حلقه for در جاوا

حلقه در برنامه نویسی برای تکرار یک بلوک خاص از کد استفاده می شود. در این آموزش ، می توانید یک حلقه for در برنامه نویسی جاوا ایجاد کنید. حلقه در برنامه نویسی برای تکرار یک بلوک خاص از کد تا زمانی که شرایط خاصی برآورده شود استفاده می شود (شرط نادرست شود).

حلقه ها همان چیزی هستند که کامپیوترها را تبدیل به ماشین های جالب می کنند. تصور کنید که باید ۵۰ بار یک جمله را روی صفحه چاپ کنید. خوب ، می توانید این کار را با استفاده از دستور چاپ ۵۰ بار (بدون استفاده از حلقه) انجام دهید. چگونه می خواهید یک میلیون بار یک جمله را چاپ کنید؟ پس باید از حلقه ها استفاده کنید.

این فقط یک مثال ساده است. شما یاد خواهید گرفت که برای نوشتن برخی از برنامه های جالب در این آموزش از حلقه استفاده کنید.

حلقهfor

ساختار حلقه در جاوا به شکل زیر است:

```
for (initialization; testExpression; update)
{
// codes inside for loop’s body
}
```

حلقه for چگونه کار می کند؟

۱ Initialization -یا مقدار دهی اولیه فقط یک بار اجرا می شود.

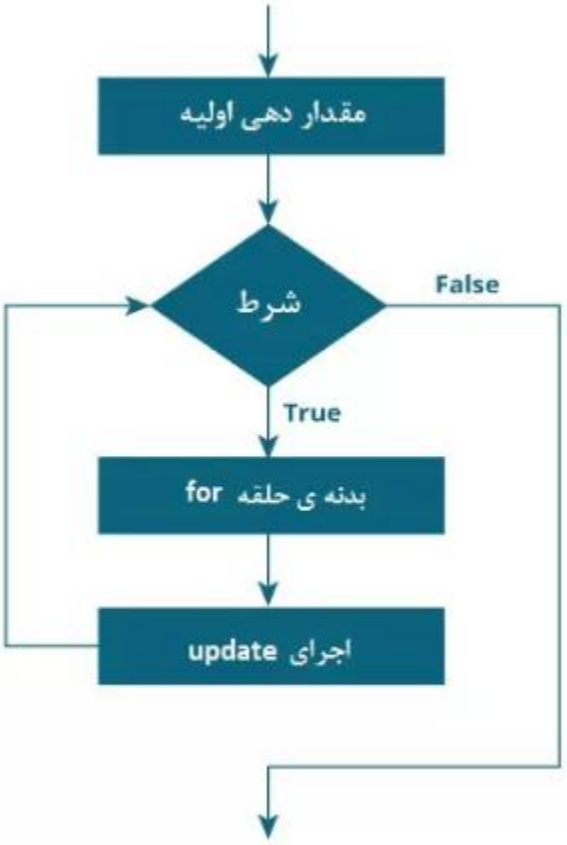
۲ -سپس شرط (در اینجا testExpression) ارزیابی می شود که یک عبارت boolean است.

۳ -اگر شرط به صورت صحیح ارزیابی شود ،

- کد های داخل بدنه حلقه اجرا می شوند.
- سپس عبارت update اجرا می شود.
- باز هم ، شرط ارزیابی می شود.
- اگر شرط صحیح باشد ، کد های داخل بدنه ی حلقه اجرا می شوند و عبارت update اجرا می شود.
- این روند تا زمانی که شرط به غلط ارزیابی شود ادامه می یابد.

۴ -اگر شرط به غلط ارزیابی شود ، حلقه for پایان می یابد.

فلوچارت حلقهfor



مثال ۱ : حلقهfor

```
1.  // Program to print a sentence 10 times
2.  class Loop {
3.  public static void main(String[] args) {
4.
5.  for (int i = 1; i <= 10; ++i) {
6.  System.out.println("Line " + i);
7.  }
8.  }
9.  }
```

خروجی

```
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
```

در اینجا ، متغیر i تعریف شده و ابتدا به آن مقدار ۱ داده می شود.

سپس شرط i <= 10 ارزیابی می شود. از آنجا که ، درست است ، بدنه حلقه اجرا می شود که Line 1را روی صفحه چاپ می کند.

سپس عبارت i++ اجرا می شود. اکنون مقدار i به ۲ افزایش یافته است. بازهم شرط i <= 10 ارزیابی می شود که صحیح است و بدنه حلقه اجرا می شود که Line 2 را روی صفحه چاپ می کند.

این روند تکرار تا i= 11 ادامه دارد. وقتی i برابر ۱۱ شد ، شرط i <= 10 نادرست است و حلقه خاتمه می یابد.

مثال ۲ : حلقهfor

```
1.  // Program to find the sum of natural numbers from 1 to 1000.
2.  class Number {
3.  public static void main(String[] args) {
4.
5.  int sum = 0;
6.
7.  for (int i = 1; i <= 1000; ++i) {
8.  sum += i; // sum = sum + i
9.  }
10.
11. System.out.println("Sum = " + sum);
12. }
13. }
```

خروجی

```
Sum = 500500
```

در اینجا ، مقدار متغیر sum از ۰ شروع می شود. سپس ، در هر بار تکرار حلقه ، متغیر sum برابر با sum+i می شود و مقدار i تا زمانی که بیشتر از ۱۰۰۰ شود افزایش می یابد.

```
۱st iteration: sum = 0+1 = 1
۲nd iteration: sum = 1+2 = 3
۳rd iteration: sum = 3+3 = 6
۴th iteration: sum = 6+4 = 10
... .. ...
۹۹۹th iteration: sum = 498501 + 999 = 499500
۱۰۰۰th iteration: sum = 499500 + 1000 = 500500
```

حلقه for بی نهایت

اگر شرط همواره درست باشد ، حلقه برای همیشه اجرا خواهد شد و به آن حلقه بی نهایت گفته می شود. مثلا:

```
1.  // Infinite for Loop
2.  class Infinite {
3.  public static void main(String[] args) {
4.
5.  int sum = 0;
6.  for (int i = 1; i <= 10; –i) {
7.  System.out.println("Hello");
8.  }
9.  }
10. }
```

در اینجا ، شرط i <= 10 هرگز نادرست نیست و Hello بارها چاپ می شود (حداقل در تئوری).

مقدار دهی اولیه متغیر ، به روزرسانی متغیر و شرط مورد استفاده در حلقه for اختیاری است. در اینجا نمونه دیگری از حلقه نامتناهی آورده شده است:

```
1.  for ( ; ; ) {
2.  }
```

حلقه for ... each در جاوا

در جاوا ، شکل دیگری برای حلقه (علاوه بر حلقه استاندارد for) برای کار با آرایه ها و مجموعه ها وجود دارد.

اگر در حال کار با آرایه ها و مجموعه ها هستید ، می توانید از ساختار دیگر حلقه for (فرم پیشرفته ی حلقه for) برای تکرار آیتم های آن ها استفاده کنید. این نوع حلقه for-each نامیده می شود زیرا حلقه از طریق هر عنصر آرایه / مجموعه تکرار می شود.

در اینجا مثالی برای تکرار عناصر یک آرایه با استفاده از حلقه استاندارد for آورده شده است:

```
1.  class ForLoop {
2.  public static void main(String[] args) {
3.
4.  char[] vowels = {'a', 'e', 'i', 'o', 'u'};
5.  for (int i = 0; i < vowels.length; ++ i) {
6.  System.out.println(vowels[i]);
7.  }
8.  }
9.  }
```

می توانید کد بالا را با استفاده از حلقه for-each هم بنویسید:

```
1.  class AssignmentOperator {
2.  public static void main(String[] args) {
3.
4.  char[] vowels = {'a', 'e', 'i', 'o', 'u'};
5.  // foreach loop
6.  for (char item: vowels) {
7.  System.out.println(item);
8.  }
9.  }
10. }
```

خروجی هر دو کد شبیه و برابر است با:

```
a
e
i
o
u
```

استفاده از حلقه for پیشرفته برای نوشتن آسان تر است و باعث می شود کد خوانا تر باشد. از این رو معمولا بیش از فرم استاندارد توصیه می شود.

ساختار حلقه یfor-each

در ابتدا به ساختار حلقه for-each نگاه کنیم:

```
for(data_type item : collection) {
...
}
```

در ساختار بالا ،

- collectionیک مجموعه یا آرایه ای است که قصد نوشتن حلقه بر روی آن را دارید.
- itemیک عنصر واحد از collection است.

حلقه ی for-each چگونه کار می کند؟

در اینجا چگونگی عملکرد حلقه for-each آورده شده است.

- تکرار از طریق هر عنصر در آرایه یا مجموعه داده شده (collection) ،
- هر مورد را در یک متغیر (item) ذخیره می کند.
- و بدنه ی حلقه را اجرا می کند.

مثال: حلقه یfor-each

- برنامه زیر مجموع تمام عناصر یک آرایه اعداد صحیح را محاسبه می کند.

```
class EnhancedForLoop {
public static void main(String[] args) {

```

```
4.  int[] numbers = {3, 4, 5, -5, 0, 12};
5.  int sum = 0;
6.
7.  for (int number: numbers) {
8.  sum += number;
9.  }
10.
11. System.out.println("Sum = " + sum);
12. }
13. }
```

خروجی

```
Sum = 19
```

- در برنامه بالا ، اجرای حلقه for-each به شرح زیر است:

Iteration	Value of number	Value of sum
1	3	3
2	4	7
3	5	12
4	-5	7
5	0	7
6	12	19

- تکرار حلقه for-each را مشاهده می کنید

- همه ی عناصر numbers تکرار می شوند.
- هر عنصر در متغیر number ذخیره می شود.
- بدنه حلقه اجرا می شود ، یعنی number به sum اضافه می شود.

تمرین

۱-برنامه ای بنویسید که ۲ عدد را دریافت کند و اولی را به توان دومی برساند و در خروجی چاپ کند

۲- برنامه ای بنویسید که یک عدد را دریافت کندو فاکتوریل آن را محاسبه و چاپ کند

فاکتوریل عدد n یعنی $n! = 1*2*3*4*5*6*....*n$

۳- برنامه ای بنویسید که جمله nام فیبوناچی را محاسبه کرده و در خروجی چاپ کند.

جمله فیبوناچی: ۱ ۱ ۲ ۳ ۵ ۸ ۱۳ ۲۱ .....

۴- برنامه ای بنویسید که یک عدد را از ورودی دریافت کند و حاصل عبارت زیر را محاسبه و چاپ کند

$$S=(2^1/1!)+(2^2/2!)+(2^3/3!)+\ldots+(2^n/n!)$$