آموزش زبان برنامه نویسی جاوا

کلمه کلیدی static

جلسه دوازدهم

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!!

کلمه کلیدی **static** برای مدیریت حافظه استفاده می شود.

کلمه کلیدی استاتیک در جاوا می تواند برای متغیرها و متدهایی که متعلق به یک کلاس هستند (متغیر های نمونه و متد هایی که در بدنه کلاس تعریف می شوند) استفاده شود.پس متغیر های محلی نمی توانند استاتیک باشند.

خب چطور یک متغیر را **static** اعلام کنیم؟

وقتی ابتدای نوع یک متغیر کلمه کلیدی **static** استفاده کنیم:

```
int n;
```

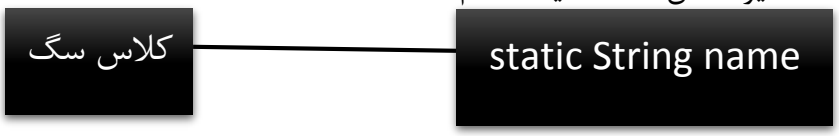
اعلام استاتیک بودن یک متغیر

```
static int n;
```

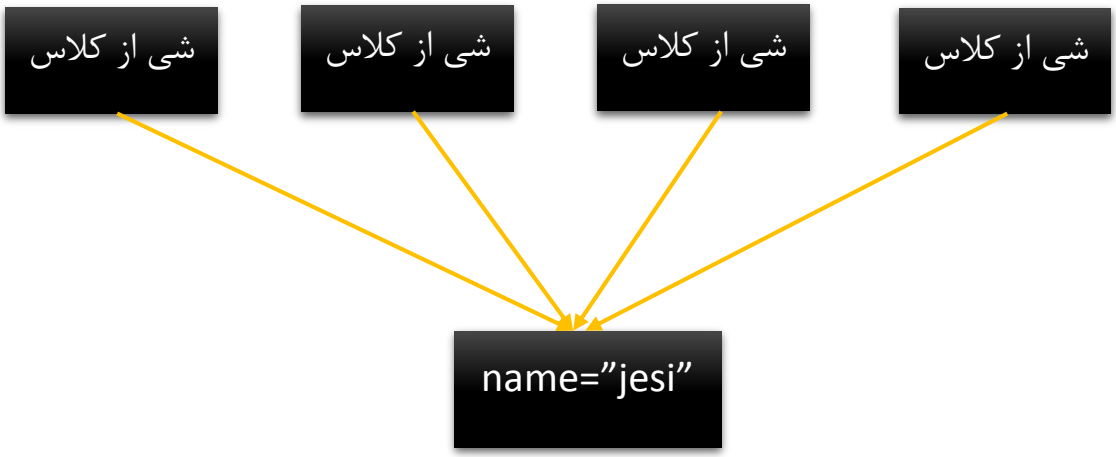
متغیر استاتیک تنها یک حافظه از کلاس را می گیرد یعنی مقدارش برای همه اشیای ساخته شده از کلاس یکسان است. برای روشن تر شدن این مطلب فرض کنید یک کلاس سگ داریم و ویژگی نام سگ استاتیک می باشد اگر ما ۱۰ سگ از این کلاس سگ بسازیم و نام "جسی" را روی یک سگ بگذاریم نام همهی ۱۰ سگ جسی خواهد شد زیرا متغیر نام استاتیک یا یکتا بوده و برای همه اشیای یک کلاس یکی است، در زیر برای درک بهتر تفاوت دسرسی اشیای ساخته شده از یک کلاس به متغیر استاتیک و غیر استاتیک آماده است:

دسرسی اشیای ساخته شده از یک کلاس به متغیر غیر استاتیک:

متغیر کلاس که استاتیک اعلام شده

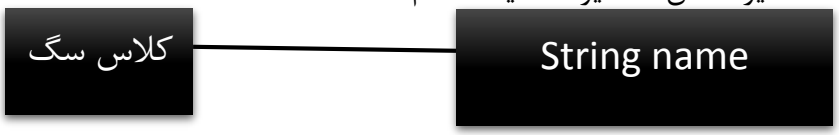


این متغیر استاتیک برای همه اشیای ساخته شده از کلاس مقدارش یکسان است.

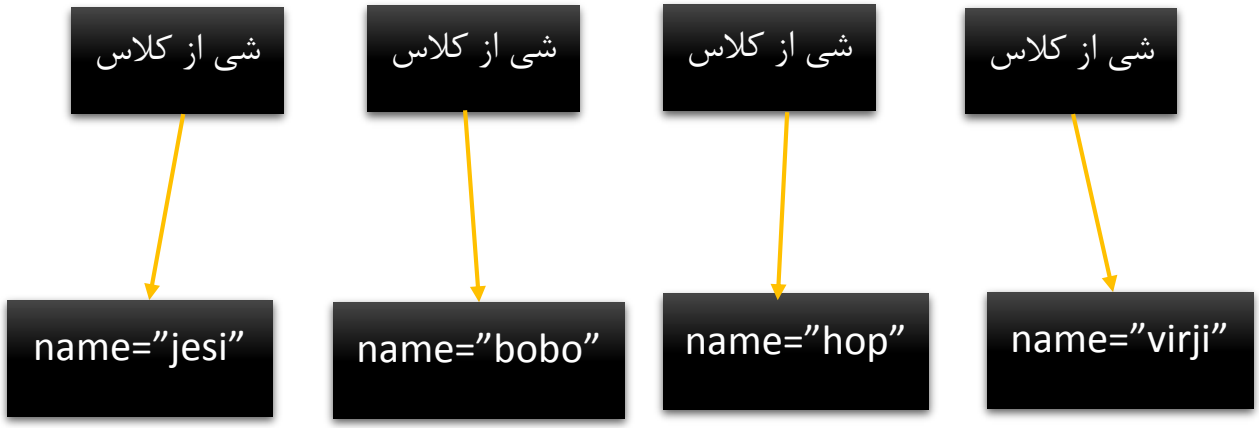


دسرسی اشیای ساخته شده از یک کلاس به متغیر استاتیک:

متغیر کلاس که غیر استاتیک اعلام شده



برای هر شی از کلاس یک متغیر جدا اختصاص داده شده است



- وقتی یک متغیر یا متد در کلاس استاتیک اعلام می شود دیگر نیاز نیست برای دسرسی به آن متغیر یا متد شی از کلاس ساخت تنها کافیست مستقیم با استفاده از نام کلاس و کاراکتر نقطه (.) به ان متغیر یا متد دسرسی پیدا کرد البته بدون نام کلاس هم می شود مستقیم تنها با بکار بردن نام متغیر از آن استفاده کرد اما در جاهایی که متغیر محلی هم نام متغیر نمونه کلاس خود داریم باید از نام کلاس برای صدا زدن متغیر یا متد استفاده کنیم:

- در مثال زیر متغیر و متد استاتیک یک کلاس را در متد main بدون شی ساختن و بکار بردن مستقیم نام کلاس صدا زده ایم.

```
package tutorial;

public class Cat {

	static String n;
	int age;

	static void eat() {
		System.out.println("cat Eating");
	}

	void sleep() {
		System.out.println("cat sleeping");
	}
	public static void main(String[] args) {

		n = "pishy";
		eat();
		System.out.println(n);

	}

}
```

خروجی:

```
cat Eating
pishy
```

- همان طور که مشاهده می کنید چون متد eat() و متغیر n استاتیک هستند دیگر برای صدا زدنشون نیاز به ساختن شی نیست.

حالا در پایین می خواهیم از متد sleep() استفاده کنیم چون استاتیک نیست باید با استفاده از شی ساخته شده از کلاس آن را صدا زد:

```
package tutorial;

public class Cat {

	static String n;
	int age;

	static void eat() {
		System.out.println("cat Eating");
	}

	void sleep() {
		System.out.println("cat sleeping");
	}
	public static void main(String[] args) {
    Cat cat=new Cat();
       cat.sleep();
        cat.age=6;
           n = "pishy";
           eat();
           System.out.println("name= "+n+" , "+"age= "+cat.age);

	}

}
```

خروجی:

```
cat sleeping
cat Eating
name= pishy , age= 6
```

و در مثال زیر چون از متغیر محلی همنام با متغیر نمونه n در متد main داریم دیگه واجب میشه که برای صدا زدن متغیر استاتیک از نام کلاس و نقطه(.) و اسم متغیر استفاده کنیم:

```
package tutorial;

public class Cat {

	static String n;
	int age;
```

```
    static void eat() {
        System.out.println("cat Eating");
    }
```

```
    void sleep() {
        System.out.println("cat sleeping");
    }
    public static void main(String[] args) {
   Cat cat=new Cat();
      cat.sleep();
       cat.age=6;
          n = "pishy";
           String n="bobo";

          eat();
          System.out.println("name= "+n);
          System.out.println("name= "+Cat.n);
    }

}
```

خروجی:

```
cat sleeping
cat Eating
name= bobo
name= pishy
```

```
n = "pishy";
           String n="bobo";
```

```
System.out.println("name= "+n);
          System.out.println("name= "+Cat.n);
```

همان طور که گفتیم اگر همزمان یک متغیر محلی همنام با متغیر نمونه استاتیک وجود داشت ناچار هستیم از نام کلاس برای صدا زدن متغیر استفاده کنیم.

مزایای متغیرهای استاتیک: در صرفه جویی حافظه برنامه شما موثر است.

در دو مثال زیر تفاوت متغیر استاتیک و غیر استاتیک را درک خواهیم کرد:

وقتی متغیر غیر استاتیک باشد:

```
package javalike;

class Counter {
	int count = 0;

	Counter() {
		count++;
		System.out.println(count);
	}

	public static void main(String args[]) {

		Counter c1 = new Counter();
		Counter c2 = new Counter();
		Counter c3 = new Counter();

	}
}
```

خروجی:

```
1
1
1
```

ما در این مثال متغیر نمونه count را غیر استاتیک تعریف کردیم و درون سازنده کلاس دستور افزایش یکی یکی این متغیر را پیاده سازی کردیم که با هربار شی ساختن این متغیر مقدارش افزایش پیدا کند اما چون با هر بار شی ساختن یک کپی از ویژگی ها و رفتار های غیر استاتیک کلاس برای شی ایجاد می شود، پس با هربار شی ساختن یک count با مقدار صفر برای شی درنظر گرفته می شود که در سازنده کلاس وقتی count++ می شود مقدارش یکی اضاف و به ۱ تغییر داده می شود و سپس در خروجی نمایش داده می شود.

وقتی متغیر استاتیک باشد:

```
package javalike;

class Counter {
	static int count = 0;
```

```
	Counter() {
		count++;
		System.out.println(count);
	}

	public static void main(String args[]) {

		Counter c1 = new Counter();
		Counter c2 = new Counter();
		Counter c3 = new Counter();

	}
}
```

خروجی:

```
1
2
3
```

- ما در این مثال متغیر نمونه count را استاتیک تعریف کردیم و درون سازنده کلاس دستور افزایش یکی یکی این متغیر را پیاده سازی کردیم که با هربار شی ساختن این متغیر مقدارش افزایش پیدا کند،چون متغیر استاتیک هست تنها یک حافظه به متغیر اختصاص داده می شود و همه اشیای ساخته شده از کلاس به این یک متغیر دسرسی دارند و اگر با یک شی مقدار این متغیر را تغییر دهیم مقدار متغیر برای سایر اشیا نیز تغییر می‌کند. و در اینجا وقتی c1 ساخته می شود مقدار count برابر ۱و با ساخته شدن c2 مقدار count برابر ۲ و در نهایت با ساخته شدن شی c3 مقدار count برابر ۳ می شود.

- متد ها نیز می توانند مانند متغیر ها استاتیک باشند اگر در ابتدای متد از کلمه کلیدی **static** استفاده کنیم.
- مانند متغیر ها اگر متد استاتیک باشد برای استفاده و صدا زدن متد نیاز به شی ساختن از کلاس نیست.
- متدهای استاتیک برای دسرسی به متغیر های استاتیک و دستکاری انها به کار می رود.

```
package tutorial;
public class Cat {
   static String name;
   int age;

   static void setName(String n) {
        name=n;

   }

   public Cat(){
```

```
        name="miuuuu";
    }
public void show(){

    System.out.println("name="+name);
}

    public static void main(String[] args) {


Cat c1=new Cat();

Cat.setName("pishi");
c1.show();


    }

}
```

خروجی:

```
name=pishi
```

```
static String name;
    int age;
```

تعریف دو متغیر یکی از نوع رشته و استاتیک و دیگری از نوع عدد صحیح و غیر استاتیک

```
static void setName(String n) {
        name=n;

    }
```

این متد که از نوع void و استاتیک می باشد ، یک رشته(String) می گیرد و متغیر رشته name که استاتیک هم می باشد مقدار دهی میکند.

```
    public Cat(){

        name="miuuuu";
    }
```

سازنده کلاس گربه که وقتی هر شی از این کلاس ساخته شد متغیر name را مقداردهی اولیه می کند.

```
public void show(){
```

```
    System.out.println("name="+name);
}
```

این متد مقدار متغیر name را در خروجی چاپ میکند.

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


Cat c1=new Cat();

Cat.setName("pishi");
c1.show();


    }
```

برای اجرای برنامه نیاز به متد main داریم که درآن یک شی از کلاس Cat ساخته ایم و در خط بعد چون متد setName استاتیک هست نیازی نیست با شی آن را صدا بزنیم فقط کافیست با نام کلاس ،نقطه(.) و نام متد آن را فراخوانی کنیم.

درپایان با متد show مقدار متغیر را در خروجی چاپ میکنیم.

یک نکته در مورد متدهای استاتیک:

متد های استاتیک نمی توانند در بدنه خود متغیر های غیر استاتیک را دستکاری کنند!!!

مثال:

```
package tutorial;

public class Cat {

	int age;

	public static void setAge() {

		age = 20;
	}

	public static void main(String[] args) {

		Cat.setAge();

	}
}
```

خروجی:

خطای کامپایل!!!

- **خطای کامپایل چیست؟** خطایی هست که در حین کد زدن در محیط Eclipse برای شما نشان داده می شود.
- **خطای زمان اجرا چیست؟**خطایی هست که وقتی برنامه خود را اجرا(run) می کنیم اتفاق می افتد.
- چون متغیر age غیر استاتیک هست و درون یک متد استاتیک دستکاری شده خطای کامپایل داده شده است.

سوال؟؟؟؟؟؟؟

چرا متد main استاتیک است؟

پاسخ: زیرا برای صدا زدن متد استاتیک نیاز به ساختن شی وجود ندارد،اگر متد main غیر استاتیک بود آن وقت با هر بار اجرا کردن برنامه خود jvm بیچاره مجبور بود ابتدا یک شی بسازد و بعد متد main را صدا بزند خب که چی؟!بزار شی بسازه اون که ادم نیست که خسته بشه؟!!!! درسته اما اگه همین جوری به شی ساختن ادامه بده با اجرای برنامه محل حافظه زیادی الکی الکی اشغال میشه.

سوال؟؟؟؟؟؟؟

میشه بدون متد main هم برنامه اجرا کرد؟

پاسخ:برای jdk های نخست میشد اما برای jdk7,8 نمی شود خطا می دهد!!!!

البته می شود متد main را پیاده سازی کرد اما دستورات خود را درون یک بلوک استاتیک قرار داد.

چطور؟پاسخ: ادامه آموزش را ببین!!

بلوک استاتیک در جاوا:

بلوک چیست؟

پاسخ: به دو آکولاد باز و بسته بلوک می گویند.

مثال:

{ }

خب بلوک استاتیک چیست؟

وقتی کلمه کلیدی **static** ابتدای این بلوک قرار بگیره میشه بلوک استاتیک!!!!

مثال: **static{ }**

کاربرد بلوک استاتیک: ما می توانیم بدون داشتن متد main (برای jdkهای پایین) دستورات برنامه خود را درون بلوک استاتیک اجرا کنیم!!! درصورتی که بلوک استاتیک ما به شکل زیر پیاده سازی شود:

```
static{
 System.exit(0);
}
```

که دستورات خود را قبل از `System.exit(0);` پیاده سازی میکنیم.

مثال:برنامه زیر برای JDK نسخه پایین جواب میده و برای JDK های ۷ و۸ خطا می دهد!!!

```
package tutorial;

public class Cat {

	int age;

	public  void setAge() {

		age = 20;
	}
	static{

		Cat c1=new Cat();
		c1.setAge();
		System.out.println("age="+c1.age);
		 System.exit(0);
	}


}
```

هشدار: این مثال برای jdk8 حتی ۷ خطا می دهد صرفا جهت اطلاع بود!!!! اما اگر این مثال را بصورت زیر پیاده سازی کنید خطا نمیده و برنامه اجرا می شود:

مثال:

```
package tutorial;

public class Cat {

	int age;

	public void setAge() {

		age = 20;
	}

	static {

		Cat c1 = new Cat();
		c1.setAge();
		System.out.println("age=" + c1.age);
	}

	public static void main(String[] args) {

	}

}
```

خروجی:

```
age=20
```

- همان طور که مشاهده میکنید در این مثال ما دستورات برنامه خود را در یک بلوک استاتیک قرار دادیم و ان را اجرا کردیم البته با کمک متد main
- نتیجه:در هر صورت برای اجرای برنامه به کمک متد main قلدر نیاز داریم!!!!!!!!حتی اگر دستورات خود را به جای main درون یک بلوک استاتیک قرار دهیم.

پیروز و موفق باشید

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.