آموزش زبان برنامه نویسی جاوا

شی و کلاس ها

جلسه نهم

نویسنده: رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!!

جاوا یک زبان برنامه نویسی شی گرا (Object-Oriented) می باشد.

ویژگی که زبان جاوا را قدرتمند و متمایز می کند همین ویژگی شی گرایی آن می باشد.

زبان شی گرایی جاوا مفاهیم زیر را دنبال میکند:

- چندریختی (Polymorphism)
- وراثت (Inheritance)
- کپسوله سازی (Encapsulation)
- انتزاع (Abstraction)
- کلاس ها (Classes)
- اشیا (Objects)
- نمونه (Instance)
- روش (Method)

❖ در این فصل نگاهی به مفاهیم کلاس ها و شی ها می اندازیم.

شی (object)

اشیا مثل شی های دنیای واقعی ما حالت ها و رفتار هایی دارند!!!!

منظور از حالت ها و رفتار ها چیه؟!!!!!!! با یک مثال به این دو مفهوم پی می بریم!!!!!

مثال: در دنیای واقعی هر چیزی یک شی در نظر گرفته می شود!!!!!!! مثل اتومبیل،موتورسیکلت،دوچرخه،خانه،اسلحه،هواپیما،جت،انواع حیوانات نظیر سگ،گربه،شیر،ماهی،مورچه و انسان ها و... شی حساب می شوند!!! درسته؟!!!!!!

خب حالا میخوایم به یکی از این شی ها مثلا سگ بپردازیم!!!

یک سگ حالت ها و رفتار های زیر را دارد!!!!!

حالت ها : هر سگ یک نام،رنگ،نژاد،سن،وزن،قد دارد.

رفتار ها: هر سگ واق واق میکند،غذا می خورد،آب می نوشید،حرکت می کند،می خوابد.

پس هر شی حالت ها و رفتار های مختص به خود را دارد.

- حالت ها: به تمام ویژگی های یک شی که قابل مقدار دهی باشد گفته می شود. مثلا ما می توانیم به سن سگ مقدار بدهیم یا به نژاد سگ مقدار بدهیم و بگیم از کدام نژاد می باشد.
- رفتار ها: به تمام اعمالی که یک شی می تواند انجام دهد رفتار می گوییم.مثلا سگ عمل حرکت کردن و راه رفتن را دارد،عمل غذا خوردن،عمل واق واق کردن و... را دارد.

❖ خب تا اینجا با حالت ها و رفتار های یک شی آشنا شدیم!!!

خب شی در برنامه نویسی چطور ساخته می شود؟!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

شی از کجا میاد؟!!!!!!!!!!!!!!!!!!

یادآوری:

یادتون میاد ما در بخش متغیر ها می گفتیم فلان متغیر از چه نوعی هست؟!!!

مثلا متغیر a از نوع عدد صحیح می باشد. یا متغیر b از نوع عدد اعشاری می باشد به مثال زیر توجه کنید:

مثال :

```
int a;
char b;
float c;
double d;
```

- متغیر a از نوع int (عدد صحیح) می باشد.
- متغیر b از نوع char (کارکتر) می باشد.
- متغیر c از نوع float (اعشاری) می باشد.

- متغیر d از نوع double (اعشاری) می باشد.

❖ توجه کرده باشید یک متغیر هنگام تعریف ابتدا نوعش رو مشخص کردیم!!!! یعنی هر متغیر یک نوعی دارد!!!!

❖ خب حالا اینا چه ربطی به شی داره؟!!!!!!!!! صبر داشته باشید!!!!!!!! ☺

❖ شی هم یک نوع هست!!

❖ **شی یک نوع از یک کلاس است!!!!!!!!!!!!**

یعنی چی؟!!!!! یعنی شما هنگام ساختن یا تعریف یک شی مثل سایر متغیر ها که نوعش رو مشخص و برای مثال int,char,double و....قرار می دادید شی را از نوع یک کلاس تعریف میکنید یعنی می توانید نوعش رو نام یک کلاس قرار دهید!!!!

مثال تعریف یک شی:

```
className      objectName
```

❖ خب میدونم هنوز مفاهیم واضع نشده!!!! نگران نباشید ادامه آموزش رو دنبال کنید یواش یواش مفاهیم روشن می شود!

کلاس (Class)

در جلسات اولیه آموزش تا حدودی در مورد کلاس و روش ساختن و ساختارش آشنا شدیم.

این که یک کلاس حاوی تمام دستورات برنامه ما می باشد.

حالا با نگاهی موشکافانه تر کلاس را بررسی میکنیم:

یک کلاس قالبی است که تمام حالت ها و رفتار های یک شی را می تواند داشته باشد.!!!!

صبر داشته باشید!!!!! ☺

اشیا در جاوا (Objects in Java)

اجازه بدهید نگاهی عمیق به اشیا بیاندازیم.

در دنیای واقعی پیرامون ما تعداد زیادی اشیا یافت می شود.ماشین ها ، سگ ها ، انسان ها و...

همه این اشیا حالت و رفتار خاص خود را دارند.

اگر ما به یک سگ نگاهی بیاندازیم حالت هایش : رنگ ، نژاد ، نام و...رفتارهایش : راه رفتن ، غذا خوردن ، واق واق کردن و...می باشد.

اگر ما شی نرم افزار را با یک شی در دنیای واقعی مقایسه کنیم مشاهده میکنیم که مشخصات هر دو شبیه به هم می باشد.

اشیا نرم افزار نیز مانند اشیا دنیای واقعی حالت ها و رفتار هایی دارند.

حالت های اشیا یک نرم افزار درون متغیر ها (fields) ذخیره می شود و رفتار های آن با متدها (methods) نمایش داده می شود.

کلاس ها در جاوا (Classes in Java):

تا حدودی با کلاس ها طریقه ایجاد کلاس و کد زنی در متد main کلاس و اجرای برنامه در جلسات قبل آشنا شدیم.

حالا میخوایم جزیی تر به کلاس ها بپردازیم:

- یک کلاس طرح یا نقشه منحصر به فرد اشیایی است که قرار بسازیم.
- اگر خواستیم یک شی از دنیای واقعی رو بسازیم ابتدا کلاسش رو می سازیم که این کلاس حاوی حالت ها و رفتار های آن شی در دنیای واقعی می باشد.
- همان طور که گفتیم حالت ها همان ویژگی ها و متغیر ها در کلاس می باشد و رفتار ها متد های درون کلاس.

بطور مثال اگر خواستیم یک سگ بسازیم!!!! بصورت زیر عمل میکنیم:

۱. ایجاد یک کلاس به نام سگ
۲. پیاده سازی ویژگی ها و متغیر ها (حالت های) سگ
۳. پیاده سازی رفتار ها(متد های) یک سگ

❖ یک نمونه از یک کلاس که در آن حالت ها(متغیرها) و رفتار ها(متدها) پیاده سازی شده در مثال زیر آورده شده است:

```
public class Dog{
        String breed;
        int age;
        String color;

        void barking(){
        }

        void hungry(){
        }

        void sleeping(){
        }
    }
```

- کلاس یک سگ را مشاهده میکنید که در بدنه کلاس ویژگی ها(حالت ها) و متد ها (رفتار های) آن پیاده سازی شده است.
- حالت ها(ویژگی هایش) نوع نژاد ، سن ، رنگ و رفتارهایش (متدها) واق واق کردن ،گرسنگی ،خوابیدن می باشد.

- String ها یا همون رشته ها(متن) رو در جلسات قبل یکم اشنایی هنگام چاپ بهش اشنایی داریم این رو بدونید که String (رشته) خود نوعی برای تعریف متغیر هست که یک جلسه مخصوص بهش میپردازیم.
- چون رنگ ها در قالب رشته می گنجند مثل blue یک متن یا کلمه هست باید از نوع String یا رشته تعریف شوند.
- سن هم که همه می دونیم یک عدد هست اینم عدد صحیح کسی نمی تونه سنش اعشاری باشه!!! به همین دلیل از نوع int تعریفش کردیم.
- اگر توجه کرده باشید این بار متغیر ها رو در بدنه کلاس و بلافاصله بعد از این که بلوک { کلاس باز شده تعریف کردیم ، پس میشود متغیر ها رو علاوه بر درون متد main مستقیم در بدنه کلاس هم تعریف کنیم که جلوتر بهش می پردازیم که چرا و به چه دلیل این کار را انجام دادیم.
- این سه متد هم سری کار برامون انجام میدن فعلا در حد این که نوعش که void هست و اسمش و محدودش مشخص کردیم ، جلوتر بهش می پردازیم.
- در پایان همه این دستورات در بدنه کلاس ما قرار دارند.

❖ یک کلاس شامل انواع متغیر های زیر است:

متغیر محلی(Local variables):

متغیر هایی که داخل متد ها (methods)،سازنده ها (constructors) یا بلوک ها تعریف می شوند متغیر محلی(local variables) نامیده می شوند.

نکته>> جلوتر به مفهوم سازنده ها (constructors) می پردازیم.

این متغیر ها در داخل متد ها تعریف و سپس مقدار دهی اولیه می شوند.

عمر و کارایی این متغیر ها تنها در محدوده بلوک {} متد ها می باشد و خارج از متد کارایی ندارند ، به همین دلیل بهشون محلی می گویند یعنی تنها بومی اون متد ها هستند و زبون هیچ کی جز افراد درون متد حالیشون نیست ☺

متغیر های نمونه (Instance variables):

به متغیر هایی که در داخل یک کلاس و بیرون متدها تعریف می شوند متغیرهای نمونه (Instance variables) گفته می شود.

این متغیر ها در بدنه کلاس می توانند مقدار دهی اولیه شوند.

متغیر های نمونه در داخل هر متد ، سازنده (constructors) ، بلوک ها و سایر کلاس ها قابل دسترسی می باشد.

متغیر های کلاس (Class variables)

بدون مقدمه چینی و به زبان ساده بهتون بگم که همون متغیر های نمونه هستند تنها با این تفاوت که قبل تعریف نوعشون از کلمه کلیدی **static** استفاده میکنیم.

مثال:

```
static int a=50;
```

- در جلسات آینده به تمامی کلمات کلیدی جاوا از جمله static می پردازیم.
- دوستان جاوا خیلی مفاهیم زیاد داره هر جلسه جز به جز تمرین کنید مثال حل کنید و کمی صبر هم چاشنی کار قرار دهید راحت می تونید یاد بگیرید مفاهیم رو چون هیچ برنامه نویسی یک روزه برنامه نویس نشده!!!!

❖ هر کلاس می تواند به هر تعداد ، انواعی از متد ها داخلش پیاده سازی شود.

سازنده (Constructors) :

سازنده (Constructors) نوع خاصی از متد(method) هست که برای مقدار دهی اولیه به شی مورد استفاده قرار می گیرد.

نقش سازنده ها (Constructors) جاوا:

- سازنده باید هم نام به اسم کلاس می باشد.
- سازنده بر خلاف متد نوع ندارد.مثلا متد نوع void و int و ....داشت اما سازنده هیچ نوعی ندارد.

پارامتر چیست؟

پارامتر همان متغیر های محلی می باشد که درون پرانتز (ورودی متدها و...) تعریف می شود.

مثال:

- پارامتر متد main ، متغیر درون پرانتز (String[] args) می باشد.

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

}
```

- پارامتر متد زیر (int a) از نوع عدد صحیح می باشد:

```
public void print(int a) {
System.out.println(a);
      }
```

- هر متد نوعی دارد که اینجا نوعش void می باشد و پارامتر ان int a که درون پرانتز جلوی متد می باشد و حاوی دستوراتی در بدنه خود که کار خاصی رو انجام می دهد.
- در مورد متد ها در جلسات آینده بیشتر صحبت خواهیم کرد.

Tip! هر کلاس می تواند چندین سازنده در آن پیاده سازی شود به شرطی که پارامتر های ورودی سازنده یکسان (شبیه) نباشد.

انواع سازنده ها (Constructors) جاوا

- سازنده پیشفرض (Default constructor) :

سازنده ای که پارامتر ندارد ،حتی اگر ما سازنده (constructor) رو پیاده سازی نکنیم خود برنامه ما بصورت خودکار آن را پیاده سازی می کند که سازنده پیش فرض نامیده می شود.

مثال : شکل پیاده سازی یک سازنده و اونم پیش فرض بصورت زیر است:

```
package iran;

public class Puppy {

	public Puppy() {
		// Default constructor. No parameter
	}
}
```

- یک کلاس تعریف کردیم به Puppy (بچه سگ)
- در بدنه کلاس یک سازنده پیاده سازی کردیم که هم نام با اسم کلاس می باشد.پس باید اسم سازنده هم نام با کلاس باشد.
- همان طور که مشاهده می کنید این سازنده نوعی ندارد
- در مورد کلمه کلیدی public و سایر کلمات کلیدی جاوا در یک جلسه مفصل صحبت خواهیم کرد.
- این سازنده مانند یک متد پرانتزی جلوی آن می باشد و هیچ پارامتر و ورودی ندارد زیرا سازنده پیشفرض می باشد.
- این سازنده مثل متد و ... یک {} باز و بسته دارد که محدوده ان را مشخص می کند.
- هر وقت خواستیم شی از کلاس Puppy بسازیم از این سازنده استفاده میکنیم.
- جلوتر روش شی ساختن از یک کلاس رو بررسی میکنیم.

- سازنده پارامتر دار(parameterized constructor):

همان طور که از اسمش پیداست سازنده ای (constructor) که بنا به نیاز بهش پارمتر می دهیم یعنی همون سازنده پیش فرض (Default constructor) هست با این تفاوت که متغیر یا پرامتر در ورودی خود دارد.

شکل پیاده سازی یک سازنده پارامتر دار در مثال زیر آمده است:

```
package iran;

public class Puppy {

	public Puppy(String name) {
		// This constructor has one parameter, name.
	}
}
```

- یک کلاس تعریف کردیم به Puppy (بچه سگ)
- در بدنه کلاس یک سازنده پیاده سازی کردیم که هم نام با اسم کلاس می باشد.پس باید اسم سازنده هم نام با کلاس باشد.
- همان طور که مشاهده می کنید این سازنده نوعی ندارد
- این سازنده مانند یک متد پرانتزی جلوی آن می باشد و دارای پارامتر و ورودی می باشد.
- این سازنده مثل متد و ... یک {} باز و بسته دارد که محدوده ان را مشخص می کند.
- هر وقت خواستیم شی از کلاس Puppy بسازیم از این سازنده استفاده میکنیم.

کلاس زیر دارای دو سازنده یکی پیش فرض و دیگری پارامتر دار می باشد:

```
package iran;

public class Puppy {
	public Puppy() {
	}

	public Puppy(String name) {
		// This constructor has one parameter, name.
	}
}
```

هر بار که شی ایجاد می شود حداقل به یک سازنده استناد می شود. به زبان ساده تر یعنی هر بار ما از یک کلاس شی می سازیم از یکی از سازنده هامون استفاده می کنیم!!!!

خلاصه ای کوتاه در مورد شی و کلاس:

- کلاس طرح و نقشه ما که حاوی حالت ها و رفتار های شی مورد نظر ما در دنیای واقعی می باشد.
- کلاس چیزی شبیه طرح و نقشه یک ساختمان می باشد که ویژگی و امکاناتی که یک ساختمان دارد رو در خود جا داده است. اما آیا می شود در یک طرح و نقشه ساختمان زندگی کرد؟!!!!
- برای این که بتوان در یک ساختمان زندگی کرد باید از روی نقشه ساختمان ،بنا را ایجاد کرد یعنی از روی نقشه و طرح ساختمان یک شی از ساختمان بسازیم که تمام ویژگی ها و امکانات ساختمان را برای زندگی داشته باشد.
- کلاس نیز مانند نقشه ساختمان هست و تنها با ایجاد یک از آن کلاس می توان از تمام ویژگی ها و حالت ها و رفتار و متد های آن استفاده کرد.
- پس لازمه استفاده از ویژگی ها ومتغیر ها و متدها و.... یک کلاس ایجاد شی از آن کلاس می باشد.
- در بدنه کلاس متغیر ها و متد ها قابل پیاده سازی می باشد.
- به متغیر هایی که داخل بدنه کلاس پیاده سازی می شود متغیر نمونه می گویند که رد تمامی محدوده کلاس و متد قابل استفاده می باشد.
- به متغیر هایی که داخل متد پیاده سازی می شود و محدوده استفاده آن تنها درون متد می باشد متغیر محلی می گویند.
- به متغیر هایی که هنگام تعریف ابتدای آن ها کلمه کلیدی static می اید متغیر کلاس می گویند.
- هر کلاس یک سازنده دارد که می تواند بدون پارمتر(پیش فرض) یا با پارامتر مورد نیاز ما باشد که هنگام ساختن شی از کلاس بهش نیاز پیدا میکنیم.

پس می توانیم یک اشیایی که در دنیای واقعی ما وجود دارد ، طرح و نقشه آن رو بصورت کلاس پیاده سازی کنیم و با ساختن شی از آن از تمام ویژگی ها و رفتارها و امکاناتش استفاده کنیم جلوتر روش ساختن شی از کلاس را بررسی میکنیم.

روش ایجاد یک شی از یک کلاس:

- همان طور که از قبل هم گفتیم یک کلاس طرح و نقشه یک شی می باشد.
- بنابراین یک شی از یک کلاس در جاوا ساخته می شود.
- در جاوا برای ایجاد یک شی جدید از یک کلاس از کلمه کلیدی **"new"** استفاده می شود.

گام‌هایی کوتاه برای ایجاد یک شی از یک کلاس بصورت زیر است:

برای ایجاد شی از یک کلاس تقریبا شبیه تعریف یک متغیر که ابتدا نوعش و سپس نامی براش انتخاب می کردیم هست فقط یک سری تفاوت هایی دارند!!!! به موارد زیر توجه کنید:

۱. ابتدا نام کلاس مورد نظر را تایپ می کنیم. (className)

۲. نامی برای شی مورد نظر انتخاب می کنیم.(objectName)

۳. تا اینجا مورد ۱ تا ۲ شبیه به تعریف متغیر می باشد که نام کلاس همان نوع شی هست و بعدش نامی به دلخواه انتخاب می کنیم که نام شی در نظر گرفته می شود.

۴. بعد از نام شی از علامت = (مساوی) استفاده می کنیم.=

۵. بعد ار علامت = (مساوی) از کلمه کلیدی new استفاده میکنیم. (new)

۶. بعد از کلمه کلیدی new دوباره نام کلاس رو تایپ میکنیم.(className)

۷. بعد از تایپ نام کلاس جلوی نام کلاس دو پرانتز باز و بسته میکنیم ()

۸. در پایان علامت ; بعد از پرانتز قرار می دهیم.

۹. مورد ۶ تا ۸ همان استفاده از سازنده های موجود در کلاس می باشد که اگر پرانتز خالی "()" باشد از سازنده پیش فرض استفاده کرده ایم:

```
new className();
```

و اگر از پرانتزی که اماده دریافت ورودی می باشد استفاده کرده باشیم یعنی سازنده پارامتر دار می باشد.

```
new className(a,b);
```

۱۰. موارد ۱تا ۸ شکل پیاده سازی شان بصورت زیر می باشد:

- ساختن یک شی از یک کلاس با سازنده پیش فرض:

```
className objectName = new className();
```

- ساختن یک شی از یک کلاس با سازنده پارامتر دار :

در این مثال کلی فرض شده است که دو متغیر صحیح به نام a , b داریم و قرار است به عنوان ورودی به سازنده خود که دارای دو پارامتر از نوع عدد صحیح می باشد داده شود:

```
int a=10;
int b=14;
```

```
className objectName = new className(a,b);
```

تا اینجا با کلاس و شی و روش ساختن آن ها اشنا شدیم به مثال های زیر در این رابطه توجه کنید:

```
package iran;
```

```
public class Puppy {

	public Puppy(int age) {
		// This constructor has one parameter, age.
		System.out.println(" age is :" + age);
	}

	public static void main(String[] args) {
		// Following statement would create an object myPuppy
		Puppy myPuppy = new Puppy(2);
	}
}
```

نام کلاس Puppy می باشد که در بدنه خود یک سازنده پارامتر دار و یک متد main دارا می باشد.

```
public Puppy(int age) {
		// This constructor has one parameter, age.
		System.out.println(" age is :" + age);
	}
```

یک سازنده تعریف کرده که طبق قانون همنام با اسم کلاس می باشد و پارامتر دار هست که پارامتر(متغیری که درون پرانتز جلوی نام آن تعریف می شود) آن یک متغیر از نوع صحیح به نام age می باشد. داخل سازنده یک دستور چاپ می باشد که هنگام ساختن شی از کلاس اگر از این سازنده استفاده شود دستور درون آن اجرا می شود.همان طور که گفتیم سازنده شبیه به متد عمل میکند با این تفاوت که نوعی ندارد و هیچ وقت مقداری را برنمیگرداند و تنها هنگام ساختن شی از یک کلاس هست که دستورات درون ان اجرا می شود.

```
public static void main(String[] args) {
		// Following statement would create an object myPuppy
		Puppy myPuppy = new Puppy(2);
	}
```

همان طور که می دونید برای اجرای دستورات و برنامه ما از متد main استفاده میکنیم.

خب اشیای خود را هنگام ساختن برای اجرا نیاز هست درون متد main پیاده سازی کنیم.

برای ساختن شی همان طور که مشاهده میکنید ابتدا نام کلاس بعد یک نام برای شی انتخاب شده و سمت راست مساوی از کلمه کلیدی new استفاده شده و بعد از آن سازنده کلاس رو پیاده سازی کرده ایم و چون پارامتر سازنده از نوع عدد int می باشد به دلخواه عدد ۲ را به عنوان ورودی بهش داده ایم.

بعد از run و اجرای برنامه متد main خط به خط اجرا می شود تا این که به خط ساختن شی می رسد و یک شی با نام myPuppy می سازد و حافظه ای بهش اختصاص می دهد و این شی چون سازنده ای دارد و داخل سازنده اش دستوری اجرا می شود بعد از اجرا سراغ سازنده میره و بلافاصله دستور درونش که اینجا چاپ مقدار متغیر age می باشد را اجرا میکند.

Tip! خب یک مثال از روش ساختن کلاس بصورت کد نویسی بررسی کردیم باز برای دید بهتر نسبت به کلاس و شی در جاوا سراغ دنیای واقعی می رویم!!! نگاه کنید مبحث شی گرایی جاوا یکی از مهم ترین قسمت های جاوا هست اگر شی گرایی جاوا را درک کردید و یادش گرفتید دیگه قسمت های دیگه جاوا یادگیریشون راحته چون پایه و اساس تمامی مفاهیمی که در اینده می خواهیم بررسی کنیم همین مبحث شی گرایی هست.

Tip! فرض کنیم در دنیای واقعی قصد داریم خودرو بنز را بسازیم!!!!!!!!!

- ابتدا تیم نقشه کشی و طراحی نقشه کامل و جامعی از این خودرو رو کاغذ پیاده سازی میکنند.
- تیم نقشه کشی و طراحی اطلاعات زیر را به ما می دهند:
- اونا میگن اگر ما میخواهیم یک خودرو رو بصورت کلی طراحی کنیم ویژگی ها و رفتار کلی این خودرو بصورت زیر است:

Tip!

- ویژگی یعنی حالت ها یعنی متغیر ها
- رفتار یعنی متدها
- در درک مسئله و پیاده سازی برنامه این دو خیلی به ما کمک میکند.

ویژگی‌ها: رنگ ، سرعت،شماره دنده،شماره گاز،مقدار موجودی بنزین،نوع مدل،قدرت موتور،وزن،طول،نوع بدنه و...

رفتارها: گاز دادن،ترمز کردن،کلاچ گرفتن،برف پاکن زدن،پخش کننده موسیقی،روشن کردن کولر،پریدن کیسه هوا و...

- حالا این ویژگی ها و رفتار ها فعلا در قالب طرح روی کاغذ می باشند برای اجرای شدن(خودرو شدن)(شی شدن) تحت عنوان خودرو بنز به کارخانه برای ساختن می دهند.
- در دنیای برنامه نویسی ساخت نرم افزار هم همین طور هست طرح و نقشه شما یک کلاس با نام خودروبنز می باشد که تمامی ویژگی ها و رفتار ذکر شده داخلش پیاده سازی می شود و یک سازنده که شبیه به کار خانه می تواند عمل کند و با ست کردن(مقدار دهی کردن) به پارامتر ها(ویژگی ها) به هنگام شی سازی یک شی خودرو بنز به ما تحویل می دهد که به تمام این امکانات (ویژگی ها و رفتار ها) دسرسی دارد یعنی می تواند رنگ دلخواه خود را انتخاب کند یا گاز بدهد ،تزمز بگیرد و....

حالا چطور می توانیم بعد از ساختن شی به این امکانات (رفتار ها و ویژگی ها)دسرسی پیدا کنیم؟!!!!

جواب»ادامه آموزش رو دنبال کن ☺

دسرسی به متغیر های نمونه و متد ها :

با ایجاد یک شی از کلاس تمامی متغیر های نمونه و متدهای آن کلاس قابل دسرس می باشند.

خب چطور؟!!!!

اینجا به لحظه حساس داریم نزدیک میشیم!!!!! :-((((((((((((((

آرامش خودتون رو حفظ کنید!!!! هیچ نگرانی وجود نداره!!!!!!

Tip! وقتی شما یک شی از یک کلاس می سازید به تمامی متغیر ها و متدهای این کلاس می توانید دسترسی داشته باشد.

```
/* First create an object */
ClassName ObjectReference = new Constructor();
```

- ClassName نام کلاس مورد نظر می باشد.
- ObjectReference همان شی از نوع کلاس می باشد.
- new Constructor(); همان سازنده کلاس که همنام با اسم کلاس می باشد که میتونه بصورت پیش فرض یا پارامتری با توجه به نیاز پیاده ساز شود که اینجا خلاصه شده نوشته Constructor.

Tip! حالا شما یک شی از کلاس به نام ObjectReference دارید.با این نام شی می توانیید به تمامی متدها و متغیر های نمونه کلاسی که ازش شی ساختید یا این شی که از نوع اون کلاس هست دسرسی پیدا کنید.خب چطوری؟!!داریم نزدیک می شیم!!!!

Tip! این شی که از کلاس ساختم (ObjectReference) فرض کنید شبیه نقش کنترل یک تلویزیون را داشته باشد!!!! و ویژگی ها و متد های کلاس نقش کانال های تلویزیون!!!!!همان طور که برای دسرسی به کانال های تلویزیون مثلا کانال ۱ نیاز هست کنترل را در دست بگیرید و با انگشت خود دکمه شماره ۱ را فشار دهید تا کانال شماره یک تلویزیون پخش شود!!!!!! برای دسترسی به متد ها و ویژگی های کلاس نیاز هست نام شی را در دست بگیریم و ویژگی ها و متدهایی که نقش دکمه روی کنترل را دارند را فشار دهیم تا بهشون دسترسی پیدا کنیم!!!!!!!!! خب چطور فشار دهیم؟!!!! با اشاره گر نقطه (.)(Point) که اشاره به ویژگی ها و متد ها میکند،خب باز چگونه این کار انجام می شود؟!!!!! بصورت زیر:

```
/* Now call a variable as follows */
//برای دسرسی که متغیر های نمونه کلاس
ObjectReference.variableName;

/* Now you can call a class method as follows */
// برای دسرسی به متد های کلاس
ObjectReference.MethodName();
```

برای دسترسی به متغیر های نمونه کلاس بصورت زیر عمل میکنیم:

- نام شی
- استفاده از نقطه(.)(point) برای اشاره کردن
- نام متغیر مورد نظر

```
ObjectReference.variableName;
```

برای دسترسی به متدهای کلاس بصورت زیر عمل میکنیم:

- نام شی
- استفاده از نقطه(.)(point) برای اشاره کردن
- نام متد مورد نظر

```
ObjectReference.MethodName();
```

به مثال زیر توجه کنید!!!!

```
package iran;

public class Dog {
	int age;
	String color;
	String race;
public Dog(){
		System.out.println("The dog was created");
			}
	public void barking() {
		System.out.println("vogh...vogh");
	}
	public void Sleep() {
		System.out.println("The dog Sleeping");
	}

	public void eat() {
		System.out.println("The dog Eating");
	}
	public static void main(String[] args) {
		Dog dogObject = new Dog();
		dogObject.age = 3;
		dogObject.color = "black";
		dogObject.race = "tazi";
		System.out.println(" dog is age= " + dogObject.age + " color= "
				+ dogObject.color + " Race= " + dogObject.race);
		dogObject.barking();

	}
}
```

خروجی:

```
The dog was created
 dog is age= 3 color= black Race= tazi
vogh...vogh
```

Tip! ما می توانید ویژگی ها و رفتار های یک کلاس برای درک و پیاده سازی بهتر بصورت یک جدول نمایش دهیم که به آن نمودار کلاس می گویند:

نام کلاس
ویژگی های کلاس
متد های کلاس

نام کلاس
نوع متغیر :نام متغیر + و...
نام متد+ و.....

Tip! نمودار کلاس مثال بالا Dog بصورت زیر است:

Dog
+age: int +color: String + race: String
+barking() +eat() +sleep()

Tip! کلاس Dog سری متغیر های نمونه و متد و سازنده در بدنه خود دارا می باشد.

Tip! کلاس Dog طرح و نقشه یک سگ می باشد که به ما داده اند و ما قصد داریم از این طرح و نقشه یک شی سگ بسازیم.

Tip! برای دسرسی به متغیر های نمونه و متدهای کلاس سگ نیاز هست شی از این کلاس ساخته شود.

```
Dog dogObject = new Dog();
```

برای ایجاد شی از کلاس در متد main دستور ساخت اولین شی سگ رو طبق دستورالعملی که یاد گرفتیم پیاده سازی میکنیم.

```
dogObject.age = 3;

//با استفاده از شی ساخته از کلاس به متغیر نمونه کلاس دسرسی پیدا کرده و مقداردهی به آن کردیم

dogObject.color = "black";

//با استفاده از شی ساخته از کلاس به متغیر نمونه کلاس دسرسی پیدا کرده و مقداردهی به آن کردیم.

dogObject.race = "tazi";

//با استفاده از شی ساخته از کلاس به متغیر نمونه کلاس دسرسی پیدا کرده و مقداردهی به آن کردیم.

System.out.println(" dog is age= " + dogObject.age + " color= "
                   + dogObject.color + " Race= " + dogObject.race);

//با استفاده از شی ساخته از کلاس به متغیر نمونه کلاس دسرسی پیدا کرده و مقدار آنها را چاپ کردیم.

dogObject.barking();

//با استفاده از شی ساخته از کلاس به یکی از متدهای کلاس دسرسی پیدا کرده وآن را اجرا کردیم.
```

شی dogObject که به کنترل تلویزیون تشبیه اش کردیم در اختیار داریم حال نقطه (.) که حکم فشردن یا اشاره کردن را در کنترل برای ما دارد و دکمه های کنترل را که به متغیر های نمونه و متدها تشبیه اش کردیم رو در اختیار داریم و با آن می توانیم متغیر های نمونه کلاس مقدار دهی کنیم یا مقدار آن ها را چاپ کنیم و همچنین می توانیم به متد های کلاس دسرسی پیدا کنیم و آن ها را اجرا کنیم.

حال بصورت تصویری نحوه عملکرد شی ساخته شده از کلاس را بررسی میکنیم:

طرح و نقشه (کلاس) Dog

- بدنه کلاس:

ویژگی های کلاس

```
int age;
String color;
String race;
```

متدهای کلاس

```
public void barking() {
    System.out.println("vogh...vogh");
}
public void Sleep() {
    System.out.println("The dog Sleeping");
}

public void eat() {
    System.out.println("The dog Eating");
}
```

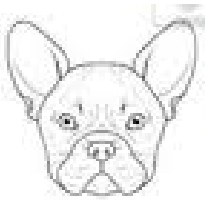
Tip! حالا ما قصد داریم یک شی سگ از این طرح و نقشه(کلاس) بسازیم:

برای این کار باید بریم سراغ کارخانه متد main در کلاس تا بتوانیم این طرح و نقشه را به اجرا در آوریم!!!!

باید یک شی از نوع طرح و نقشه (کلاس) Dog بسازیم در این کارخانه!!!!

```
Dog dogObject = new Dog();
```

شی dogObject بشکل زیر بصورت خام تولید می شود که تمامی ویژگی ها و رفتار های درون طرح و نقشه(کلاس) را دارا می باشد.

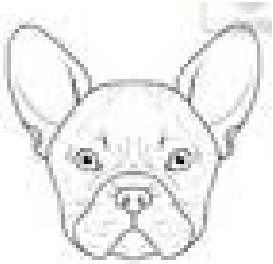


dogObject

نام شی سگ ما dogObject است!!!!! حالا باز در کارخانه نیاز هست به این سگ خام خود سن و رنگ و اسم و نژاد بدهیم!!!!!

- برای ثبت(مقدارهی به) نام و رنگ و نژاد سگ خام خود بصورت زیر عمل می کنیم!!!!!!

- قبل از مقدار دهی به ویژگی هامون شناسنامه سگ dogObject عزیز ما بصورت زیر است!!!!!!



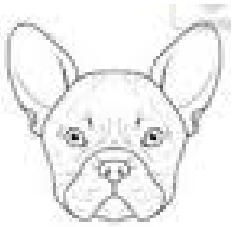
شناسنامه (ویژگی ها)	
race	خالی
age	خالی
color	خالی

dogObject

- نمیشه سگ مون رو بدون شناسنامه بگذاریم خب!!!! پس قشنگ براش شناسنامش رو پر (مقدار دهی به ویژگی ها) می کنیم:

- ابتدا به age آن مقدار ۳ میدیم:

شی dogObject شبیه به کنترل تلویزیون و ویژگی ها و رفتار آن شبیه به دکمه های کنترل تلویزیون عمل می کند،کافیست با نام شی و با استفاده از نقطه(.)به ویژگی ها و متدهای آن دسرسی پیدا کنیم(این دکمه ها(ویژگی ها و رفتارها) را فشار دهیم)!!!



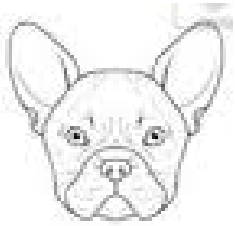
dogObject.age = 3;

dogObject.age

در پنج گام زیر می توانید با یک شی دسرسی به متغیر های نمونه و متدها داشته باشیدو آن ها را مقدار دهی کنید:

```
1) dogObject
2) dogObject.
3) dogObject.age
4) dogObject.age =
5) dogObject.age = 3;
```

- بقیه موارد دسترسی به متغیر و مقدار دهی به ان ها شبیه به مراحل ۱ تا ۵ می باشد.
- به color ان مقدار black می دهیم:



```
dogObject.color = "black";
```


```
dogObject.color
```

Tip!

این جعبه مشکی رنگ بالا که به دو بخش متغیر های نمونه و متدها تقسیم شده همان شی شما می باشد که تمام ویژگی ها و کلاسی که ازش ساخته شده در بر دارد و تنها با یک نقطه (.) می توانید به این ویژگی ها و متد ها درسی پیدا کنید.

در پنج گام زیر می توانید با یک شی دسترسی به متغیر های نمونه و متدها داشته باشید و ان ها را مقدار دهی کنید:

```
1) dogObject
2) dogObject.
3) dogObject.color
4) dogObject.color =
5) dogObject.color = "black";
```

- به race ان مقدار tazi می دهیم:

```
dogObject.race = "tazi";
```

dogObject.race

Tip!

فرض کنید شی ای که از نوع یک کلاس ساخته اید جعبه ای می باشد که تمام ویژگی ها و رفتارهای آن کلاس را در خود دارد و شما می توانید با نقطه(.) به این ویژگی ها و رفتار ها اشاره کنید و بهشون مقدار بدید یا بخواهید یکی از اعمال موجود(متدها) رو انجام دهد.

در پنج گام زیر می توانید با یک شی دسرسی به متغیر های نمونه و متدها داشته باشید و آن ها را مقدار دهی کنید:

```
1) dogObject
2) dogObject.
3) dogObject.race
4) dogObject.race =
5) dogObject.race = "tazi";
```

Tip!

حالا سگ ما(شی سگ) شناسنامه دار شد و به شکل زیبای زیر دراومد!!!!!!!!

dogObject

شناسنامه (ویژگی ها)	
race	tazi
age	3
color	black

حالا قصد داریم شناسنامه(مقدار متغیر های) شی سگ خود را چاپ کنیم!!!!!!

```
System.out.println(" dog is age= " + dogObject.age + " color= "
                    + dogObject.color + " Race= " + dogObject.race);
```

تنها با استفاده از شی خود و نقطه(.) و اشاره کردن به متغیر مورد نظر می توانیم به مقدار آن دسرسی پیدا کنیم و چاپش کنیم.

حال برای اشاره کردن به متدهای درون شی نیز می توان همانند اشاره کردن به متغیر ها ،متدها را نیز فراخوانی کنیم و عملی را که انجام می دهند را به اجرا در آوریم:

```
dogObject.barking();
```


```
dogObject.barking();
```

در سه گام زیر می توانید با یک شی دسرسی به متغیر های نمونه و متدها داشته باشید:

1) dogObject
2) dogObject.
3) dogObject.barking();

با روش ساختن شی از یک کلاس و طریقه دسرسی و اشاره کردن به ویژگی ها و رفتار آن آشنا شدیم.

می توانیم n تا (بی شمار) شی از یک کلاس بسازیم و به ویژگی ها و رفتار آن را دسرسی داشته باشیم.

جالب است بدانید هر بار که شی از یک کلاس ساخته می شود تمام ویژگی ها و رفتار هایی که کلاس دارد برایش در نظر گرفته می شود که می تواند مقدار این ویژگی ها با شی های قبل متفاوت باشد برای مثال اگر شما ۱۰ شی از کلاس Dog مذکور صفحات قبل بسازید ۱۰ شی سگ داریم که می تواند ۱۰ رنگ و... متفاوت داشته باشد مثلا سگ اول رنگ مشکی سگ دوم رنگ سفید سگ سوم رنگ قهوه ای سگ چهارم رنگ زرد و.....یعنی اگر به ویژگی رنگ سگ اول مقدار دادیم دیگر ربطی به رنگ سایر سگ ها ندارد و تمام ویژگی تنها برای اون شی مذکور تغییر داده می شود.!!!!!!!!!!

در مثال زیر سه شی از کلاس

سگ ایجاد و مقدار دهی شده است:

```
package iran;

public class Dog {

 int age;
 String color;
 char jensiat;
 String name;

 public Dog() {

        System.out.println("The dog was created");

 }

 public void barking() {

        System.out.println("vogh...vogh");

 }

 public void Sleep() {

        System.out.println("The dog Sleeping");

 }

 public void eat() {
        System.out.println("The dog Eating");
 }

 public static void main(String[] args) {
        Dog dog1 = new Dog();
        Dog dog2 = new Dog();
        Dog dog3 = new Dog();
        // ==========================
        dog1.name = "Roopi";
        dog1.age = 4;
        dog1.color = "white";
        dog1.jensiat = 'f';
```

```
        // ============================
        dog2.name = "Raki";
```

```
                dog2.age = 2;
                dog2.color = "black";
                dog2.jensiat = 'm';
                // ============================
                dog3.name = "BoBo";
                dog3.age = 1;
                dog3.color = "Brown";
                dog3.jensiat = 'm';
                // ============================
                System.out.println(dog1.name + " " + dog1.age + " " + dog1.color + " "
                        + dog1.jensiat);
                // ================================
                System.out.println(dog2.name + " " + dog2.age + " " + dog2.color + " "
                        + dog2.jensiat);
                // ================================
                System.out.println(dog3.name + " " + dog3.age + " " + dog3.color + " "
                        + dog3.jensiat);
                // ================================

        }

    }
```

خروجی:

```
The dog was created
The dog was created
The dog was created
Roopi 4 white f
Raki 2 black m
BoBo 1 Brown m
```

- همان طور که در دنیای واقعی می شود از روی یک طرح و نقشه ساختمان چندین ساختمان ایجاد کرد یا از روی طرح و نقشه یک خودرو چندین خودرو تولید کرد مثلا از روی طرح اتومبیل بنز می شود بیشمار خودرو بنز با رنگها و ویژگی های متفاوت تولید کرد در برنامه نویسی شی گرایی هم همین طور می باشد و می شود از یک کلاس چندین شی ساخت که همگی از ویژگی ها و رفتار های همان کلاس استفاده می کنند با این تفاوت که ویژگی های هر شی مقدارش یا ویژگی های شی های دیگر ساخته شدن از کلاس متفاوت است.
- همان طور که در مثال مشاهده میکنید ما سه شی از کلاس Dog ساختیم که همگی از متغیر های کلاس Dog استفاده میکنند یعنی نام متغیر ها یکسان می باشد اما هر شی در حافظه خود مقداری متفاوت می تواند به این متغیر ها بدهد.
- هر شی هنگام ساخته شدن، سازنده آن اجرا می شود و دستور موجود در سازنده بلافاصله بعد از ایجاد شی اجرا می شود.
- یکی از مزیت های خوب شی گرایی و شی ساختن این است که از کدهای تکرای جلوگیری کرده است.مثلا اگر در برنامه خود نیاز به ۱۰۰ سگ دارید بجای نوشتن ۱۰۰ بار کد و دستور مشابه برای ۱۰۰ سگ تنها کافیست یک بار کلاسی که تمام ویژگی

ها و رفتار مورد نیاز شما را دارا می باشد بنویسید و سپس ۱۰۰ شی از این کلاس ایجاد کنید و این باعث می شود کد برنامه نویسی شما خلاصه تر و و اجرای برنامه شما سریع تر شود.

- شاید بگید ۱۰۰ شی ساختن یعنی ۱۰۰ بار تعریف کردن یک شی از کلاس چقدر میتونه خسته کننده باشه؟!!!!! اما صبر کنید در جلسات آموزشی آینده با سری دستورات و ساختمان داده ها برای ذخیره اطلاعات و مقدار ها اشنا خواهید شد که این کار را برای شما خیلی آسان می کند.

```
package iran;
```

- کلاس ما درون پکیج iran قرار دارد.

یادآوری: بریا نظم و مرتبا سازی و فهرست بندی و دسته بندی کلاس ها برای دسرسی سریع تر به کلاس های خود از پکیج بندی در جاوا استفاده میکنیم.

```
public class Dog {
```

- نام کلاس که با حرف بزرگ شروع شده و قبل آن کلمه کلیدی public و class استفاده شده است.

```
int age;
String color;
char jensiat;
String name;
```

- تعریف متغیر های نمونه در بدنه کلاس، این متغیر های در تمامی محدوده کلاس و متغیر ها و... قابل دسرس می باشند.

```
public Dog() {

        System.out.println("The dog was created");

}
```

- سازنده کلاس که با هربار شی ساختن از کلاس این سازنده و دستورات درون آن اجرا می شود.(این یک سازنده پیش فرض است)

```
public void barking() {

        System.out.println("vogh...vogh");
```

```
}
```

- یک متد که مقداری را برنمی گرداند و تنها دستورات درون خود را اجرا میکند.
- یک متد یک نوع و پارامتری می تواند داشته باشد و دستور مورد نیاز ما را چاپ میکند.
- متدها بیشتر برای نظم بخشیدن به برنامه نویسی ایجاد می شوند.
- در جلسات آینده مفصل در مورد متدها و صحبت خواهیم کرد.

```
public void Sleep() {

        System.out.println("The dog Sleeping");

  }

  public void eat() {
        System.out.println("The dog Eating");
  }
```

- سایر متد های کلاس که می توان با فراخوانی آن ها در متد main کلاس ، دستورات درون آن ها را به اجرا دراورد.

```
public static void main(String[] args) {
Dog dog1 = new Dog();
Dog dog2 = new Dog();
Dog dog3 = new Dog();
```

- در متد main کلاس سه شی از یک کلاس ایجاد کردیم که بعد از ساخته شدن سازنده ها فراخوانی و تمام دستورات درون سازنده به اجرا در می آید.و این اشیا به تمامی ویژگی ها و رفتار های کلاس Dog دسرسی پیدا میکنند چون نوعی از کلاس Dog هستند.

```
dog1.name = "Roopi";
dog1.age = 4;
dog1.color = "white";
dog1.jensiat = 'f';
```

- با نام شی و نقطه (.) می توانیم به ویژگی ها و رفتار آن دست پیدا کردن و تغییرات خود را اعمال کنیم.جالب است که این تغییرات در مقدار متغیر ها با متغیر های سایر اشیا یاخته از از این کلاس متفاوت است یعنی همگی از متغیر های یکسان اما با مقدار متفاوت استفاده میکنند زیرا هربار که شی از یک کلاس ساخته می شود یک حافظه برای تمام ویژگی های آن در نر گرفته می شود.

```
dog2.name = "Raki";
```

```
dog2.age = 2;
dog2.color = "black";
dog2.jensiat = 'm';
// ============================
dog3.name = "BoBo";
dog3.age = 1;
dog3.color = "Brown";
dog3.jensiat = 'm';
```

- مثل دستور مشابه قبل مقدار دهی کردن سایر اشیای ساخته شده از یک کلاس.

```
System.out.println(dog1.name + " " + dog1.age + " " + dog1.color + " "
+ dog1.jensiat);
// ================================
System.out.println(dog2.name + " " + dog2.age + " " + dog2.color + " "
+ dog2.jensiat);
// ================================
System.out.println(dog3.name + " " + dog3.age + " " + dog3.color + " "
+ dog3.jensiat);
```

- در پایان مقادیر متغیر های اشیای ما چاپ می کنیم و همان طور که مشاهده میکنید با وجود استفاده اشیا از متغیر های یکسان اما مقادیر متفاوت می باشد.

در این جلسه آموزش با بخشی از شی گرایی جاوا یعنی کلاس و شی ساختن و جزییات مربوط به آن آشنا شدیم دیدیم که با شی ساختن از کلاس از بسیاری از کد های تکرای جلوگیری شد قبول این مبحث سنگین بود!!!!!! یکی از مهم ترین مباحث برنامه نویسی جاوا بود که سعی کردم به ساده ترین شکل بصورت تصویری و همراه بامثال بیانش کنم حتما حتما تمرین کنید مثال حل کنید صبر داشته باشید با مثال های کوچک و ساده شروع کنید اگر مفاهیم رو خوب یاد نگرفتید هیچ اشکالی نداره دوباره مرور کنید و تمرین مطمئن باشید با صبر و حوصله و انگیزه برای یاد گیری حتما موفق می شوید.

- با این آموزش خیالم راحت شد چون دیگه دستم باز شد که مفاهیم پیشرفته تری از جاوا را به شما اموزش بدم :-)))))))))

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.