Introduction to Programming

Rania Baghernejad

Java – part 6

Loops

مباحث

حلقه while ۲
حلقه شمارنده‌ای(Counter-controlled loop) ۳
حلقه نگهبان‌دار یا متوقف‌شونده (Sentinel-controlled loop) ۳
حلقه‌های تو در تو (Nested while loop) ٤
حلقه do-while ٤
حلقه for ٥
حلقه تو در تو با (Nested for loop) for ٥
دستورات break و continue ٥
دستور break ٥
دستور continue ٦
فلوچارت حلقه‌ها ۸

حلقه‌ها (Loops)

در برنامه‌نویسی، گاهی لازم است مجموعه‌ای از دستورات را چندین بار اجرا کنیم. برای انجام این کار بدون تکرار کدها، از حلقه‌ها استفاده می‌شود. حلقه‌ها باعث می‌شوند برنامه‌ها خواناتر، کارآمدتر و قابل توسعه‌تر باشند.

جاوا چند نوع حلقه مختلف دارد، اما سه ساختار اصلی عبارتند از:

۱. while
۲. do-while
۳. for

ما ابتدا با حلقه‌ی while شروع می‌کنیم و سپس به do-while و در نهایت به for می‌پردازیم.

حلقه while

حلقه‌ی while در برنامه‌نویسی زمانی استفاده می‌شود که بخواهیم یک دستور تا زمانی که یک شرط برقرار است، تکرار شود.

ابتدا کلمه while را نوشته و شرط مد نظر را داخل پرانتز جلوی آن می‌نویسیم. سپس با گذاشتن دستور کد مد نظر داخل { } به حلقه پایان می‌دهیم.

نحوه عملکرد:

- شرط بررسی می‌شود: اگر مقدار آن True باشد، بدنه‌ی حلقه اجرا می‌شود.
- تغییر مقدار متغیر کنترلی: باید مقدار متغیر کنترل‌کننده تغییر کند تا از تکرار بی‌نهایت جلوگیری شود.
- خروج از حلقه: زمانی که شرط False شود، اجرای حلقه متوقف خواهد شد.

ساختار حلقه while به این شکل است:

```
while (condition) {
    // statements to repeat
}
```

- condition باید یک مقدار بولی باشد (یعنی true یا false)
- تا زمانی که شرط برقرار باشد (true باشد)، دستورات داخل بلوک اجرا می‌شوند.
- اگر از ابتدا شرط برقرار نباشد، ممکن است کد داخل حلقه حتی یک بار هم اجرا نشود.
- حلقه‌ی **while** به دو صورت قابل پیاده‌سازی است:

- تکرار کنترل‌شده با شمارنده (**Counter-controlled loop**)
- تکرار کنترل‌شده با سنتینل (**Sentinel-controlled loop**)

تفاوت اصلی:

در حلقه‌ی شمارنده‌ای، **تعداد تکرار از قبل مشخص** است.

در حلقه‌ی سنتینلی، **تعداد تکرار مشخص نیست** و اجرای حلقه تا رسیدن به مقدار مشخصی (سنتینل) ادامه دارد.

حلقه شمارنده‌ای(Counter-controlled loop)

در این روش، تعداد اجرای حلقه از قبل معلوم است. متغیری به عنوان شمارنده مقداردهی اولیه می‌شود، مقدار آن در هر تکرار تغییر می‌کند و حلقه زمانی متوقف می‌شود که شمارنده به مقدار موردنظر برسد.

مثال: چاپ اعداد ۱ تا ۵

```
int i = 1;
while (i <= 5) {
    System.out.println("Count: " + i);
    i++;
}
```

حلقه نگهبان‌دار یا متوقف‌شونده (Sentinel-controlled loop)

در این نوع حلقه، شرط حلقه بر اساس یک "مقدار خاص (sentinel value) " (به این مقدار خاص flag نیز گفته می‌شود) تعیین می‌شود که نشان‌دهنده پایان ورود داده‌ها یا تکرارهاست.

مثال: گرفتن ورودی از کاربر تا وقتی عدد منفی وارد شود

```
Scanner input = new Scanner(System.in);
int number = 0;

while (number >= 0) {
    System.out.print("Enter a number (negative to stop): ");
    number = input.nextInt();
}
```

در اینجا مقدار sentinel همان عدد منفی است .

حلقه‌های تو در تو (Nested while loop)

می‌توان یک حلقه while را درون یک حلقه دیگر قرار داد. این برای مواردی مثل جدول ضرب مفید است.

مثال: چاپ یک مستطیل ۳×۴ از ستاره‌ها

```
int row = 1;
while (row <= 3) {
    int col = 1;
    while (col <= 4) {
        System.out.print("* ");
        col++;
    }
    System.out.println();
    row++;
}
```

حلقه do-while

حلقه do-while شبیه while است، با این تفاوت که شرط آن پس از اجرای بدنه حلقه بررسی می‌شود. بنابراین، حداقل یک بار اجرا خواهد شد، حتی اگر شرط از ابتدا برقرار نباشد.

ساختار:

```
do {
    // statements
} while (condition);
```

مثال: گرفتن ورودی تا وقتی عدد مثبت است

```
Scanner input = new Scanner(System.in);
int number;

do {
    System.out.print("Enter a number (0 to stop): ");
    number = input.nextInt();
} while (number != 0);
```

حلقه for

حلقه for همانند حلقه while نوع شمارنده‌ای برای اجرای تعداد مشخصی از تکرارها بسیار مناسب است. ساختار آن سه بخش دارد: مقداردهی اولیه، شرط تکرار، و افزایش/کاهش مقدار.

ساختار:

```
for (initialization; condition; update) {
    // statements
}
```

مثال: چاپ اعداد ۱ تا ۵ با for

```
for (int i = 1; i <= 5; i++) {
    System.out.println("i = " + i);
}
```

حلقه تو در تو با for (Nested for loop)

مثال: جدول ضرب ۱ تا ۳

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 3; j++) {
        System.out.print(i * j + "\t");
    }
    System.out.println();
}
```

دستورات break و continue

دستور break

دستور break باعث خروج کامل از حلقه می‌شود، حتی اگر شرط حلقه هنوز برقرار باشد. معمولاً وقتی استفاده می‌شود که شرط خاصی در داخل حلقه برقرار شود و دیگر نیازی به ادامه‌ی اجرای حلقه نباشد.

مثال:

```
for (int i = 1; i <= 10; i++) {
    if (i == 5) {
        break;
    }
```

```
    System.out.println(i);
}
```

خروجی:

```
1
2
3
4
```

توضیح: وقتی مقدار i برابر ۵ می‌شود، دستور break اجرا شده و حلقه فوراً متوقف می‌شود. بنابراین اعداد ۵ به بعد چاپ نمی‌شوند.

دستور continue

برخلاف break، دستور continue باعث رد شدن از ادامه اجرای حلقه در همان تکرار فعلی می‌شود و مستقیماً به تکرار بعدی می‌رود، بدون آن‌که حلقه را متوقف کند.

مثال:

```
for (int i = 1; i <= 5; i++) {
    if (i == 3) {
        continue;
    }
    System.out.println(i);
}
```

خروجی:

```
1
2
4
5
```

توضیح: وقتی i == 3 باشد، continue باعث می‌شود دستور System.out.println(i); اجرا نشود و حلقه مستقیماً به i == 4 برود.

مقایسه کلی:

دستور	عملکرد

break	اجرای حلقه را کاملاً متوقف می‌کند.
continue	اجرای آن دور از حلقه را متوقف کرده و به تکرار بعدی می‌رود.

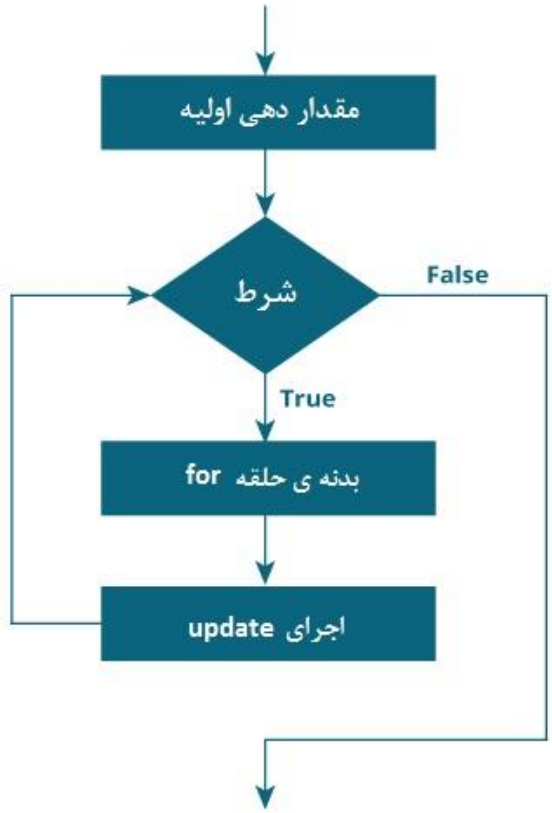
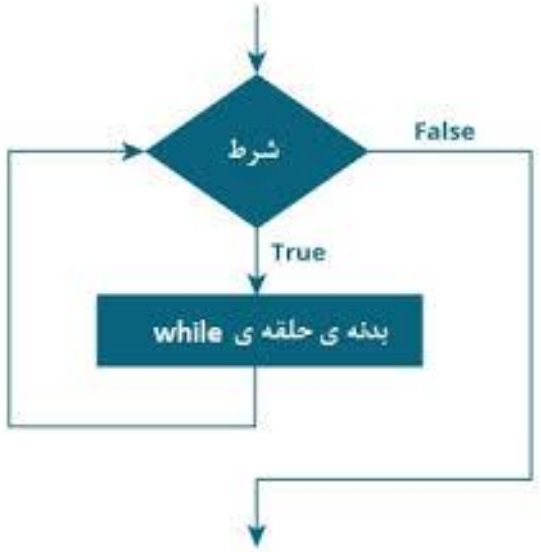
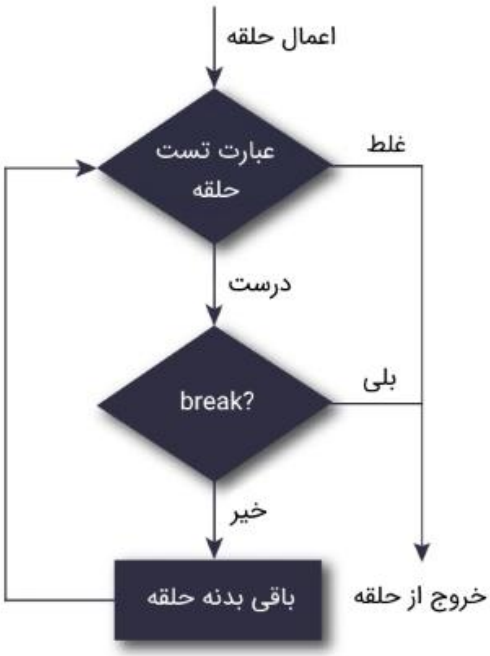
یکی از استفاده‌های رایج از ساختار while (true) در برنامه‌نویسی، ساختن **حلقه‌های ورودی پیوسته** است که کاربر باید چیزی وارد کند تا حلقه متوقف شود. در این حالت، معمولاً از ترکیب while (true) همراه با break برای خروج استفاده می‌شود.

مثال از while (true) همراه با break و continue

برنامه‌ای که از کاربر عدد می‌گیرد و جمع آن‌ها را محاسبه می‌کند، تا وقتی که کاربر عدد منفی وارد کند:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int sum = 0;

        while (true) {
            System.out.print("Enter a number (negative number to stop): ");
            int num = scanner.nextInt();

            if (num < 0) {
                break;
            }

            if (num == 0) {
                System.out.println("Zero is ignored.");
                continue;
            }

            sum += num;
        }

        System.out.println("Total sum: " + sum);
    }
}
```

توضیح:

- while (true) حلقه‌ای است که بی‌نهایت ادامه پیدا می‌کند، مگر اینکه با break متوقف شود.
- اگر عدد منفی وارد شود break اجرا می‌شود و حلقه خاتمه می‌یابد.
- اگر عدد صفر وارد شود با continue، آن دور از حلقه رد می‌شود و جمع انجام نمی‌شود.
- بقیه‌ی اعداد مثبت به جمع اضافه می‌شوند.

فلوچارت حلقه‌ها