیم یک کار را چند بار انجام دهیم (مثلاً چاپ سلام، جمع دو عدد، محاسبه میانگین)، بهتر است آن را فقط یک‌بار بنویسیم و هر بار که نیاز بود، صدا بزنیم. این باعث می‌شود:

- از تکرار کد جلوگیری کنیم.
- برنامه مرتب و قابل‌فهم‌تر باشد.
- نگهداری کد ساده‌تر شود.

ساختار کلی متد

```
modifier returnType methodName(parameters) {
    // code commands in the method
}
```

اجزای این ساختار:

- modifier مشخص می‌کند این متد از کجا قابل‌دسترسی است (مثل public یعنی همه‌جا قابل استفاده است.)
- returnType نوع داده‌ای که متد برمی‌گرداند (مثل int, String, double یا void که یعنی هیچ چیزی برنمی‌گرداند.)

- methodName نام متد.
- parameters چیزهایی که متد برای کار کردن لازم دارد، مانند x و y در f(x, y).

مثال ۱: متد بدون ورودی و بدون خروجی

```
public static void sayHello() {
    System.out.println("Hello!");
}
```

این متد فقط عبارت "Hello!" را چاپ می‌کند. نیازی به ورودی ندارد (پس داخل پرانتز را خالی می‌گذاریم.)، و چیزی هم برنمی‌گرداند (پس از void استفاده می‌کنیم.)

مثال ۲: متد با ورودی

```
public static void greet(String name) {
    System.out.println("Hello, " + name + "!");
}
```

این متد به یک ورودی از نوع رشته name نیاز دارد، و بر اساس آن پیغام خوشامدگویی چاپ می‌کند. (در این جا باز خروجی نداریم پس void قرار می‌دهیم.)

یک سوال رایج: «چرا وقتی خروجی چاپ می‌کنیم باز هم نوع تابع را void قرار می‌دهیم؟»

زمانی که یک تابع یا متد تعریف می‌شود، نوع بازگشتی آن نشان می‌دهد که این تابع قرار است چه نوع داده‌ای را به محل فراخوانی خود بازگرداند. اگر تابعی هیچ داده‌ای را بازنگرداند و فقط عملی را انجام دهد (مانند چاپ روی صفحه)، نوع بازگشتی آن void خواهد بود.

چاپ کردن با استفاده از System.out.println() به این معنا نیست که تابع دارای خروجی از نظر برنامه‌نویسی است. این دستور صرفاً یک پیام را روی خروجی استاندارد (معمولاً صفحه نمایش) چاپ می‌کند و **هیچ داده‌ای به برنامه یا تابع فراخوان برگردانده نمی‌شود**. بنابراین این عمل یک اثر جانبی (side effect) محسوب می‌شود، نه یک مقدار بازگشتی.

به‌طور خلاصه، خروجی‌ای که توسط System.out.println() مشاهده می‌شود، به معنای وجود خروجی برای تابع نیست. این فقط یک نمایش برای کاربر است. اگر تابع داده‌ای را برای استفاده در بخش دیگری از برنامه برنگرداند، از نظر زبان برنامه‌نویسی، آن تابع دارای بازگشت نیست و باید با void تعریف شود.

پس تفاوت اصلی میان "چاپ کردن" و "بازگرداندن مقدار" در این است که چاپ کردن برای کاربر است و اثری در منطق اجرای برنامه ندارد، اما بازگرداندن مقدار برای خود برنامه اهمیت دارد و امکان استفاده از آن در بخش‌های دیگر برنامه وجود دارد.

مثال ۳: متد با ورودی و خروجی

```
public static int add(int a, int b) {
    return a + b;
}
```

متد بالا دو عدد a و b را می‌گیرد و مقدار جمع آن‌ها را برمی‌گرداند.

در main می‌توان آن را این‌گونه صدا زد:

```
int result = add(3, 5);
System.out.println(result); // خروجی: 8
```

---

return یعنی چه؟

کلمه‌ی return یعنی نتیجه‌ای را از متد به جایی که آن متد صدا زده شده، برگردانیم. مثل زمانی که می‌گوییم:

$f(x) = x + 2$

وقتی x را بدهیم، نتیجه $x + 2$ به ما برگردانده می‌شود.

مثال ۴: متد برای چاپ آرایه

فرض کنیم لیستی از اعداد داریم و می‌خواهیم همه آن‌ها را چاپ کنیم.

```
public static void printArray(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        System.out.println(arr[i]);
    }
}
```

Overloading یا متدهای همنام با ورودی‌های متفاوت

جاوا اجازه می‌دهد چند متد با نام یکسان ولی ورودی متفاوت داشته باشیم:

```
public static int multiply(int a, int b) {
    return a * b;
}
```

```java
public static double multiply(double a, double b) {
    return a * b;
}
```

وقتی یکی از این متدها صدا زده شود، جاوا بر اساس نوع داده‌ها خودش انتخاب می‌کند کدام را اجرا کند.

مثال:

برنامه‌ای بنویس که برای یک کافی‌شاپ، قیمت نهایی سفارش مشتری را چاپ کند. مشتری ممکن است:

- فقط یک نوشیدنی سفارش دهد،
- یا نوشیدنی همراه با یک خوراکی،
- یا نوشیدنی همراه با خوراکی و تعداد مهمان‌ها برای محاسبه‌ی کل سفارش.

ما چند متد با همین نام خواهیم داشت که بسته به تعداد و نوع ورودی‌ها، رفتار متفاوتی نشان دهند.

پاسخ:

```java
public class CafeOrder {

    public static void main(String[] args) {
        printOrder("Coffee");
        printOrder("Coffee", "Croissant");
        printOrder("Coffee", "Croissant", 3);
    }

    public static void printOrder(String drink) {
        System.out.println("You ordered: " + drink);
        System.out.println("Total: $" + getPrice(drink));
    }

    public static void printOrder(String drink, String snack) {
        System.out.println("You ordered: " + drink + " and " + snack);
        System.out.println("Total: $" + (getPrice(drink) + getPrice(snack)));
    }

    public static void printOrder(String drink, String snack, int people) {
        double totalPerPerson = getPrice(drink) + getPrice(snack);
        System.out.println("Order for " + people + " people: " + drink + " and "
+ snack + " each.");
        System.out.println("Total: $" + (totalPerPerson * people));
    }
```

```
    public static double getPrice(String item) {
        if (item.equals("Coffee")) {
            return 3.5;
        } else if (item.equals("Croissant")) {
            return 2.0;
        } else {
            return 0.0;
        }
    }
}
```

توضیح:

- در این برنامه، متدی به نام printOrder سه بار تعریف شده، اما هر کدام پارامترهای متفاوتی دارند. به این مفهوم در جاوا بارگذاری متد (Method Overloading) می‌گویند.
- در زمان اجرا، بسته به تعداد آرگومان‌هایی که به متد می‌دهیم، نسخه‌ی مناسب از متد انتخاب و اجرا می‌شود.
- متد کمکی getPrice قیمت نوشیدنی یا خوراکی را برمی‌گرداند. چون قیمت را به برنامه برمی‌گرداند (نه چاپ)، نوع بازگشتی آن doubleاست.
- همه‌ی printOrderها از نوع void هستند چون وظیفه‌شان فقط چاپ اطلاعات سفارش برای کاربر است، نه بازگرداندن مقدار برای پردازش.