Introduction to Programming

Rania Baghernejad

Java – part 5

Conditional Statements

مباحث

دستورات شرطی(Conditional Statements)

در برنامه‌نویسی، اغلب نیاز داریم بر اساس شرایط مختلف تصمیم‌گیری کنیم. برای این منظور، از دستورات شرطی استفاده می‌کنیم تا بخش‌هایی از کد فقط در صورت برقرار بودن یک شرط اجرا شوند.

فلوچارت (Flowchart)

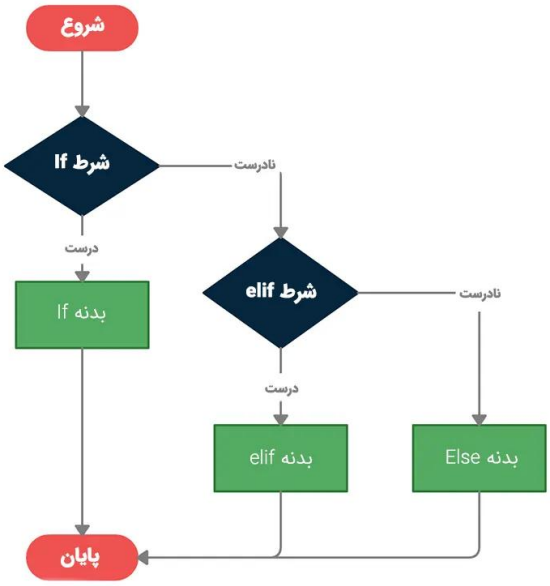
یک نوع نمودار است که برای نمایش روند یا جریان اجرای یک فرایند یا برنامه استفاده می‌شود. این نمودار معمولاً از انواع مختلف اشکال هندسی مانند مستطیل، لوزی و دایره تشکیل شده است که هر کدام نقش خاصی دارند. در فلوچارت‌ها، خطوط یا پیکان‌ها برای نشان دادن جهت جریان فرایند از یک مرحله به مرحله بعدی استفاده می‌شود.

اجزای اصلی فلوچارت‌ها:

- **دایره (Start/End):** این شکل برای نشان دادن نقطه شروع یا پایان فرایند استفاده می‌شود.
- **مستطیل (Process):** نشان‌دهنده یک مرحله از فرایند است که معمولاً شامل یک عمل یا دستور است.
- **لوزی (Decision):** برای نشان دادن یک تصمیم‌گیری استفاده می‌شود. در این بخش معمولاً شرطی بررسی می‌شود که مسیر بعدی بسته به نتیجه آن انتخاب می‌شود (مثلاً بله یا خیر).
- **خطوط یا پیکان‌ها:** این‌ها برای نشان دادن جهت جریان از یک مرحله به مرحله بعدی به کار می‌روند.

فلوچارت‌ها به ویژه در برنامه‌نویسی برای طراحی الگوریتم‌ها و نمایش گام به گام روند اجرای برنامه بسیار مفید هستند. این نمودارها به برنامه‌نویس‌ها و دیگر افراد کمک می‌کنند تا فرآیندهای پیچیده را به شکل ساده‌تری درک کنند و اشتباهات را سریع‌تر شناسایی کنند.

فلوچارت دستورات شرطی:



دستور if

زمانی استفاده می‌شود که می‌خواهیم یک بخش از کد فقط در صورتی اجرا شود که یک شرط خاص برقرار باشد.

ابتدا کلمه if را نوشته و شرط مد نظر را داخل پرانتز جلوی آن می‌نویسیم. سپس با گذاشتن دستور کد مد نظر داخل { } این دستور شرطی را به اتمام می‌رسانیم.

نحوه عملکرد:

- اگر مقدار شرط true (درست) باشد، کد داخل { } اجرا می‌شود.
- اگر مقدار شرط false (نادرست) باشد، کد داخل { } نادیده گرفته می‌شود.

ساده‌ترین نوع تصمیم‌گیری در جاوا:

```
if (condition) {
    // code to be executed if condition is true
}
```

مثال:

```
int age = 20;

if (age >= 18) {
    System.out.println("You are an adult.");
}
```

در این مثال، اگر مقدار age بزرگ‌تر یا مساوی ۱۸ باشد، پیام چاپ خواهد شد.

دستور if-else

در صورتی که بخواهیم تصمیمات متناوبی بگیریم (یعنی اگر یک شرط برقرار نباشد، گزینه دیگری را امتحان کنیم)، از دستور if-else استفاده می‌کنیم. این دستور به ما این امکان را می‌دهد که برای هر شرایطی که True نباشد، کد دیگری اجرا شود.

نحوه عملکرد:

- اگر شرط True باشد، کد داخل بخش if اجرا می‌شود.
- اگر شرط False باشد، کد داخل بخش else اجرا می‌شود.

برای اجرای یک بخش از کد در صورت نادرست بودن شرط:

```
if (condition) {
    // code if condition is true
} else {
    // code if condition is false
}
```

مثال:

```
int number = 3;

if (number % 2 == 0) {
    System.out.println("Even number");
} else {
    System.out.println("Odd number");
}
```

دستور if-else if-else

گاهی اوقات نیاز داریم که چند شرط مختلف را بررسی کنیم و برای هرکدام از آن‌ها یک رفتار خاص تعریف کنیم. برای اینکار از دستور else if استفاده می‌کنیم. این دستور به ما اجازه می‌دهد که بیش از دو گزینه را بررسی کنیم.

- اگر شرط اول Trueباشد، کد داخل اولین بخش if اجرا می‌شود.
- اگر شرط اول False باشد، شرط بعدی بررسی می‌شود. اگر True باشد، کد داخل بخش else if اجرا می‌شود.
- اگر هیچ‌کدام از شرایط برقرار نباشند، کد داخل بخش else اجرا می‌شود.

برای بررسی چندین شرط مختلف به ترتیب:

```
if (condition1) {
    // code if condition1 is true
} else if (condition2) {
    // code if condition2 is true
} else {
    // code if none of the above conditions is true
}
```

مثال:

```
int score = 85;

if (score >= 90) {
    System.out.println("Grade: A");
```

```
} else if (score >= 80) {
    System.out.println("Grade: B");
} else {
    System.out.println("Grade: C or lower");
}
```

شرط‌های تو در تو (Nested if)

در جاوا، می‌توانیم از دستور if در داخل دستور if دیگری استفاده کنیم. این کار را شرط تو در تو (Nested if) می‌نامند. این روش زمانی مفید است که بخواهیم چند شرط پیچیده را در داخل یکدیگر بررسی کنیم.

نحوه عملکرد:

- ابتدا شرط اول بررسی می‌شود. اگر برقرار باشد، شرط داخلی نیز بررسی می‌شود.
- اگر شرط داخلی برقرار باشد، کد داخل آن اجرا می‌شود.
- اگر شرط داخلی برقرار نباشد، کد داخل بخش else اجرا می‌شود.
- اگر شرط اول برقرار نباشد، کد داخل بخش else اجرا می‌شود.

```
public class Welcome1 {
    public static void main(String[] args) {
        int number = 15;

        if (number > 0) {
            if (number > 10) {
                System.out.println("number is larger than 10.");
            } else {
                System.out.println("number is less or equal to 10.");
            }
        } else {
            System.out.println("number is less than 0");
        }
    }
}
```

نکات مهم:

- شرط‌ها باید داخل پرانتز نوشته شوند.
- بلاک‌های کد با {}مشخص می‌شوند. اگر فقط یک خط کد دارید، می‌توان بدون آکولاد هم نوشت، اما استفاده از آکولاد همیشه توصیه می‌شود.
- شرایط معمولاً با عملگرهای مقایسه‌ای و منطقی ساخته می‌شوند.

دستور switch

وقتی بخواهیم مقدار یک متغیر را با **چند مقدار خاص** مقایسه کنیم، به جای استفاده از چندین if-else if پشت‌سرهم، می‌توانیم از switch استفاده کنیم که ساختار تمیزتر و خواناتری دارد.

ساختار کلی switch

```
switch (expression) {
    case value1:
        // code block
        break;
    case value2:
        // code block
        break;
    ...
    default:
        // code block
}
```

توضیحات اجزاء:

- expression: عبارتی که بررسی می‌شود (می‌تواند int, char, String, enumباشد.)
- case: هر مقدار ممکن که با expression مقایسه می‌شود.
- break: باعث می‌شود پس از اجرای یک case، از switch خارج شویم.
- default: در صورتی اجرا می‌شود که هیچ‌یک از caseها مطابقت نداشته باشد (اختیاری ولی توصیه‌شده).

مثال:

```
int day = 3;

switch (day) {
    case 1:
        System.out.println("Monday");
        break;
    case 2:
        System.out.println("Tuesday");
        break;
    case 3:
        System.out.println("Wednesday");
        break;
    default:
        System.out.println("Invalid day");
}
```

در این مثال، چون مقدار day برابر با ۳ است، خروجی خواهد بود:

```
Wednesday
```

نکات مهم:

۱. اگر breakرا ننویسید، جاوا اجرای همه‌یcase های بعدی را ادامه می‌دهد (رفتاری به نام **fall-through**)

۲. switch فقط می‌تواند از انواع خاصی استفاده کند — (int, char, String, enum) از boolean یا float پشتیبانی نمی‌کند.

۳. استفاده از default باعث می‌شود برنامه در صورت وقوع مقادیر غیرمنتظره هم واکنش درستی نشان دهد.

عدم استفاده از break در دستور switch

اگر در دستور switchپس از هر caseاز دستور breakاستفاده نکنید، اجرای کد از همان نقطه‌ای که caseمطابقت می‌کند، ادامه پیدا خواهد کرد و به صورت **اتوماتیک** به موارد caseبعدی نیز می‌رود. این رفتار به نام **fall-through**شناخته می‌شود.

مثال بدون استفاده از break:

```
int day = 3;

switch (day) {
    case 1:
        System.out.println("Monday");
    case 2:
        System.out.println("Tuesday");
    case 3:
        System.out.println("Wednesday");
    default:
        System.out.println("Invalid day");
}
```

خروجی:

```
Wednesday
Invalid day
```

چرا این اتفاق می‌افتد؟

در اینجا، مقدار day برابر با ۳ است، بنابراین ابتدا دستور case 3 اجرا می‌شود و "Wednesday" چاپ می‌شود. از آنجا که هیچ break وجود ندارد، اجرای کد به default می‌رود و "Invalid day" نیز چاپ می‌شود. به همین دلیل، هیچ break به معنای **اجرا شدن همه‌ی کدهایی است که بعد از case جاری قرار دارند.**

نکات مهم در مورد fall-through:

- این رفتار می‌تواند مفید باشد، مثلاً زمانی که بخواهید چند case مختلف را با یک دستور مشترک اجرا کنید.
- اگر هدف شما جلوگیری از این رفتار است، باید حتماً از break بعد از هر case استفاده کنید.

مثال از fall-through بدون break

```
int number = 1;

switch (number) {
    case 1:
    case 2:
    case 3:
        System.out.println("Number is between 1 and 3");
        break;
    default:
```

```
        System.out.println("Number is outside the range");
}
```

خروجی:

```
Number is between 1 and 3
```

در اینجا، چون هیچ break در بین caseها وجود ندارد، تمامی caseها به هم متصل شده و در صورتی که numberبرابر با ۱، ۲ یا ۳ باشد، همان پیغام چاپ می‌شود.