آموزش زبان برنامه نویسی جاوا

حلقه های کنترلی

جلسه یازدهم

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!!

ممکن است گاهی برای نوشتن برنامه ای نیاز داشته باشیم یک بلوک کد چندین بار اجرا شود!!!!!

یا این که بخواهید یک یا چندین دستور را با توجه به یک شرط ، هر بار بصورت تکرار پی در پی اجرا کنید!!!!

با مثال هایی از دنیای واقعی اطراف خود شروع میکنیم:

- تا هنگامی که شیر آب باز است: آب جاری است.
- تا هنگامی که شیر گاز باز است: اجاق گاز روشن است.
- تا هنگامی که پا روی پدال گاز است:ماشین حرکت میکند.
- و.....

در این مثال ها یک شرط وجود دارد و تا زمانی که شرط ما درست (true) باشد نتیجه شرط اجرا می شود. مثلا تا هنگامی که ما پدال گاز اتومبیل را فشار می دهیم اتومبیل حرکت میکنم (یا حرکت اتومبیل که یک دستور تکراری است پی در پی اجرا می شود) اگر پا رو پدال برداریم اتومبیل حرکتش متوقف می شود.

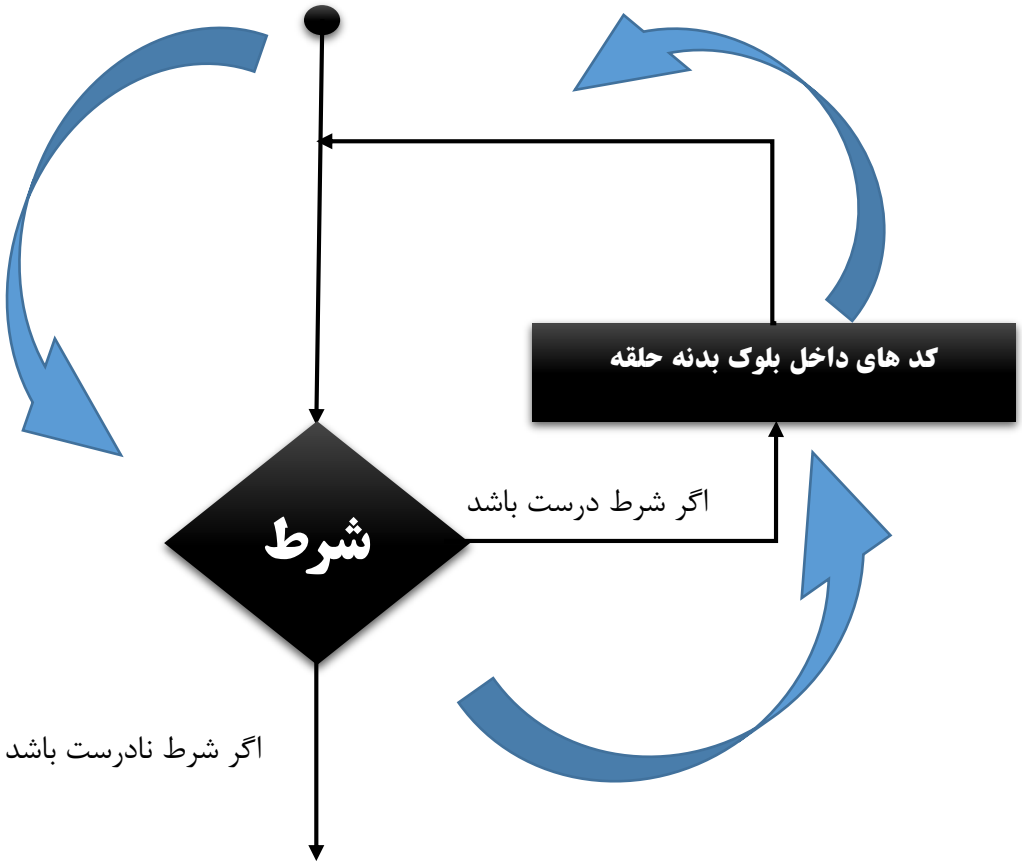
پس در این مثال ها در صورت درست بودن شرط وارد یک حلقه می شویم و یک یا چند دستور درون حلقه را بصورت تکرار پی در پی اجرا میکنیم.

Tip! در برنامه نویسی هم حلقه ها شرط دارند که تا هنگامی که این شرط درست می باشد یک یا چندین دستور را پی در پی تکرار میکند و در صورت نادرست شدن شرط از حلقه خارج می شود در مثال پدال گاز همین که ما پا رو پدال میگذاریم شرط درست (true) شده وارد حلقه تکرار می شویم و ماشین حرکت میکند و همین که پا رو پدال برداریم شرط نادرست (false) شده و از حلقه تکرار خارج می شویم و ماشین متوقف می شود.

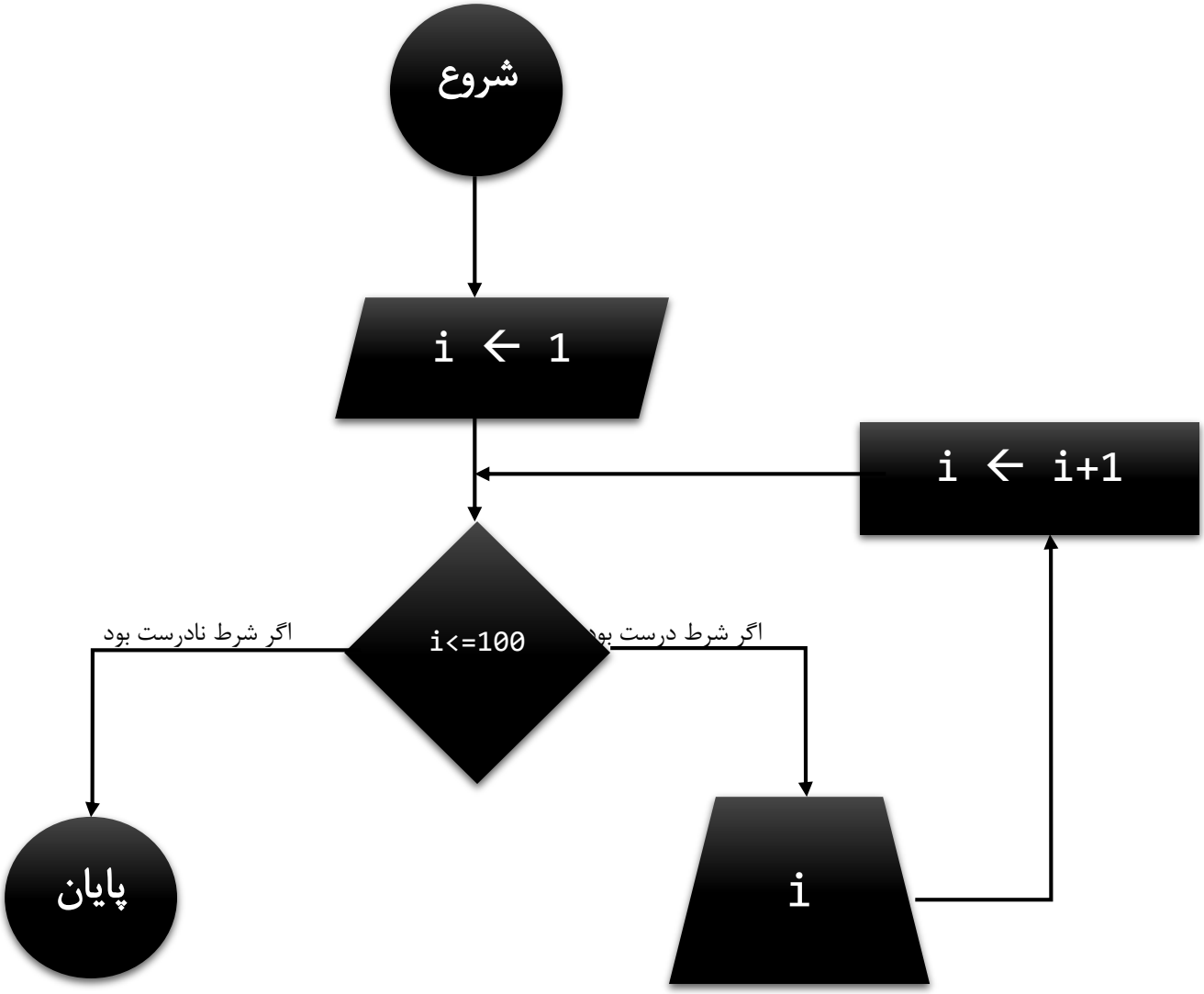
قبل از این که به سراغ حلقه ها در برنامه نویسی برویم یک نگاه فلوچارتی به ان می اندازیم :

برای ساختن حلقه ها از ترکیب یک شرط و یک سری دستورالعمل استفاده میکنیم.

فلوچارت کلی یک حلقه بصورت زیر است:



فرض کنید قصد داریم اعداد ۱ تا ۱۰۰ را چاپ کنیم فلوچارت این برنامه بصورت زیر می باشد:



- برنامه شروع میشه، مقدار ۱ به متغیر i نسبت داده می شود بعد شرط تا هنگامی که i کوچکتر یا مساوی یا ۱۰۰ هست مقدار i را چاپ کن یکی به i اضاف کن و بریز داخل i و باز شرط را برسی کن اگر درست بود ادامه بده اگر نه برو به سمت پایان.
- حالا که دید و درک بهتری از حلقه ها پیدا کردیم به سراغ حلقه ها در برنامه نویسی جاوا می رویم ☺

در زبان برنامه نویسی جاوا انواع حلقه ها برای رفع نیاز ما در برنامه نویسی فراهم شده است که جزییات آن را بررسی میکنیم.

حلقه while

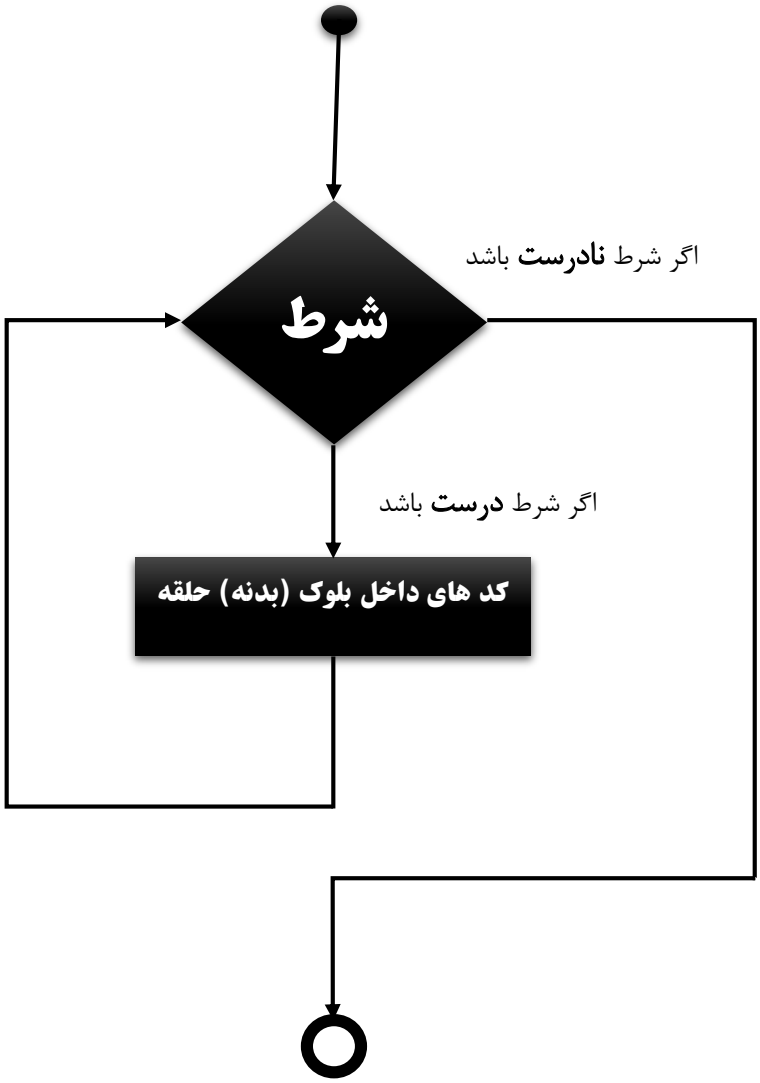
توصیف:

- تکرار یک یا گروهی از دستورات تا هنگامی که شرط درست (true) باشد.
- ابتدا شرط را بررسی میکند بعد اجرای دستورات بدنه حلقه.
- **while** تمامی حروفش از حرف انگیسی کوچک می باشد

شکل پیاده سازی حلقه while :

```
while(Boolean_expression)
{
   //Statements
}
```

- Statements ممکن است شامل یک دستور که اجباری نیست درون دو بلوک {} قرار گیرد یا شامل چندین دستور که لازم است این دستورات درون دو بلوک {} قرار بگیرند.
- شرطی که درون پرانتز قرار می گیرد می تواند شامل هر عبارتی (expression) باشد.
- اگر نتیجه Boolean_expression درست (true) بود وارد حلقه می شویم و تا هنگامی که Boolean_expression درست (true) باشد اجرای دستورات داخل حلقه اجرا می شود.
- وقتی شرط نادرست(false) می شود، بلافاصله از حلقه خارج می شویم و کنترل برنامه به خط بعد از حلقه عبور می کند و دستورات خط بعد از حلقه را اجرا می شود.
- نکته کلیدی که در مورد حلقه **while** وجود دارد این است که احتمال داره هرگز این حلقه اجرا نشود (وارد کدهای درون بلوک (محدوده) حلقه نشویم. چه موقع؟!!! وقتی که نتیجه عبارت شرطی درون پرانتز روبروی while نادرست (false) شود در این مواقع کامپایلر از اجرای حلقه while و دستورات درون آن صرف نظر کرده و از روی آن می پرد به سمت خط بعد از حلقه while و دستورات خط بعد از این حلقه را اجرا میکند.
- در زیر فلوچارت مربوط به حلقه **while** را مشاهده می کنید:


شرط
اگر شرط نادرست باشد
اگر شرط درست باشد
کد های داخل بلوک (بدنه) حلقه

در برنامه زیر اعداد ۱ تا ۱۰ چاپ می شود.

```
package iran;

public class Eleventh_Session {

        public static void main(String args[]) {
            int i = 1;

            while( i <= 10 ) {
               System.out.print("value of i : " + i );
               i++;
               System.out.print("\n");
            }
         }
}
```

خروجی:

```
value of i : 1
value of i : 2
value of i : 3
value of i : 4
value of i : 5
value of i : 6
value of i : 7
value of i : 8
value of i : 9
value of i : 10
```

```
package iran;
```

- پکیج، فولدری که کلاس ما درونش می باشد.

```
public class Eleventh_Session {
```

- کلاس ما که با حرف بزرگ شروع شده که ابتدا کلمه کلیدی **public** بعد کلمه کلیدی **class** بعد **نام کلاس** که باید با حرف بزرگ شروع شود و در نهایت بلوک کلاس } که باز شده.

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

- متد main که کاربرد آن برای اجرای دستورات ما می باشد.

```
int i = 1;
```

- تعریف متغیر از نوع عدد صحیح int و مقدار دهی اولیه مستقیم به آن.

```
while( i <= 10 ) {
```

- حلقه **while** که عبارت شرطی درون پرانتز آن " i <= 10 " می باشد.
- در صورتی وارد کد های داخل بلوک حلقه می شوید که نتیجه شرط " i <= 10 " درست باشد و تا هنگامی دستورات درون حلقه **while** اجرا می شود که شرط مذکور برقرار و درست باشد.

```
while( i <= 10 ) {
            System.out.print("value of i : " + i );
            i++;
            System.out.print("\n");
         }
```

- ابتدا شرط درون حلقه بررسی می شود اگر درست است دستورات درون بدنه حلقه اجرا می شود.
- این دستورات ابتدا چاپ مقدار i و سپس یکی به مقدار متغیر i اضاف می کند و می ریزد داخل متغیر i و دستور بعد چاپ رشته "\n" که حکم عمل دکمه Enter در برنامه Word که باعث می شود به سطر بعد برویم دارد.
- همین طور این دستورات درون حلقه اجرا می شود تا این که شرط حلقه نادرست (false) شود یعنی مقدار i بزرگ تر از ۱۰ شود.

حلقه **while** را بررسی کردیم حالا می رویم به سراغ حلقه دیگری که زبان برنامه نویسی جاوا برامون فراهم کرده آشنا شویم:

حلقه for

توصیف:

- تکرار متوالی یک یا چندین دستور بر اساس تعداد تکرار یا شرط خاص موجود در حلقه
- این حلقه از سه بخش تشکیل شده است:
  ۱. تعریف متغیر و مقدار دهی اولیه به متغیر
  ۲. شرط ،که می تواند براساس تعداد تکرار یا شرطی خاص و..... باشد
  ۳. عملیاتی ریاضی جبری نظیر جمع ،ضرب ،تقسیم ، تفریق و...که روی متغیر مورد ۱ اعمال می شود.
- به طور کلی می توان گفت این حلقه دارای یک شمارنده هست که شرط آن (تعداد تکرار اجرای دستورات درون حلقه) براساس مقدار شمارنده تعیین می شود که متغیری که نقش شمارنده دارد تا شرط و عملیات بر روی شمارنده درون پرانتز روبروی دستور **for** قرار میگیرد.
- **for** تمامی حروفش از حرف انگیسی کوچک می باشد

شکل پیاده سازی حلقه for :

```
for(initialization; Boolean_expression; update)
{
//Statements
}
```

- Statements ممکن است شامل یک دستور که اجباری نیست درون دو بلوک {} قرار گیرد یا شامل چندین دستور که لازم است این دستورات درون دو بلوک {} قرار بگیرند.
- جز به جز این دستور را بصورت زیر بررسی میکنیم:

Initialization

- در این گام یک متغیر تعریف کرده و آن را مقدار دهی اولیه می کنیم که به این متغیر شمارنده می گویند.متغیری که اساس تعداد تکرار حلقه ما را تشکیل می دهد.
- اولیت اجرا در حلقه: این گام اولین بار و فقط برای یک بار در حلقه اجرا می شود.
- و در پایان آن علامت کاما نقطه (semi colon) (;) می گذاریم.

Boolean expression

- گام بعدی ارزیابی **Boolean expression** می باشد که عبارت شرط ما را تشکیل می دهد.اگر درست (true) بود دستورات درون بدنه حلقه **for** اجرا می شود. اگر نادرست (false) بود هیچ دستوری درون بدنه این حلقه اجرا نمی شود به عبارت دیگر هرگز وارد بدنه حلقه نمی شویم!!! و کامپایلر از اجرای دستورات این حلقه صرف نظر کردن به اصطلاح از روی آن می پرد!!!! و دستور خط بعد حلقه **for** را اجرا میکند.
- این عبارت شرطی تعداد تکرار حلقه ما را مشخص می کند.
- هربار شرط بررسی در صورت درست بودن دستورات درون بدنه حلقه اجرا می شود
- در صورتی که دیگر شرط بر قرار نباشد (نادرست (false)) از بدنه حلقه خارج می شویم.
- و در پایان آن علامت کاما نقطه (semi colon) (;) می گذاریم.

update

- هر بار بعد از اجرای دستورات درون بدنه حلقه کامپایلر میاد سراغ دستور موجود در بخش **update** و آن را اجرا میکند.
- این دستور به شما اجزا میدهد که متغیری که ابتدا در حلقه **for** تعریف و مقدار دهی کردید رو دستکاری(نظیر عملیات جبری ریاضی) کنید.

به طور کلی روش عمل کردن و اولیت اجرا در حلقه **for** بصورت زیر است:

- ابتدا و فقط برای یک بار بخش `initialization`
- بعد عبارت `Boolean_expression` اگر درست بود بدنه حلقه اجرا می شود بعد از بدنه حلقه بخش `update` اجرا می شود و دوباره عبارت `Boolean_expression` بررسی می شود اگر درست بود همان روال قبل یعنی اجرای دستورات درون بدنه حلقه و سپس بخش `update` اجرا می شود و دوباره سراغ شرط و ادامه تا زمانی که عبارت `Boolean_expression` نادرست (false) شود که از حلقه خارج می شویم.
- اولیت تکرار در حلقه for به روایتی دیگر ☺

زمانی که برای اولین بار حلقه for اجرا می شود و شرط برقرار باشد:

```
for(initialization; Boolean_expression; update)
{

//Statements
      }
```

(1) (2) (4) (3)

و در صورتی که شرط برقرار باشد ادامه تکرار بصورت زیر است:

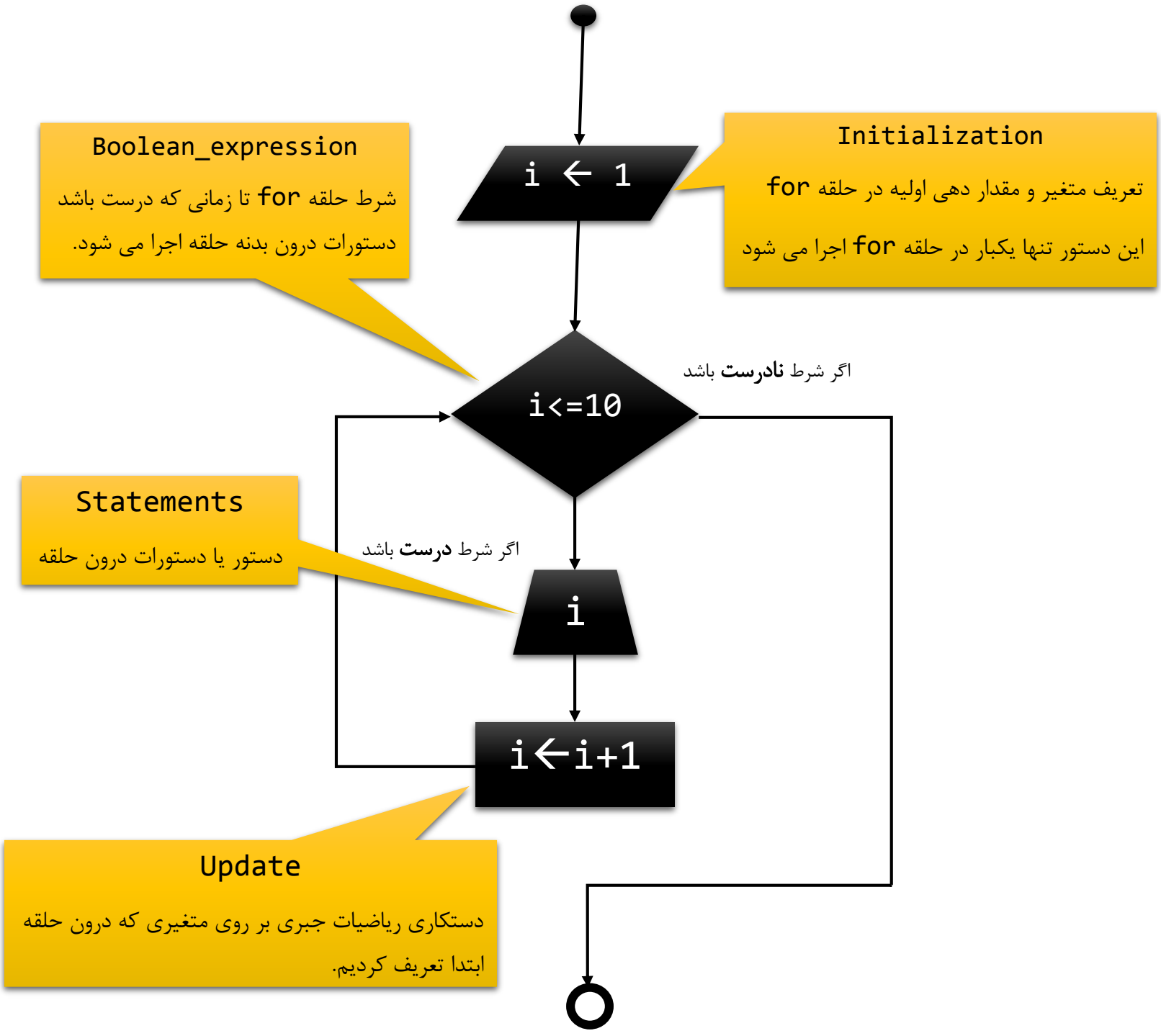
```
for(initialization; Boolean_expression; update)
{

//Statements

}
```

(1) (3) (2)

```
for (int i = 1; i <= 10; i++) {

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

```
}
```

فلوچارت عملکرد حلقه **for** مثال بالا بصورت زیر است:



مثال زیر اعداد ۱ تا ۱۰ را چاپ می کند.

```
package iran;

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

		for (int i = 1; i <= 10; i = i + 1) {
			System.out.print("value of i : " + i);
			System.out.print("\n");
		}
	}
}
```

خروجی:

```
value of i : 1
value of i : 2
value of i : 3
value of i : 4
value of i : 5
value of i : 6
value of i : 7
value of i : 8
value of i : 9
value of i : 10
```

```
for (int i = 1; i <= 10; i = i + 1) {
			System.out.print("value of i : " + i);
			System.out.print("\n");
		}
```

- به متغیرهایی نظیر i که به این شکل تعریف و یکی یکی به مقدار آن اضافه می شود شمارنده می گویند.
- همان طور که مشاهده می کنید ابتدا متغیری به نام i تعریف و مقدار ۱ به ان نسبت داده شده است.
- سپس شرط " i <= 10 " را داریم که تا زمانی که درست باشد دستورات درون بدنه حلقه for اجرا می شود.
- بعد از بررسی شرط و اجرای دستور درون بلوک بدنه حلقه for سراغ دستور "i=i+1" می رویم و بعد از آن دوباره بررسی شرط همون آش و همون کاسه ☺
- دستور زیر معادل حلقه for بالا می باشد:در این دستور بجای i=i+1 معادل آن i++ استفاده کردیم.

```
for (int i = 1; i <= 10; i++) {
			System.out.print("value of i : " + i);
```

```
            System.out.print("\n");
        }
```

حلقه do...while

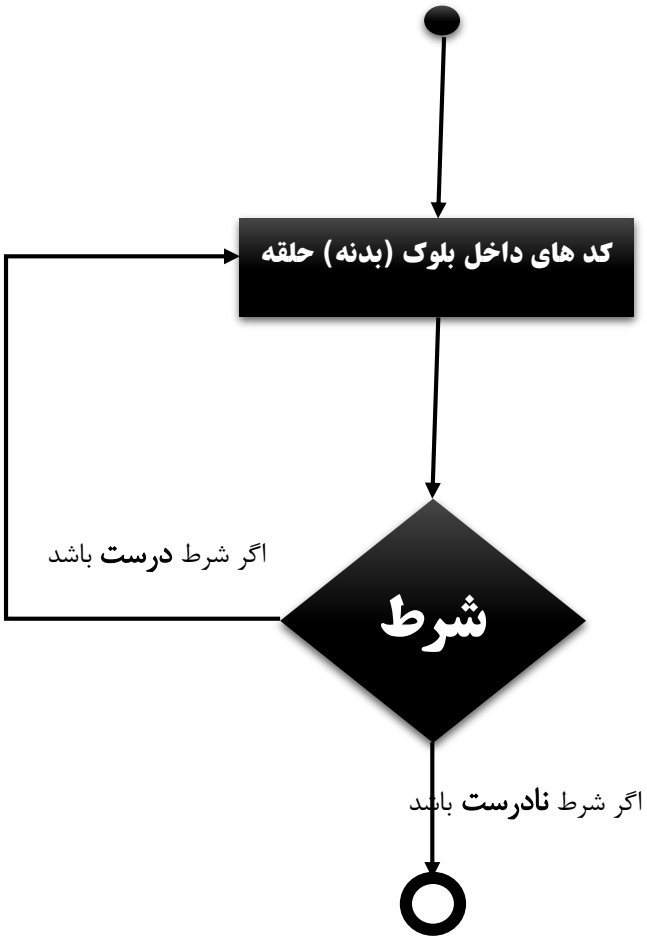
توصیف:

- شبیه به حلقه while می باشد با این تفاوت که در تکرار ابتدا دستورات درون بدنه حلقه را اجرا کرده و سپس شرط را بررسی میکند.

شکل پیاده سازی حلقه do...while:

```
do
{
//Statements
}while(Boolean_expression);
```

- Statements ممکن است شامل یک دستور که اجباری نیست درون دو بلوک {} قرار گیرد یا شامل چندین دستور که لازم است این دستورات درون دو بلوک {} قرار بگیرند.
- ابتدا دستورات درون بدنه حلقه (Statements) اجرا می شود و سپس عبارت شرط (Boolean_expression) بررسی می شود در صورت درست بودن تکرار دستورات درون حلقه ادامه پیدا میکند.
- پروسه تکرار دستورات درون حلقه do...while تا زمانی که شرط (Boolean_expression) درست باشد ادامه پیدا میکند و زمانی که شرط نادرست شد (false) شد از حلقه خارج می شویم و دستور خط بعد از حلقه را اجرا می کند.
- تفاوتی که این حلقه با سایر حلقه ها دارد حداقل برای یکبار دستورات درون حلقه اجرا می شود حتی اگر شرط نادرست باشد زیرا این حلقه ابتدا دستوران درون بدنه حلقه را اجرا و سپس شرط را بررسی میکند.
- فلوچارت عملکرد حلقه **do…while** بصورت زیر است:



در مثال زیر اعداد ۱ تا ۱۰ چاپ می شود:

```
package iran;

public class Eleventh_Session {
	public static void main(String args[]) {
		int i = 1;

		do {
			System.out.print("value of i : " + i);
			i++;
			System.out.print("\n");
		} while (i <= 10);
	}
```

```
}
```

خروجی:

```
value of i : 1
value of i : 2
value of i : 3
value of i : 4
value of i : 5
value of i : 6
value of i : 7
value of i : 8
value of i : 9
value of i : 10
```

```
do {
                System.out.print("value of i : " + i);
                i++;
                System.out.print("\n");
            } while (i <= 10);
```

- چون دستورات درون بدنه حلقه بیش از یک دستور است از دو بلوک {} استفاده کردیم.
- مقدار متغیر i (شمارنده) ما ابتدا ۱ بوده و مقدار i چاپ و سپس یکی به مقدار i اضافه می شود و مقدار i به ۲ تغییر میکند و سپس به سطر بعد می رویم و شرط بررسی می شود چون مقدار i که ۲ می باشد کمتر از ۱۰ هست دوباره دستورات درون شرط اجرا و باز شرط برسی و.....تا اینکه مقدار i از ۱۰ بزرگ تر شود در این صورت از حلقه خارج می شویم.

کنترل دستورات درون بدنه حلقه

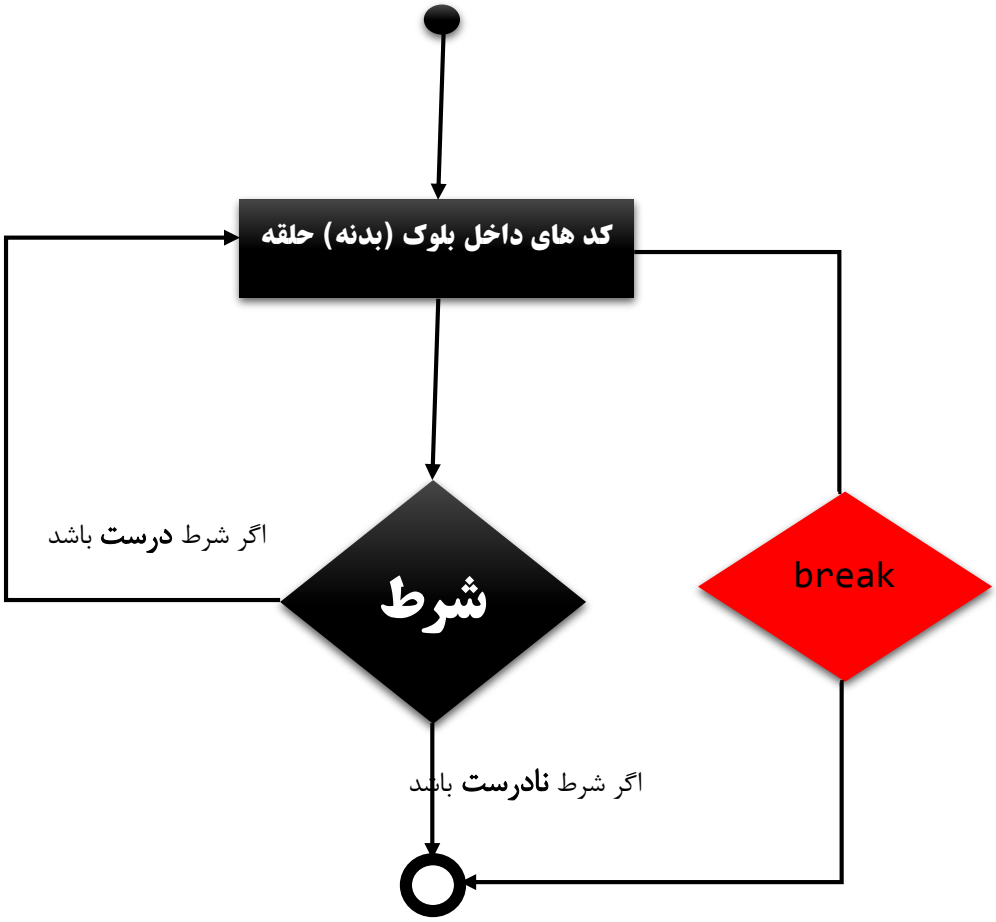
توصیف:

- دستوراتی که روند عادی تکرار دستورات درون بدنه حلقه را تغییر می دهد! به نوعی کنترل حلقه خود را به دست میگیریم!!!
- گاهی نیاز پیدا میکنیم چوب لای چرخ روند عادی تکرار پی در پی حلقه خود کنیم!!!!! ☺
- جاوا از دستورات زیر برای کنترل حلقه ها استفاده می کند:

دستور break

توصیف:

- با این دستور بلافاصله ادامه اجرای دستورات درون بدنه حلقه متوقف و پایان می یابد حتی اگر شرط برقرار باشد!!!!
- وقتی این دستور در بدنه حلقه پیاده سازی شود دستورات خط بعد از این دستور دیگر اجرا نمی شود و به اصطلاح حلقه را می شکند و از آن خارج می شویم.
- این دستور را در مفاهیم دستورات شرطی جلسات قبل نظیر switch-case هم داشتیم که در هر case پیاده سازی میشد.
- فلوچارت عملکرد و تاثیر دستور **break** در حلقه را در زیر مشاهده میکنیم:



در مثال زیر اعداد ۱ تا ۱۵ شمرده و چاپ می شود:

```
package iran;

public class Eleventh_Session {
	public static void main(String args[]) {
		int i = 1;

		do {
			System.out.print("value of i : " + i);
			i++;
			if (i >15)
				break;
			System.out.print("\n");
		} while (i <= 100);
```

```
        }
}
```

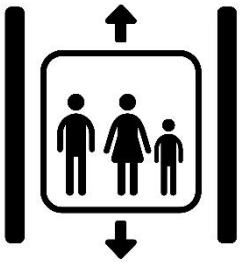
خروجی:

```
value of i : 1
value of i : 2
value of i : 3
value of i : 4
value of i : 5
value of i : 6
value of i : 7
value of i : 8
value of i : 9
value of i : 10
value of i : 11
value of i : 12
value of i : 13
value of i : 14
value of i : 15
```

هرچند یک شمارنده داریم که اعداد ۱ تا ۱۰۰ را می شمارد اما برخلاف شرط حلقه while (`i <= 100`) تنها اعداد ۱ تا ۱۵ شمرده و چاپ می شود زیرا در بدنه حلقه یک شرط if داریم که میگه اگر شمارنده i مقدارش از عدد ۱۵ بزرگ تر شد حلقه را break و متوقف کن و از آن خارج شو.

دستور continue:

توصیف:



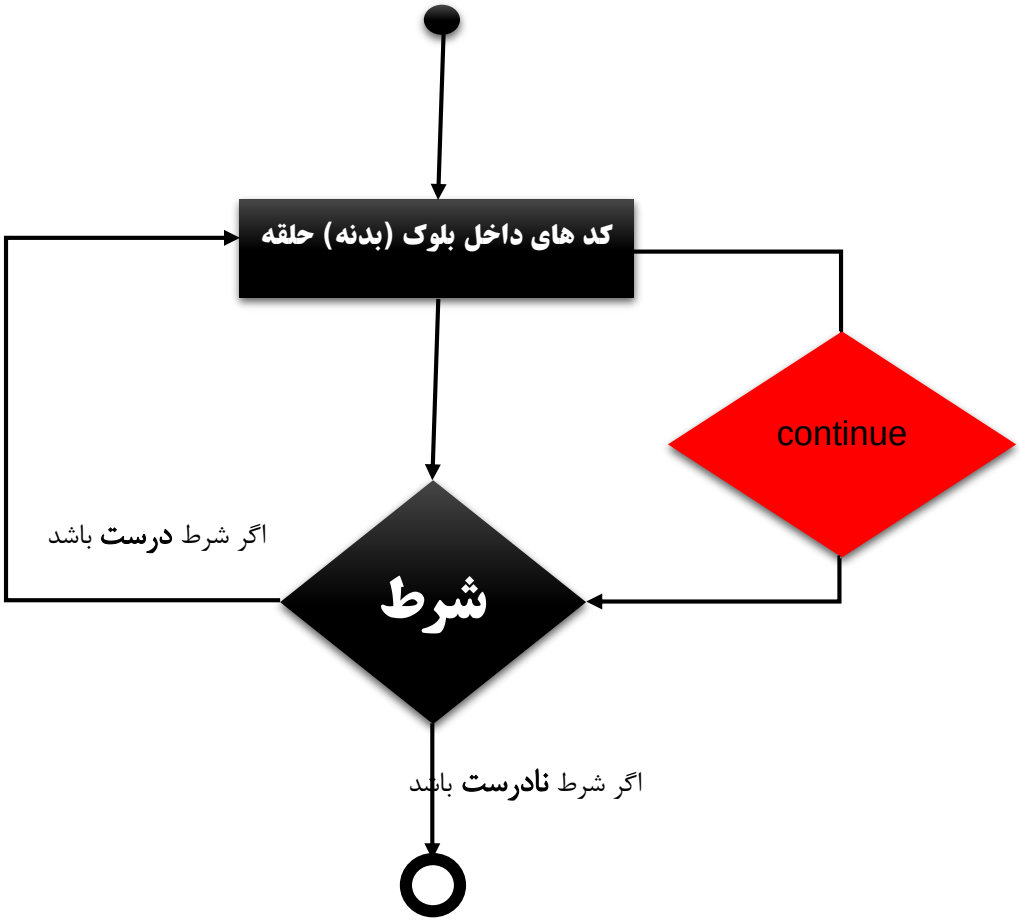
اگر این دستور در خطی از بدنه حلقه پیاده سازی شود باعث می شود که خطوط بعد از آن در حلقه اجرا نشود و تنها دستورات قبل از آن بصورت تکرار پی در پی اجرا شود!!!

برای درک بهتر این دستور به مثال زیر در دنیای واقعی توجه کنید!!!!!

مثال: فرض کنید یک خانواده ایرانی در یک ساختمان ۲۰ طبقه ، در طبقه ۱۵ زندگی می کنند!!!! در یک روز خوب پاییزی این خانواده از تفرجگاه برمی گردند و قصد دارند با آسانسور به طبقه ۱۵ بروند!!!!اما از انجایی که ساکنین طبقه ۱۰ به بعد پول شارژ ساختمان را پرداخت نکردن عمو جعفر مدیر ساختمان آسانسور دستکاری کرده که وقتی آسانسور به طبقه ۱۰ رسید جلوتر نره و بیاد سر جای اولش!!!!! این خانواده از طبقه اول سوار بر اسانسور می شوند اما همین که به طبقه ۱۰ رسیدن آسانسور سریع برمی گرده طبقه اول باز دکمه ۱۵ رو میزنند در کمال ناباوری باز اسانسور تنها تا طبقه ۱۰ رفته و سپس میاد طبقه اول از این خانواده اصرار و از آسانسور انکار ☺ طبق روال عادی آسانسور بایستی میرفت طبقه ۱۵ و

برمیگشت اما عمو جعفر با پیاده سازی یک **continue** در طبقه ۱۰ اجازه پیشروی آسانسور به طبقات بالاتر از ۱۰ رو نداده و باعث شده همین که آسانسور به طبقه ۱۰ رسید دوباره از نوع از طبقه اول شروع کنه بیاد بالا!!!!!

فلوچارت عملکرد و تاثیر دستور **continue** در حلقه را در زیر مشاهده میکنیم:



به مثال زیر توجه کنید:

```
package iran;

public class Eleventh_Session {
	public static void main(String args[]) {
		int i = 1;
		do {
			System.out.print("value of i : " + i + " ");
			i++;
			if (i > 11)
				continue;
			System.out.print("\n");
		} while (i <= 15);
```

```
    }

}
```

خروجی:

```
value of i : 1
value of i : 2
value of i : 3
value of i : 4
value of i : 5
value of i : 6
value of i : 7
value of i : 8
value of i : 9
value of i : 10
value of i : 11 value of i : 12 value of i : 13 value of i : 14 value of i : 15
```

در این مثال اعداد ۱ تا ۱۵ شمارش و چاپ شده اما به همان طور که مشاهده می کنید به بازه خاصی که رسیدیم روش چاپ شدن متفاوت شده است!!!! اعداد ۱ تا ۱۰ بصورت عادی سطر به سطر چاپ شده است!!!! اما با قرار دادن شرط زیر :

```
if (i > 11)
continue;
```

روال عادی چاپ سطر به سطر اعداد تغییر کرد و اعداد ۱۱ به بعد همگی در یک سطر چاپ شدند زیرا با قرار دادن شرط مذکور که حاوی دستور **continue** بود به برنامه گفته شد همین که شمارنده i مقدارش از ۱۱ بیشتر شد دیگر خط بعد از **continue**اجرا نشود و تنها خطوط قبل از **continue** اجرا کن.همون کار که عمو جعفر در آسانسور ایجاد کرد!!! باعث میشه حلقه تکرار تنها دستورات قبل از اجرا **continue** کند.

الگوریتم برنامه هم به این صورت هست که یک متغیر به عنوان شمارنده تعریف کردیم که وارد حلقه شده مقدار شمارنده چاپ و سپس یکی بهش اضاف میشه و اگر شرط if برقرار نباشد می رود سراغ خط بعد یعنی چاپ "\n" که باعث رفتن به سطر بعد در محیط کنسول می شود. و در پایان شرط بررسی میکنه اگر برقرار (درست) باشه دوباره دستورات درون بدنه حلقه اجرا و.... تا این که اگر شرط if برقرار بود دیگر دستور خط بعد از شرط if که حاوی **continue** می باشد را اجرا نمیکند مستقیم می رود سراغ شرطwhile در صورت درست بودن ، دستورات درون حلقه دوباره اجرا می شود اگر دوباره شرط if برقرار باشد دستور بعد از آن اجرا نمی شود اما اگر شرط if برقرار نبود برنامه روال عادی خود رو پی می گیره و تمام دستورات درون حلقه رو اجرا میکندو...

حلقه های بی نهایت!!!

توصیف: وقتی که شرط حلقه همیشه برقرار (درست) باشد دستورات درون بدنه حلقه بی نهایت اجرا و تکرار می شوند که آن وقت میگوییم حلقه ما بی نهایت است.

- اگر برنامه ما وارد یک حلقه بی نهایت شد هیچ وقت از ان حلقه خارج نمی شود و جز دستورات درون حلقه هیچ دستور دیگری در برنامه اجرا نمی گردد و برنامه ما پایانی ندارد!!!!
- حلقه های بی نهایت بعضی وقتا خواسته و بر روی نیاز مفید و گاهی ناخواسته که دچار آن شویم جز دردسر چیز دیگری ندارد!!!
- البته حلقه بی نهایت رو میشه با قرار دادن سری شروط کنترل کرد
- کابرد حلقه های بی نهایت در ساخت بازی ها نظیر جریان بازی ،ساخت منو،پیروز شدن و باختن در بازی و.....در کل جاهایی که قصد دارید یک جریان دائمی قابل کنترل پیاده سازی کنید قابل استفاده است.
- روش ایجاد حلقه های بی نهایت در انواع حلقه هایی که تا حالا بررسی کردیم بصورت زیر است:

- در مثال های زیر انوع حلقه های بی نهایت مثال زده که خروجی همگی بی نهایت بار چاپ "*" می باشد:

```
package iran;

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


		do {

			System.out.println("*");

		} while (true);
	}

}
```

```java
package iran;

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

		while (true) {

			System.out.println("*");
		}
	}

}
```

```java
package iran;

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

		for(;;) {

			System.out.println("*");
		}
	}

}
```

- در جلسات آینده که پروژه کار میکنیم با حلقه های بی نهایت بصورت کاربردی آشنا خواهیم شد.

حلقه های تو در تو!!!!

توصیف:

شناخته ترین و پر کاربرد ترین حلقه ها برای حلقه های تو در تو حلقه **for** می باشد.

- وقتی یک حلقه for درون حلقه for دیگر قرار گیرد می شود حلقه تو در تو به همین سادگی!!!
- گاهی برای چاپ جدول و کار با برنامه هایی که نیاز به سطر و ستون دارند یا برای پیاده سازی سایر نیازهامون در برنامه نویسی به حلقه تو در تو نیاز پیدا میکنیم.
- قالب پیاده سازی حلقه for تو در تو به شکل زیر است:

```
for (int i = 1; i <= 5; i++) {
for (int j = 1; j <=3; j++) {

}
}
```

- حلقه for ای که با رنگ سبز مشخص شده درون حلقه for زرد رنگ قرار دارد.
- عملکرد و اولیت اجرای این دو حلقه بصورت زیر است:

i	j
1	1
	2
	3

جدول(۱)

i	j
1	1
2	2
	3

جدول(۲)

i	j
1	1
2	2
3	3

جدول(۳)

i	j
1	1
2	2
3	3
4	

جدول(۴)

i	j
1	1
2	2
3	3
4	
5	

جدول(۵)

- وقتی مقدار متغیر (شمارنده) i حلقه اول (زرد) ۱ می باشد دستورات درون حلقه دوم (سبز) ۳ بار اجرا می شود.جدول(۱)
- وقتی مقدار متغیر (شمارنده) i حلقه اول (زرد) ۲ می باشد دستورات درون حلقه دوم (سبز) ۳ بار اجرا می شود.جدول(۲)
- وقتی مقدار متغیر (شمارنده) i حلقه اول (زرد) ۳ می باشد دستورات درون حلقه دوم (سبز) ۳ بار اجرا می شود.جدول(۳)
- وقتی مقدار متغیر (شمارنده) i حلقه اول (زرد) ۴ می باشد دستورات درون حلقه دوم (سبز) ۳ بار اجرا می شود.جدول(۴)
- وقتی مقدار متغیر (شمارنده) i حلقه اول (زرد) ۵ می باشد دستورات درون حلقه دوم (سبز) ۳ بار اجرا می شود.جدول(۵)
- ابتدا که متغیر i درون حلقه اول (زرد) تعریف و مقدار دهی می شود کنترل برنامه می رود سراغ اجرای دستور درون بدنه حلقه اول (زرد) که داخل بدنه این حلقه اول خود حلقه دومی می باشد پس متغیر (شمارنده) j درون حلقه دوم (سبز) تعریف و مقدار دهی می شود و سپس دستورات درون حلقه دوم (سبز) اجرا می شود و سپس می رود سراغ بروز رسانی متغیر حلقه

دوم (سبز) j++ و بروز رسانی انجام می شود و بعد شرط حلقه دوم (سبز) بررسی میکند در صورت برقرار بودن باز دستورات تکرار و بروز رسانی انجام می شود تا اینکه دیگر شرط حلقه دوم (سبز) برقرار نمی باشد،از حلقه خارج می شویم و کنترل برنامه می رود سراغ اجرای بروز رسانی حلقه اول i++ (چون حلقه دوم دستورات حلقه اول است و بعد از اجرا دستورات درون هر حلقه کنترل برنامه به سراغ بخشش بروز رسانی متغیر می رود)و بعد شرط حلقه اول (زرد) بررسی می شود در صورت درست بودن باز وارد حلقه دوم باز شمارنده حلقه دوم از نو تعریف و دستورات و تکرار حلقه انجام می شود و ادامه ماجرا تا اینکه شرط حلقه اول زرد برقرار نباشد و از کل حلقه خارج می شویم.

- با اولیت اجرا در حلقه for اشنا هستیم ابتدا تعریف و مقدار دهی متغیر فقط برای یک بار اجرا می شود و سپس ترتیب اجرای تکرار بصورت شرط ← دستورات درون بدنه حلقه ← و بخش دستکاری یا بروز رسانی مقدار متغیر
- خب ترتیب اجرای حلقه for تو در تو هم همین شکل است با این تفاوت که دستورات درون بدنه حلقه اول خود یک حلقه است که به همین ترتیب دستورات درون خود را تکرار میکند .

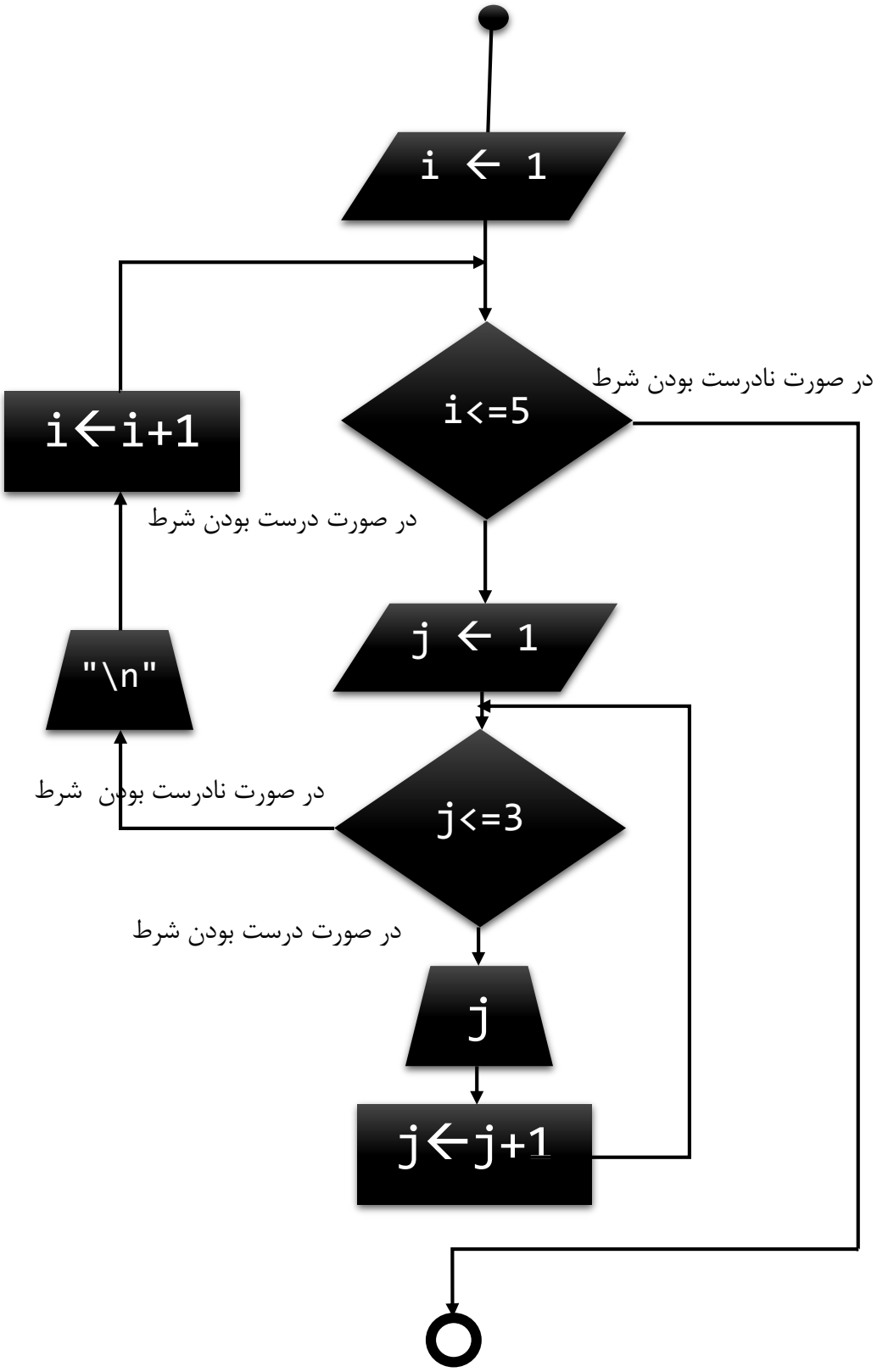
مثال زیر رو نگاه کنید:

```
package iran;

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

		for (int i = 1; i <= 5; i++) {
			for (int j = 1; j <= 3; j++) {
				System.out.print(j+" ");
			}
			System.out.print("\n");
		}
	}
}
```

خروجی :

```
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3
```

فلوچارت عملکرد این مثال در زیر آمده است ، ترتیب اجرای دستورات درون فلوچارت رو بدقت بررسی کنید.


i ← 1
i<=5
در صورت نادرست بودن شرط
i←i+1
در صورت درست بودن شرط
j ← 1
"\n"
در صورت نادرست بودن شرط
j<=3
در صورت درست بودن شرط
j
j←j+1

مثال:

```
package iran;

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

		for (int i = 1; i <= 5; i++) {
			for (int j = 1; j <= 3; j++) {
				System.out.print(i+j+" ");
			}
			System.out.print("\n");
		}
	}
}
```

خروجی:

```
2 3 4
3 4 5
4 5 6
5 6 7
6 7 8
```

- فرآیند چاپ خروجی:
- اصطلاحات: out خروجی برنامه،،،،،،،new line رفتن به سطر بعد

```
i=1;
j=1;
out:i+j+" "=2 ;          2
```

```
i=1;
j=2;
out:i+j+" "=3 ;          2 3
```

```
i=1;
j=3;
out:i+j+" "=4 ;          2 3 4
```

```
out:"\n"=new line;
```

```
i=2;
j=1;
out:i+j+" "=3 ;          2 3 4
                         3
```

```
i=2;
j=2;
out:i+j+" "=4 ;          2 3 4
                         3 4
```

```
i=2;
j=3;
out:i+j+" "=5 ;          2 3 4
                         3 4 5
```

```
out:"\n"=new line;
```

```
i=3;
j=1;
out:i+j+" "=4 ;          2 3 4
                         3 4 5
                         4
```

```
i=3;
j=2;
out:i+j+" "=5 ;          2 3 4
                         3 4 5
                         4 5
```

```
i=3;
j=3;
out:i+j+" "=6 ;          2 3 4
                         3 4 5
                         4 5 6
```

```
out:"\n"=new line;
```

```
i=4;
j=1;
out:i+j+" "=5 ;          2 3 4
                         3 4 5
                         4 5 6
                         5
```

```
i=4;
j=2;
out:i+j+" "=6 ;          2 3 4
                         3 4 5
                         4 5 6
                         5 6
```

```
i=4;
j=3;
out:i+j+" "=7 ;          2 3 4
                         3 4 5
                         4 5 6
                         5 6 7
```

```
out:"\n"=new line;
```

```
i=5;
j=1;
out:i+j+" "=6 ;          2 3 4
                         3 4 5
                         4 5 6
                         5 6 7
                         6
```

```
i=5;
j=2;
out:i+j+" "=7 ;          2 3 4
                         3 4 5
                         4 5 6
                         5 6 7
                         6 7
```

```
i=5;
j=3;
out:i+j+" "=8 ;          2 3 4
                         3 4 5
                         4 5 6
                         5 6 7
                         6 7 8
```

مثال:برنامه ای به زبان جاوا بنویسید که که جدول ضرب ۱ تا ۷ را پیاده سازی کنید.

```
package iran;

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

		for (int i = 1; i <= 7; i++) {
			for (int j = 1; j <= 7; j++) {
				System.out.print(j + " * " + i + " = " + (i * j) + "\t");
			}
			System.out.print("\n");
		}
	}
}
```

خروجی:

- عبارت "\t" همان عمل دکمه Tab کیبورد در برنامه word را انجام می دهد یعنی هشت فاصله (" ") ایجاد می کند.
- عبارت "\n" باعث می شود به سطر جدید برویم.عمل دکمه Enter کیبورد در برنامه Word را انجام می دهد.

```
1 * 1 = 1    2 * 1 = 2    3 * 1 = 3    4 * 1 = 4    5 * 1 = 5    6 * 1 = 6    7 * 1 = 7
1 * 2 = 2    2 * 2 = 4    3 * 2 = 6    4 * 2 = 8    5 * 2 = 10   6 * 2 = 12   7 * 2 = 14
1 * 3 = 3    2 * 3 = 6    3 * 3 = 9    4 * 3 = 12   5 * 3 = 15   6 * 3 = 18   7 * 3 = 21
1 * 4 = 4    2 * 4 = 8    3 * 4 = 12   4 * 4 = 16   5 * 4 = 20   6 * 4 = 24   7 * 4 = 28
1 * 5 = 5    2 * 5 = 10   3 * 5 = 15   4 * 5 = 20   5 * 5 = 25   6 * 5 = 30   7 * 5 = 35
1 * 6 = 6    2 * 6 = 12   3 * 6 = 18   4 * 6 = 24   5 * 6 = 30   6 * 6 = 36   7 * 6 = 42
1 * 7 = 7    2 * 7 = 14   3 * 7 = 21   4 * 7 = 28   5 * 7 = 35   6 * 7 = 42   7 * 7 = 49
```

مثال: برنامه ای به زبان جاوا بنویسید که یک مثلث به شکل زیر پیاده سازی و چاپ کند:

خروجی:

```
$
$$
$$$
$$$$
$$$$$
$$$$$$
$$$$$$$
```

پاسخ :

```
package iran;

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

		for (int i = 1; i <= 7; i++) {
			for (int j = 1; j <= i; j++) {
				System.out.print("$");
			}
			System.out.print("\n");
		}
	}
}
```

پیروز و موفق باشید

سایت آموزش زبان جاوا به زبان ساده،آسان و شیرین!!!

www.JAVAPro.ir

آموزش جاوا SE را با تجربه شخصی و به زبان خودمونی یاد بگیرید!!!!!

بازدید ازکانال

بازدید از سایت

هر روز مفاهیم و مثال های جدید به سایت اضافه می شود برای اطلاع از مطالب جدید روی سایت عضو کانال شوید.