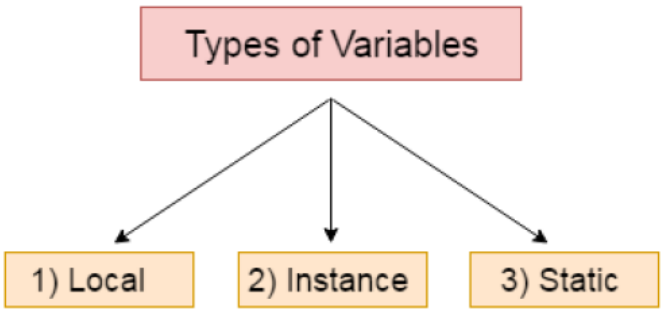
متغیرهای جاوا

- متغیرها : متغیرها به عنوان یک ظرف واحد برای نگهداری مقادیر و ارزش ها استفاده می شود. جاوا شامل سه نوع کلی متغیر می باشد: متغیرهای محلی، متغیرهای کلاس (استاتیک) ، متغیرهای نمونه (غیر استاتیک)



- **متغیرهای محلی local** : این متغیرها در داخل متد ، ساختار یا بلوک تعریف می شوند.و فقط در همان محدوده معتبر است.
- **متغیرهای نمونهinstance** : این متغیرهای داخل یک متد تعریف شده اما در سایر متدهای کلاس نیز معتبر است.
- **متغیرهای کلاس static** : این متغیرها داخل ی کلاس و خارج از متدهای آن با کلمه ی کلیدی static تعریف می شوند.

شناسه های جاوا

تمام اجزای جاوا نیاز به نام دارند. نام هایی که برای کلاس ها، متغیرها و متدها استفاده می شود ، شناسه نام دارند. در تعریف شناسه در جاوا باید نکات زیر را در نظر داشته باشید:

- تمام شناسه ها باید با یک حرف (A به Z یا a به z)، ارز ($) یا یک زیر خط (_) شروع شود.
- پس از اولین کاراکتر، شناسه می تواند هر ترکیبی از کاراکترها را داشته باشد.
- یک کلمه کلیدی نمی تواند به عنوان یک شناسه استفاده شود.
- مهمتر از همه اینکه، شناسه ها حساس به حروف هستند.
- نمونه هایی از شناسه های معتبر در جاوا :age, $salary, _value, __1_value .
- نمونه هایی از شناسه های غیر معتبر : ۱۲۳abc، -salary.

- **کلمات کلیدی جاوا**

لیست زیر، کلمات رزرو شده در جاوا را نشان می دهد. این کلمات رزرو شده ممکن است به عنوان متغیر یا هر شناسه دیگری تعریف شده باشند، که از آن ها نمی توان برای تعریف دیگری استفاده کرد:

abstract	assert	boolean	break	byte	case	catch	char
class	const	continue	default	do	double	else	if
enum	extends	final	finally	float	for	goto	static
public	implements	import	instanceof	int	interface	switch	short
long	native	new	package	this	private	protected	throws
while	volatile	void	synchronized	try	super	transient	throw

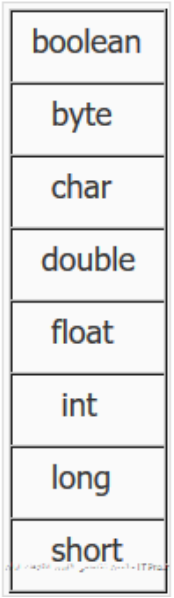
انواع داده

انواع داده دو دسته هستند :

1. نوع داده اصلیPrimitive
2. نوع داده ارجاعی یا شیئیReference

نوع داده اصلی

در زبان جاوا 8 نوع داده اصلی وجود دارد. انواع داده اصلی در زبان جاوا از قبل تعریف شده اند و قابل گسترش نیستند. به عبارت دیگر انواع داده اصلی به عنوان پایه انواع داده های دیگر می باشد. انواع داده اصلی دارای نام هایی هستند که این نام ها جزء کلمات کلیدی است.



1. byte

نوع داده byte دارای خواص زیر است:

- نوع داده byte معادل 8 بیت حافظه لازم دارد.
- فقط می توان عدد بدون ممیز در این نوع داده ذخیره کرد
- این نوع داده کوچکترین عددی را که می تواند در خود نگه دارد -128 است
- بزرگترین عددی که این نوع داده می تواند در خود نگه دارد 127 می باشد.
- مقدار پیشفرض این نوع داده 0 است. به این معنی که وقتی که یک متغیر از نوع byte تعریف می کنیم کامپایلر مقدار آن را 0 قرار می دهد.
- کد زیر مثالی از تعریف متغیر از نوع داده byte است.

```
byte a=100;
byte b=-43;
```

2. short

این نوع داده اعداد بدون ممیز را در خود نگهداری می نماید.

- نوع داده short یک نوع داده علامت دار 16 بیتی است.
- کوچکترین مقداری که این نوع داده می تواند در خود نگهداری نماید -32768 می باشد.
- بزرگترین مقداری که این نوع داده می تواند در خود نگهداری نماید 32767 می باشد.
- مقدار پیشفرض این نوع داده 0 است.
- حافظه مصرفی این نوع داده نصف نوع داده int است.

مثال تعریف متغیر از نوع: short

```
short s=10000;
short r=-3400
```

3. int

- نوع داده int یک نوع داده عددی علامت داره 32 بیتی است.
- کوچکترین عددی که این نوع داده می تواند در خود نگهداری نماید -2147483648 می باشد.
- بزرگترین عددی که این نوع داده می تواند در خود نگهداری کند 2147483647 می باشد.
- عموما برای متغیرهای اعداد صحیح از این نوع داده استفاده می شود ولی برای استفاده بهینه تر از حافظه می توان از انواع داده قبلی هم استفاده کرد. ولی امروزه به علت بالا بودن حجم حافظه ها صرفه جویی در حافظه مسئله حادی نیست.
- مقدار پیشفرض این نوع داده 0 است.

در زیر مثالی از این نوع داده آورده شده است:

```
int m=60;
int c=-200000;
int h=800000;
```

4. long

- نوع داده long یک نوع داده برای نگهداری اعداد صحیح علامت دار است که 64 بیت حافظه اشغال می کند.
- کوچکترین عددی که این نوع داده می تواند در خود نگهداری کند -9223372036854775808 است
- بزرگترین عددی که این نوع داده می تواند در خود نگهداری کند عدد 9223372036854775807 می باشد.
- این نوع داده زمانی استفاده می شود که بخواهیم با اعداد صحیح بسیار بزرگ کار کنیم.
- مقدار پیشفرض این نوع داده 0L است.

در زیر مثالی از این نوع داده آورده شده است.

```
long a=1000000L;
```

5. float

- نوع داده float برای نگهداری اعداد ممیز دار با 32بیت است.
- این نوع داده بر اساس استاندارد ممیز شناور IEEE 754 کار می کند.
- مقدار پیشفرض این نوع داده 0.0f است
- این نوع داده برای جاهایی که دقت بسیار مهم است مورد استفاده قرار نمی گیرد. مثلا برای نگهداشتن واحدهای پولی از این نوع داده استفاده نمی شود چون دقیق نیست.

مثال تعریف متغیر از این نوع داده به شکل زیر است:

```
float f1=234.5f;
```

6. double

- این نوع داده یک نوع داده 64 بیتی دقیق است.
- این نوع داده از استاندارد ممیز شناور IEEE 754 استفاده می کند.
- این نوع داده نوع داده پیشفرض برای اعداد ممیزی می باشد.

- این نوع داده با این که از float دقیق تر است ولی هنوز خطا دارد و نباید برای مقادیر پولی مورد استفاده قرار گیرد.
- مقدار پیش فرض این نوع داده 0.0 dمی باشد.

مثال این نوع داده در کد زیر آمده است.

```
doubel d1=123.4;
```

7. boolean

- این نوع داده فقط دو مقدار به خود میگیردtrue,false
- این نوع فقط برای نگهداری نتایج عبارات شرطی مورد استفاده قرار می گیرد.
- مقدار پیشفرض این نوع داده false است.

مثالی از تعریف متغیر از این نوع داده در کد زیر آمده است.

```
boolean one=true;
```

8. char

- این نوع داده برای نگهداری کاراکتر های یونیکد می باشد.
- این نوع داده 16بیت فضا از حافظه برای خود می گیرد.

مثالی از این نوع داده به شکل زیر است:

```
char letter='a';
```

انواع داده ارجاعی

- انواع داده های ارجاعی از کلاس های تعریف شده ساخته می شوند. متغیرهایی که از این نوع ساخته می شوند برای دسترسی به اشیا ساخته شده از کلاس هستند.
- آرایه ها و اشیا از این نوع هستند و در دسته انواع داده ارجاعی قرار میگیرند.
- مقدار پیشفرض این نوع داده ها null است.
- یک متغیر از نوع ارجاعی می تواند به هر نوعی که با نوع خودش سازگار باشد انتساب داشته باشد.

مثالی از این نوع داده در کد زیر آمده است.

```
Animal a=new Animal("rabbit");
```

تعریف متغیر

برای متغیرها باید نوع داده آن را مشخص کنیم. این نوع داده نشان می دهد که متغیر چه مقدار حافظه اشغال کرده است و چه چیزی را می تواند در خود نگهداری نماید و چه عملیاتی بر روی آن قابل انجام است. شما باید هر متغیر را قبل از این که از آن استفاده کنید معرفی نمایید. شکل اصلی معرفی متغیر به شکل زیر است

```
data type variable [ = value][, variable [= value] ...] ;
```

کلمه Data type در خط بالا نشان دهنده یکی از انواع داده ای است که در مطلب قبلی به آن پرداخته شد. برای آن که چند متغیر پشت سرهم از یک نوع تعریف کنیم باید آنها را با استفاده از کاما یا ، جدا نماییم. در کد زیر نمونه های صحیحی از تعریف متغیر ها را مشاهده می کنید.

```
int a, b, c;          // معرفی سه متغیر.
int a = 10, b = 10;   // معرفی به همراه مقداردهی
byte B = 22;          // معرفی یک متغیر از نوع بایت
double pi = 3.14159; // معرفی متغیر برای عدد پی
char a = 'a';         // معرفی متغیر کاراکتری
```

توضیحات در جاوا

- برای افزودن توضیحات در داخل برنامه یا غیر فعال کردن کدها، همانند زبان های C و C++ باید توضیحات یا کدهای مورد نظر را داخل /* */ قرار دهید.
- همچنین برای غیر فعال کردن یک خط کد از " // " قبل از خط کد مورد نظر استفاده کنید.

عملگرهای جاوا

عملگرهای جاوا به دسته های زیر تقسیم می شوند که در ادامه مطلب هرکدام را به طور کامل شرح خواهیم داد

1. عملگرهای محاسباتی
2. عملگرهای رابطه ای
3. عملگرهای بیتی
4. عملگرهای منطقی
5. عملگرهای انتسابی

عملگرهای محاسباتی

عملگرهای محاسباتی در عبارات ریاضی و جبری استفاده می شوند.

دقت داشته باشید که برای دیدن نتیجه این عملگرها مقدار متغیر A برابر 10 و مقدار B مقدار 20 است. این عملگرها عبارتند از:

- **عملگر جمع کردن**: این عملگر دو مقدار عددی را با هم جمع می کند و علامت آن + می باشد. برای مثال نتیجه A+B برابر 30 خواهد شد.
- **عملگر تفریق کردن**: این عملگر مقدار عددی دوم را از مقدار عددی اول کم خواهد کرد و علامت این عملگر - می باشد. برای مثال نتیجه A-B برابر -10 خواهد شد.
- **عملگر ضرب** دو مقدار را در هم ضرب می نماید و علامت این عملگر می باشد برای مثال نتیجه AB برابر با 200 خواهد بود.
- **عملگر تقسیم:** این عملگر مقدار اول را بر مقدار دوم تقسیم می کند. علامت این عملگر اسلش *() می باشد. برای مثال نتیجه* BA برابر 2 خواهد بود. نکته ای که باید به آن توجه داشته باشید این است که مقدار مقسوم علیه در کامپیوتر نباید صفر باشد.
- **عملگر باقی مانده**: این عملگر مقدار اولی را بر مقدار دوم تقسیم می کند و باقی مانده را برمی گرداند. علامت این عملگر در جاوا ٪ می باشد. برای مثال نتیجه B%A برابر 0 خواهد بود.
- **عملگر افزایش:** این عملگر یک عملگر تک عملوندی است. به این معنی که فقط با یک عدد یا مقدار کار می کند. وظیفه این عملگر این است که به مقدار موجود یک واحد اضافه می نماید. علامت این عملگر ++ می باشد. برای مثال نتیجه B++ برابر 21 خواهد بود.
- **عملگر کاهش:** این عملگر از مقدار قبلی یک واحد کم می کند. علامت این عملگر -- می باشد. برای مثال عمل B-- برابر 19 خواهد بود.

نکته: اگر عملگر افزایش یا کاهش قبل یا بعد از عملوند قرار گیرد رفتار متفاوتی از خود نشان خواهد داد. به این شکل که اگر عملگر قبل از عملوند قرار بگیرد ابتدا عمل کاهش یا افزایش صورت گرفته و سپس عمل چاپ شدن انجام می شود.

عملگرهای رابطه ای

عملگرهای رابطه ای عملگرهایی هستند که در مورد رابطه دو مقدار باهم تصمیم گیری می کنند. نتیجه تصمیم گیری هم مقدار درست یا نادرست است (true/false)برای مثال A را برابر با 10 و B را برابر با 20 در نظر می گیریم. در ادامه به بررسی عملگرهای رابطه ای می پردازیم:

- **عملگر تساوی:** این عملگر برای مشخص کردن تساوی دو مقدار به کار می رود که علامت آن == می باشد و مشخص می کند که دو مقدار با هم مساوی هستند یا خیر. برای مثال نتیجه A==B برابر با false خواهد بود.
- **عملگر نامساوی:** این عملگر به این گونه عمل می کند که اگر دو مقدار با هم مساوی باشند مقدار true و اگر دو مقدار با هم مخالف باشند مقدار trueبرمی گرداند. علامت این عملگر در جاوا != می باشد. برای مثال نتیجه A!=B برابر با true خواهد بود.
- **عملگر بزرگتر:** این عملگر مشخص می کند که آیا مقدار سمت چپ از مقدار سمت راست بزرگتر است یا خیر. اگر مقدار سمت چپ بزرگتر بود جواب true و در غیر این صورت چواب false می دهد. علامت این عملگر >می باشد. برای مثال مقدار A>B نتیجه غلط یا False در پی خواهد داشت. دقت داشته باشید که اگر دو مقدار مساوی هم باشند این عملگر مقدار غلط در بر خواهد داشت.
- **عملگر کوچکتر:** این عملگر مخالف عملگر بزرگتر است و مشخص می کند که آیا مقدار سمت چپ عملگر از مقدار سمت راست عملگر کوچکتر است یا خیر. در صورت کوچکتر بودن جواب true می دهد. علامت این عملگر به شکل <می باشد. برای مثال مقدار عبارت A<B برابر true است.
- **عملگر بزرگتر مساوی** به این شکل است که اگر مقدار سمت چپ بزرگتر از مقدار سمت راست باشد یا با مقدار سمت راست مساوی باشد مقدار true برمی گرداند. علامت این عملگر به شکل =>برای مثال جواب A>=Bبرابر false خواهد بود.
- **عملگر کوچکترمساوی:** این عملگر مشخص می کند که عملوند سمت چپ از عملوند سمت راست کوچکتر است یا هردو عملوند با هم مساوی هستند. علامت این عملگر به شکل =<می باشد برای مثال نتیجه عبارت A<=Bبرابر true است.

عملگرهای بیتی

جاوا دارای چندین عملگر بیتی است که این عملگرها بر روی انواع داده int, long, short, char, byte قابل اعمال است. عملگرهای بیتی بر روی تک تک بیتهای عملوند ها کار می کنند. برای مثال در متغیر a مقدار 60 را قرار می دهیم و در متغیر b عدد 13 را قرار می دهیم. حال شکل باینری این دو متغیر به شکل زیر خواهد بود

```
a=00111100
b=00001101
```

- عملگر and بیتی (&) این عملگر بیت های دو عملوند را دو به دو با هم and می کند و مقدار را مشخص می نماید برای مثال نتیجه and بیتی a&b برابر 12 خواهد بود که مقدار باینری آن 00001100 می باشد.
- عملگر or بیتی این عملگر بیت های دو عملوند را دو به دو با هم or می نماید. برای مثال نتیجه عمل a|b برابر مقدار 61 یعنی عدد باینری 00111101 خواهد بود.

- عملگر xor بیتی (^) این عملگر بیت های دو عدد را با هم xor می نماید. یعنی اگر یکی از مقادیر مقدار یک و دیگری مقدار 0 داشته باشد جواب 1 و اگر هردو یکسان باشند جواب 0 می دهد. برای مثال نتیجه عمل a^bبرابر 49 که معادل باینری آن 00110001 می باشد است.
- عملگر مکمل بیتی(~) این عملگر یک عملگر تک عملوندی است که بیت های عملوند خود را برعکس می کند یعنی اگر بیت مورد نظر 1 باشد آن را 0 و اگر 0 باشد آن را 1 می کند. برای مثال a~ برابر -61 با عدد باینری 11000011 است.
- عملوند شیفت به چپ (<<) این عملوند مقدار عدد را به تعداد گفته شده به سمت چپ شیفت بیتی می دهد. برای مثال اگر بخواهیم a را دو بار به سمت چپ شیفت بدهیم باید به این شکل بنویسیم a<<2.در این صورت نتیجه برابر 240 خواهد بود زیرا که معادل باینری آن 11110000 می باشد.
- عملگر شیفت به راست (>>)این عملگر مقدار عدد را به تعداد گفته شده به راست شیفت می دهد. برای مثال نتیجه عمل a>>2 برابر مقدار 15 با عدد باینری 1111 خواهد شد.
- عملگر شیفت به راست با ورود 0 : عملگرهای شیفت به راست و شیفت به چپ که گفته شده مقدار عدد خارج شده را به عنوان مقدار ورودی شیفت وارد می کردند ولی عملگر شیفت به راست با ورودی 0 عدد را به سمت راست شیفت می دهد و مقدار صفر را از سمت چپ به آن وارد می نماید. برای مثال نتیجه عمل a>>>2 عدد 15 می باشد.

عملگرهای منطقی

عملگرهای منطقی بر روی داده های از نوع boolean و یا عباراتی که نتیجه boolean یعنی مقدار درست یا غلط دارند عمل می کنند. برای مثال فرض کنید که دو متغیر a, b داریم که مقدار a برابر true و مقدار b برابر false می باشد.

1. عملگر and منطقی (&&) این عملگر در صورتی جواب true برمی گرداند که هردو عملوند چپ و راست آن trueیا غیرصفر باشند. برای مثال نتیجه عمل a&&b مقدار false است.
2. عملگر or منطقی این عملگر در صورتی نتیجه true می دهد که یکی از عملوند ها true یا غیر صفر باشد. برای مثال نتیجه عمل a||b مقدار true است.
3. عملگر نقیض منطقی(!) این عملگر یک عملگر تک عملوندی است. اگر عملوند این عملگر درست باشد مقدار Falseو در غیر این صورت مقدار true برمی گرداند. برای مثال حاصل عبارت (a&&b)! مقدار true می باشد.

عملگرهای انتسابی

عملگرهای انتسابی که توسط جاوا پشتیبانی می شوند عبارتند از:

- عملگر انتساب(=) این عملگر مقدار سمت راست عملگر را در عملوند سمت چپ می ریزد. برای مثال C=A+Bمقدار C برابر با جمع A و B خواهد بود.
- عملگر جمع و انتساب(+=) این عملگر به این شکل عمل می کند که ابتدا عملوند سمت راست را با عملوند سمت چپ جمع نموده و در عملوند سمت چپ ذخیره می نماید. برای مثال C+=A معادل C=C+A می باشد.
- عملگر تفریق و انتساب(-=) مانند همان عملگر جمع و انتساب عمل می کند با این تفاوت که عملوند سمت راست را از عملوند سمت چپ کم کرده و ذخیره می کند. برای مثال عمل C-=A معادل C=C-A می باشد.
- عملگر ضرب و انتساب(*=) و عملگر (/=) و عملگر (٪=) معادل اعمال قبلی هستند.

تقدم عملگرها

ردیف	تقدم عملگر ها
1	. [] ()
2	~ ++ — !
3	* / %
4	+ –
5	<< >> >>>
6	>= < > <=
7	== !=
8	&
9	^
10	|
11	&&
12	||
13	? :
14	*= /= %= ^= = += -=
15	>>= >>>= &= |= <<=

ساختارهای تکرار

ممکن است که حالاتی پیش بیاید که بخواهیم یک قسمت از کد چند بار اجرا شود. زبان های برنامه نویسی راه های مختلفی فراهم کرده اند که این نوع اجرا خطی را تغییر دهد. یکی از این راه ها ساختارهای تکرار (یا حلقه loop) هستند. حلقه ها می توانند که یک سری دستورات را چندین بار اجرا کنند. در بسیاری از زبان های برنامه نویسی فلوچارت حلقه تکرار به شکل زیر است:

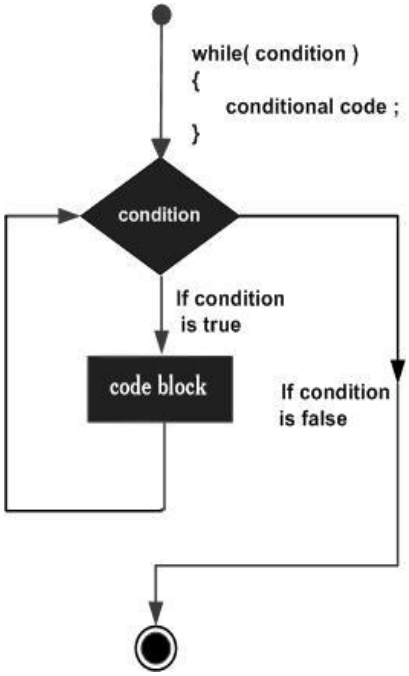
حلقهwhile

حلقه while در زبان جاوا تا وقتی که شرطش درست باشد یک قسمت از کد را اجرا می کند و وقتی که شرط آن falseشد از اجرا باز می ایستد. نحوه نوشتن کد این حلقه به شکل زیر است:

```
while(عبارت شرطی)
{
    //دستورات حلقه که باید تکرار شوند
}
```

دستورات داخل حلقه می تواند یک خط دستور باشند یا یک بلوک کد باشند. عبارت شرطی ممکن است که یک متغیر باشد و یا چندین عبارت شرطی باشد و یا مقادیر true و یا false باشد. هنگامی که نتیجه عبارت شرطی while درست باشد کد داخل حلقه اجرا خواهد شد و اگر false باشد اجرا نمی شود. نمودار دیاگرام این حلقه در شکل زیر آمده است:

نکته ای که وجود دارد این است که ممکن است یک حلقه while اصلا اجرا نشود. این در حالی است که شرط حلقه همان اول کار false باشد دراین حالت بدنه حلقه اصلا اجرا نخواهد شد و به کل نادیده گرفته می شود



حلقهfor

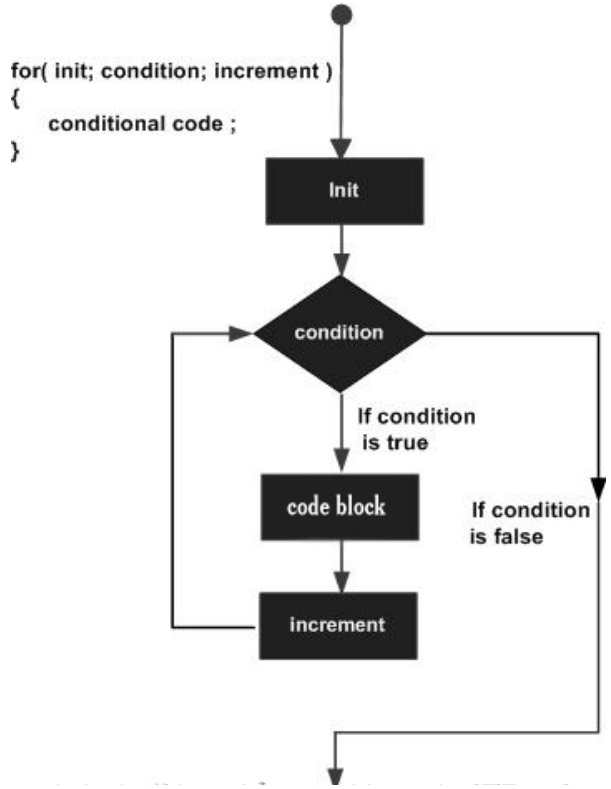
اگر شما می خواهید که قطعه کدی از برنامه شما به تعداد خاصی تکرار شود از حلقه for استفاده می کنیم. این حلقه در زمانی که تعداد تکرار یک قطعه کد از قبل مشخص باشد، مناسب است. شکل نوشتاری این حلقه به شکل زیر است:

```
for(initialization; Boolean_expression; update)
{
    //قطعه کد تکرار شونده
}
```

هریک از قسمت های موجود در این حلقه را در ادامه شرح می دهیم:

1. بخش initialization در ابتدا اجرا می شود. این قسمت در کل حلقه فقط یک بار اجرا خواهد شد. این مرحله به شما این اجازه ر ا می دهد که مقدمات اجرای حلقه را فراهم کنید. به عبارت دیگر متغیری که قرار است حلقه را کنترل کند در این بخش معرفی و مقداردهی می کنیم.
2. قسمت دوم Boolean Expression می باشد. در این بخش یک عبارت شرطی وجود دارد که اگر نتیجه این بخش true باشد حلقه اجرا خواهد شد و اگر false باشد حلقه اجرا نمی شود. در این بخش می توان متغیر کنترل کننده حلقه را چک کرد.
3. پس از این که بدن حلقه یعنی همان قطعه کد اجرا شونده اجرا شود قسمت update اجرا خواهد شد. دقت داشته باشید که قسمت update هر بار پس از اجرای بدنه حلقه اجرا می شود. در قسمت update می توانید اعمالی مانند زیاد یا کم کردن متغیر کنترل کردن حلقه در این قسمت انجام می شود. این قسمت از حلقه حتی می تواند خالی باشد.

زمانی که قسمت Boolean expression مقدار false برگرداند حلقه دیگر اجرا نمی شود. نمودار دیاگرام این حلقه به شکل زیر است:

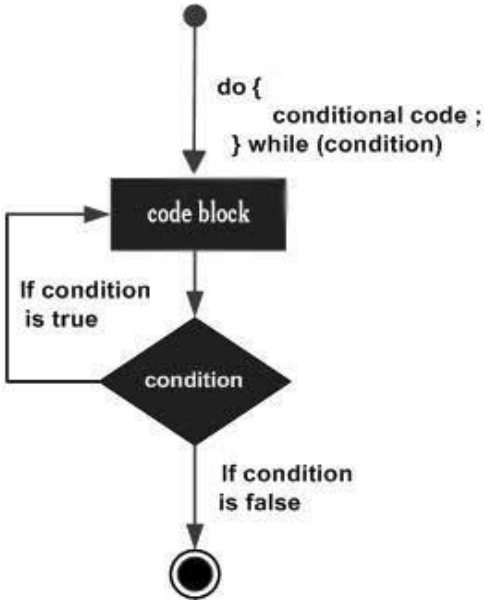


حلقهdo…while

این حلقه بسیار شبیه به while می باشد. تنها تفاوتی که بین این دو حلقه وجود دارد این است که شرط حلقه در انتها چک می شود. به همین دلیل کد بدنه حلقه حداقل یک بار حتما اجرا می شود. کد این حلقه به شکل زیر است:

```
do
{
    //بدنه حلقه
}while(شرط حلقه);
```

همانگونه که می بینید شرط حلقه در انتها است پس یک بار بدنه حلقه اجرا می شود. وقتی که به شرط حلقه می رسیم اگر مقدار true داشته باشد دوباره به ابتدای بدنه بازمی گردیم و دوباره حلقه اجرا می شود. اما اگر شرط حلقه falseباشد بازگشت انجام نمی شود و از حلقه خارج می شویم. دیاگرام این حلقه به شکل زیر است:



عبارات کنترل کننده حلقه ها

حلقه ها یک قسمت از کد را تکرار می کنند. حال اگر بخواهیم در حالت هایی حلقه در وسط اجرا تمام شود و از حلقه خارج شویم. یا بخواهیم برخی دستورات را اجرا نکند. برای این کار از برخی عبارات کنترل کننده حلقه ها استفاده می کنیم که در ادامه به شرح آنها می پردازیم:

- ✓ **دستورbreak**

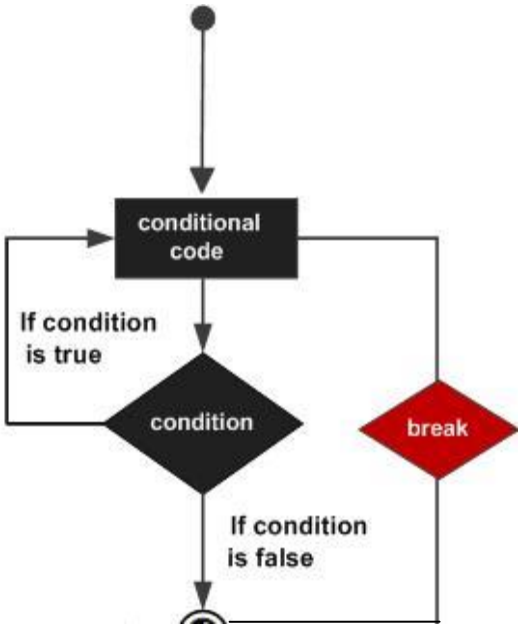
دستور break در زبان جاوا دو کاربرد دارد.

1. وقتی که در اجرای حلقه به خط break می نویسیم حلقه در همان نقطه تمام شده و خارج می شود. و برنامه از خط بعد از حلقه ادامه پیدا می کند.
2. در بلوک switch مورد استفاده قرار می گیرد که شرط case ها از یکدیگر جدا باشد .

نحوه نوشتن این دستور به شکل زیر است:

```
break;
```

دیاگرام این دستور به شکل زیر است:

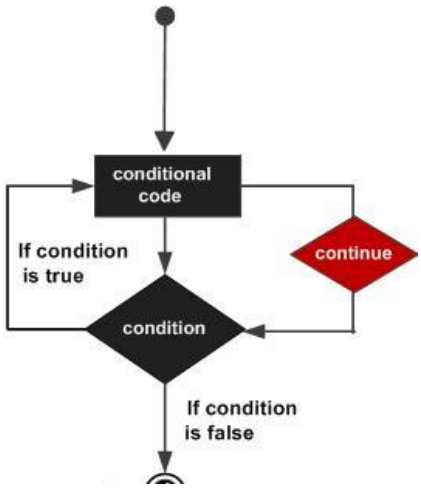


دستورcontinue

این دستور در همه ی حلقه ها قابل استفاده است. نحوه کار این دستور به این شکل است که در هنگام اجرای حلقه وقتی که به این دستور میرسیم در هر جای حلقه قرار داشته باشیم ادامه اجرا لغو می شود و به ابتدای حلقه بازمی گردیم. دقت داشته باشید که اگر در داخل حلقه for قرار داشته باشیم بخش update پس از رسیدن به continue اجرا خواهد شد و سپس شرط حلقه چک می شود. همچنین اگر در حلقه while یا do…while باشد وقتی به دستور continue می رسیم به ابتدای حلقه رفته و شرط حلقه چک می شود. شکل نوشتاری این دستور به شکل زیر است:

```
continue;
```

دیاگرام این دستور به شکل زیر خواهد بود:



ساختارهای شرطی

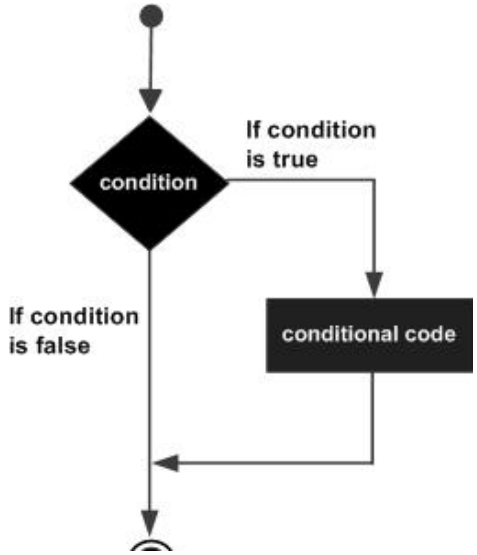
زبان جاوا عبارات شرطی زیر را فراهم می آورد که آنها را شرح خواهیم داد.

ساختار شرطیif

یک عبارت شرطی if از یک شرط و یک یا چند عبارت اجرایی تشکیل شده است. شکل نوشتاری آن مانند کد زیر است.

```
if(Boolean_expression)
{
   //Statements will execute if the Boolean expression is true
}
```

اگر شرط موجود در قسمت Boolean expression نتیجه true دهد پس بلوک کد داخل if اجرا می شود اما اگر مقدار شرط false باشد قسمت کد داخل بلوک if اجرا نمی شود و خط بعد از بلوک if اجرا خواهد شد. دیاگرام این شرط به شکل زیر خواهد بود.

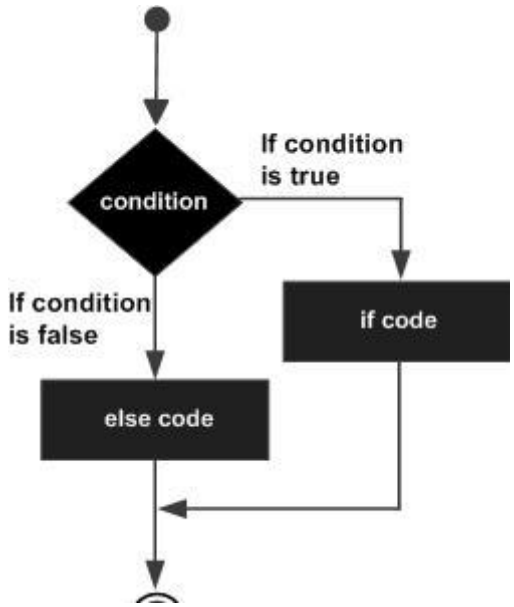


شرطif-else

یک شرط if می تواند با یک else همراه باشد. که نحوه ی این شرط به این شکل است که اگر شرط مورد نظر درست باشد دستور خود if و اگر شرط غلط باشد دستور else اجرا خواهد شد. نحوه نوشتاری این شرط به شکل زیر است:

```
if(Boolean_expression){
   //Executes when the Boolean expression is true
}else{
   //Executes when the Boolean expression is false
}
```

ردیاگرام این شرط به شکل زیر خواهد بود



شرط if… else if … else

یک شرط if می تواند با یک یا چند else if همراه باشد. این حالت زمانی استفاده می شود که بخواهیم حالت های مختلف شرط را بررسی کنیم. برای استفاده از این شرط باید به نکات زیر دقت کنید:

1. اگر پس از if از هر چند تا else if استفاده کنید اما بهتر است که در انتهای آن از یک بلوک else استفاده کنید.
2. هیچگاه نمی توان بعد از بلوک else از else if استفاده کرد.
3. اگر یک else if اجرا شود بقیه else if ها و else ها اجرا نخواهند شد.

نحوه نوشتن این شرط به شکل زیر خواهد بود:

```
if(Boolean_expression 1){
   //Executes when the Boolean expression 1 is true
}else if(Boolean_expression 2){
   //Executes when the Boolean expression 2 is true
}else if(Boolean_expression 3){
   //Executes when the Boolean expression 3 is true
}else {
   //Executes when the none of the above condition is true.
}
```

اگر مقدار Boolean expression 1 درست باشد بلوک if اجرا خواهد شد و اگر Boolean expression2 درست باشد بلوک else if اجرا خواهد شد و اگر شرط Boolean expression3 درست باشد else if دوم اجرا خواهد شد. اگر هیچکدام از شرط های گفته شده درست نباشند بلوک else اجرا می شود.

دستورات if تودرتو

در داخل هر سه شکل if که گفته شد می توان از دستوارت if به شکل های گفته شده به صورت تو در تو استفاده کرد. برای درک بهتر به مثال زیر توجه کنید:

```
if(Boolean_expression 1){
   //Executes when the Boolean expression 1 is true
   if(Boolean_expression 2){
      //Executes when the Boolean expression 2 is true
   }
}
```

فقط توجه داشته باشید که برای دستورات چند خطی می توان از {} برای ساخت بلوک دستور استفاده کرد ولی اگر دستور مورد نظر یک خطی باشد نیازی به استفاده از این علائم نیست. همچنین برای if های تو در تو باید قانون تقدم و تاخر بلوک ها حفظ شود به این معنی که باید همیشه بلوک های داخلی که دیرتر باز شده اند زودتر بسته شوند و بلوک های بیرونی که زودتر از همه باز شده اند دیرتر از همه بسته شوند. برای راحتی کار می توان همچنان که در برنامه های زبان جاوا مصطلح است بلوک های داخلی را کمی داخل تر از بلوک های خارجی نوشت. برای مثال در کد بالا if داخلی کمی فرورفته تر از if خارجی است. با رعایت این قائده کد شما بسیار خواناتر خواهد بود

شرطswitch

اگر یک متغیر چندین حالت داشته باشد و حالت های رخ دهنده آن متغیر برای ما مشخص باشد می توان از شرط switchبرای بررسی ان استفاده کرد. هر مقداری که ممکن است برای متغیر پیش بیاید یک case نامیده می شود و مقدار کنونی متغیر با کیس های گوناگون بررسی خواهد شد و اگر با یک کیس تناسب داشت کدهای متناظر با آن کیس اجرا خواهد شد. دقت کنید که همان طور که در توضیح break در بخش حلقه ها توضیح دادیم انتهای هر caseیک break خواهیم داشت. البته اگر در انتهای case از break استفاده نکنیم با case بعدی هم متغیر بررسی خواهد شد و اگر مناسب باشد کد آن case هم اجرا خواهد شد. شکل نوشتاری ان شرط به شکل زیر است:

```
switch(expression){
    case value :
       //Statements
       break; //optional
    case value :
       //Statements
       break; //optional
    //You can have any number of case statements.
    default : //Optional
       //Statements
}
```

برای استفاده از switch قواعد زیر وجود دارد.

1. نوع داده مقداری که به هر case می دهیم باید با نوع داده متغیری که برای switch استفاده کرده ایم یکی باشد. همچنین مقداری که برای case در نظر می گیریم باید ثابت باشد یا مقدار باشد و نمی توان یک مقدار نامعلوم در آن قرار داد.
2. وقتی که برنامه در حال اجرا به خط break برسد باید از کل بلوک switch خارجی می شود.
3. گذاشتن break برای هر case اختیاری است ولی اگر پس از کد case break گذاشته نشود case بعدی هم بررسی خواهد شد و در صورت تطبیق ان نیزاجرا خواهد شد تا وقتی که به یک break برسد و از بلوک switchخارج شود.

شکل دیاگرام این شرط به شکل زیر خواهد بود: