آموزش زبان برنامه نویسی جاوا

چندریختی را بهتر یادبگیریم

جلسه بیست و چهارم

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!!

جلسه قبل در مورد چندریختی صحبت کردیم برای درک بهتر این مبحث مهم شی گرایی جلسه ای دیگر را نیز به چندریختی اختصاص داده ایم.قبل از مطالعه این جلسه جلسه قبل یعنی چندریختی در جاوا را مطالعه کنید.

پلی مورفیسم یا چندریختی یکی از مفاهیم شی گرایی در جاواست.این مفهوم تا حدودی دشوار و گیج کننده است!!! به همین خاطر یک جلسه دیگه هم برای بررسی این مفهوم چغر و بد بدن قرار داده ایم :-)

دلیل گیج کننده بودن این مفهوم این است که نسبت به سایر مفاهیمی که تا الان بررسی کردم انتزاعی است،با این حال شما میتونید در پلی مورفیسم استاد بشید و برآن مسلط بشید و به کمر خاکش کنید ☺

برای یادگیری بهتر چندریختی یا هر مفهوم جاوای دیگر باید تمرین و مثال عملی زیادی حل کنید تا یواش یواش مفهومش دستتون بیاد.چون امکان داره توضیحات تئوری جز گیج تر کردن شما دست آورد دیگه ای نداشته باشه.

بیایید صدای گربه را بررسی کنید!!!!! گربه یک عمل صداکردنی داره و صدای آن میو میو...!!!! هستش سگ هم صدا داره و عمل صدا کردن سگ واق واق..... هستش!!!! پس قبول میکنید که هر حیوانی یک **عمل صدا کردنی** دارد که بین همه ی آن ها مشترک هستش اما با **صدایی متفاوت**؟!!!!!!!!!!!!!!!!

عمل صدا کردن را یک **متد** به نام makeSound() در نظر میگیریم .

تفاوت صدا گربه با سگ و سایر حیوانات را **دستور** درون متد در نظر میگیریم.

یعنی یک متد که نام مشترکی برای همه حیوانات دارد اما دستوری (تفاوت صدا) متفاوت.

مثلا متد makeSound() را برای حیوانات بصورت زیر بررسی میکنیم:

گربه

```
public void makeSound(){

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

}
```

سگ

```
public void makeSound(){

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

}
```

مرغ

```
public void makeSound(){

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

}
```

در سه مثال بالا همان طور که مشاهده میکنید سه متد با نام یکسان (عمل صدا کرد) اما دستوری متفاوت (تفاوت در صدای حیوانات) در بدنه خود دارند.یعنی همه حیوانات `makeSound()` (عمل صدا کردن) را دارند اما دستوری متفاوت (صدایی متفاوت) با توجه به نوع حیوان به نمایش میزارند.

قبول دارید که مرغ و خروس و سگ و گربه و.... همه حیوان هستند اما در شکل و قالبی متفاوت. یعنی گربه حیوانی است که در شکل گربه ظاهر شده یا مرغ حیوانی است که در شکل مرغ ظاهر شده یعنی نوع همه آنها حیوان است اما در شکل گربه و مرغ و خروس و... ظاهر شده اند.چندریختی یا پلومورفیسم هم این مفهوم را بیان میکند،چندریختی میگه یک حیوان میتونه در شکل های گوناگون ظاهر شود.مثلا یک حیوان متد `makeSound()` و عمل صدا کردن را دارا می باشد.حالا این متد با توجه به نوع حیوان عمل (صدای) متفاوتی را به نمایش میگذارد.

در چندریختی تمرکز ما روی متدهای مشترک با دستورات متفاوت می باشد.

مثل متد صدا کردن که بین همه حیوانات مشترک هست اما برای هر حیوان دستور صدا متفاوت است.

در چندریختی مثلا وقتی کلاس حیوان میخواد در شکل کلاس گربه،کلاس سگ،کلاس مرغ ظاهر شود متدهایی نظیر عمل صدا کردن که بین گربه و سگ و مرغ مشترک هست در همه این کلاس ها override (بازنویسی) می شود.یعنی متد صدا کردن که درون کلاس حیوان هست با توجه به نوع صدا و دستور هر حیوان درون کلاس های گربه و سگ و مرغ override می شود.

کلاس Animal را در زیر مشاهده میکنید که متد makeSound در بدنه آن پیاده سازی شده است که یک دستور کلی را اجرا میکند. این یعنی همه حیوانات یک عمل صدا کردنی را دارند اما نوع صدا کردن باتوجه به نوع حیوان تغییر میکند.خب حالا ما میخواهیم متد makeSound() در کلاس Animal صدای گربه را بدهد. برای این کار باید یک کلاس تعریف کنیم به نام Cat که کلاس Animal را به ارث می برد. متد makeSound کلاس Animal را در کلاس Cat override (بازنویسی) میکنیم یعنی همون متد با همون نام با همون پارامتر و با همان نوع اما بادستوری متفاوت را در کلاس فرزند یعنی Cat پیاده سازی میکنیم.

```
class Animal {
	public void makeSound() {

		System.out.println("animal  sounds");
	}

}

class Cat extends Animal {
	public void makeSound() {

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

}
```

- ما برای این که متد makeSound() کلاس Animal صدای گربه بدهد یعنی کلاس Animal در زمینه صدا کردن به شکل گربه دربیاد متد makeSound را درون کلاس Cat ، override کرده ایم.
- حالا ما باید کاری کنیم که وقتی یک شی از کلاس Animal ایجاد کردیم اون شی زمانی که متد makeSound را صدا میزند بجای پیام "animal sounds" پیام "miu miu" را در خروجی چاپ کند.یعنی شی کلاس Animal به شکل شی کلاس Cat ظاهر شود. یعنی وقتی شی کلاس Animal میخواد عمل صدا کردن را انجام دهد بگه "miu miu"خب چطور باید این کار را انجام بدیم؟

پس رسالت این است که ما کلا با شی کلاس پدر یعنی Animal سروکار داریم و کلاس های فرزند نظیر کلاس گربه که کلاس حیوان را به ارث برده است در عمل فقط یک حالت انتزاعی و پنهان هستند و ما تنها طرف حسابمون کلاس و شی پدر هستش.

قبل از این که ادامه این مثال را بررسی کنیم بزارید تا یک مثال از دنیای واقعی برای درک این موضوع برای شما بگم:

ایا شما پرنده طرقه را میشناسید؟طرقه پرنده ای خوش آواز هستش که اواز آن تقلیدی از آواز سایر پرندگان می باشد.یعنی طرقه گاهی به شکل بلبل درمیاد و اواز بلبل را سر میدهد گاهی شبیه قناری ظاهر میشه و مثل قناری میخواند گاهی به شکل پرنده ای دیگر در می اید و میخواند.

پرنده طرقه شبیه به گنجشک و خاکستری رنگ می باشد. ما در آواز خوانی طرقه هیچ ظاهری و شمایلی از قناری خوش رنگ و بلبل و .... نمی بینیم گویی قناری و بلبل و... در کالبد طرقه پنهان شده اند و با وجود این که صدای سایر پرندگان را میشنویم اما طرف حساب ما تنها پرنده طرقه هست. یعنی میتوان گفت پرندگان دیگر طرقه را به ارث برده اند و طرقه پدر آنها هستش و متد آواز خوانی طرقه در بدنه همه کلاس های فرزندان بازنویسی شده و طرقه میتونه از این طریق به شکل هر کدوم از پرنده ای که دوست دارد ظاهر شود و اواز بخواند. در چندریختی هم دقیقا همین طور هست همه فرزندان کلاس پدر در کالبد پدر پنهان شده اند و پدر با دستور خاص خود به شکل هر فرزندی که دوست داشت درمی آید. تغییر شکل کلاس پدر به شکل کلاس فرزند در قالب متد رخ می دهد. خب ما در پلی مورفیسم هم تنها با شی کلاس پدر سرو کار داریم و شی کلاس پدر تنها به تمام ویژگی ها و رفتار های کلاس خودش دسرسی دارد با این تفاوت که هر متدی از کلاس پدر که در کلاس فرزند override یا بازنویسی شده هنگام صدا زدن آن به جای اجرای دستور متد پدر دستور متد فرزند اجرا می شود. مثلا متد makeSound() که درون کلاس Animal هستش و دستور "Animal Sounds " را در خروجی چاپ میکند اگر این متد در کلاس یکی از فرزندان مثلا کلاس Cat، override یا بازنویسی شود و دستور درون این متد در کلاس Cat به چاپ پیام "miu miu" تغییر کند.شی ای که به روش چند ریختی (پلی مورفیسم) از کلاس پدر یعنی Animal ایجاد میشود هنگام صدا زدن متد makeSound() بجای چاپ پیام `"animal sounds"` که درون کلاس پدر یعنی Animal قرار دارد دستور چاپ پیام `"miu miu"` این متد که درون کلاس فرزند یعنی Cat قرار دلرد را اجرا میکند.دقیقا شبیه پرنده طرقه که ما در ظاهر با طرقه سر و کار داریم اما طرقه آواز قناری را میخواند.

خب تا اینجا شما بصورت تئوری با مفهوم چندریختی آشنا شدید حالا میخواهیم کد مثال بالا که شی حیوان میخواد به شکل شی گربه دربیاد را بصورت عملی پیاده سازی کنیم:

در مثال زیر شی سازی حالت عادی که از کلاس ها ایجاد میکردیم را مشاهده میکنید:

```
package polymorphism_JavaPro;

class Animal {
	public void makeSound() {

		System.out.println("animal  sounds");
	}

	public void eat() {

		System.out.println("animal  eat");
	}
}

class Cat extends Animal {
	public void makeSound() {

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

}

public class Test {

	public static void main(String[] args) {
		Animal a1=new Animal();
		a1.eat();
		a1.makeSound();

	}
}
```

خروجی:

```
animal  eat
animal  sounds
```

- در این مثال یک شی به نام a1 از کلاسAnimal ایجاد کرده ایم.
- شی a1 به تمام ویژگی ها و رفتارهای کلاس Animal دسرسی دارد.
- از طریق شی a1 متدهای makeSound() و eat() را صدا زده ایم.
- همان طور که مشاهده میکنید دستورات اجرا شده درون این دو متد کاملا متعلق به کلاس Animal هستش.

- حالا قصد داریم کلاس Animal در شکل کلاس Cat ظاهر شود مثلا وقتی شی ای از کلاس Animal ساختیم آن شی صدای گربه را دربیارد. برای این کار کلاس Cat باید کلاس Animal را به ارث ببرید و متدی (عمل صدا کرد) makeSound() کلاس Animal را درون کلاس Cat بازنویسی و override کنیم. حالا وقت شی ساختن از کلاس Animal میرسه که این شی همان طور که گفتیم تنها و تنها و تنها!!!!!!! به متغیرها و متدهای کلاس پدر یعنی Animal دسرسی دارد.یعنی مثل روان عادی که شی یک کلاس میتونه به متغیرها و متدهای کلاسش دسرسی پیدا کند و فقط تنها تفاوت این است که شی کلاس Animal هنگام صدا زدن متدهایی از کلاس Animal که در کلاس فرزندش یعنی Cat ، override شده بجای اجرای دستورات خود متد پدر ،دستور override شده اجرا می شود:
- خب ما تا اینجا کلاس Animal و کلاس Cat را ایجاد کرده ایم و کلاس Cat کلاس Animal را به ارث برده است و متد makeSound درون کلاس Cat، override شده است حالا وقت شی ساختن از کلاس Animal به روش چندریختی هست برای این کار در جلسه قبل شی ساختن یک کلاس بصورتی که در شکل شی کلاس دیگر ظاهر شود را یادگرفیتیم اما دوباره مروری میکنیم:

سازنده کلاس فرزند + new + = + نام شی + نام کلاس پدر

شی ساختن به روش چندریختی یعنی وقتی شی کلاس پدر میخواد در شکل شی کلاس فرزند ظاهر شود بصورت بالاست.

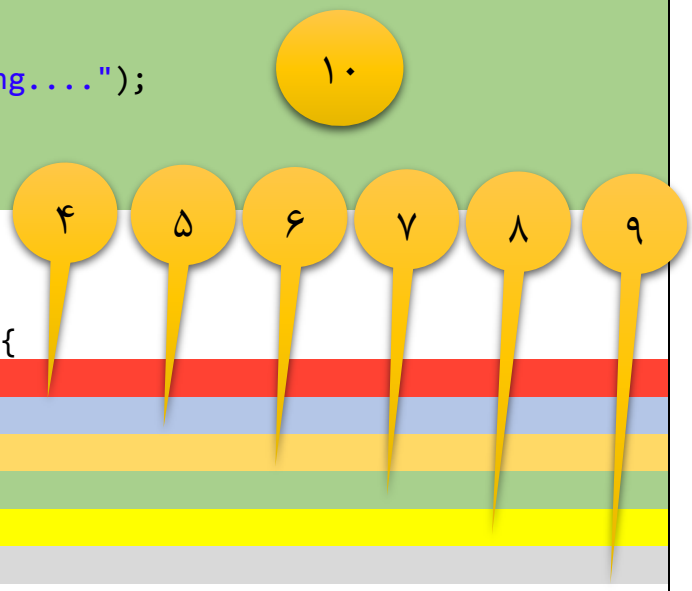
```
package polymorphism_JavaPro;

class Animal {
	public void makeSound() {

		System.out.println("animal  sounds");
	}

	public void eat() {

		System.out.println("animal  eat");
	}
}

class Cat extends Animal {
	public void makeSound() {

		System.out.println("miu miu");
	}
```

۱

۲

۳

```
public void sleep() {

            System.out.println("cat is sleeping....");
      }

}

public class Test {

      public static void main(String[] args) {
            Animal a1 = new Animal();
            a1.eat();
            a1.makeSound();
            Animal a2 = new Cat();
            a2.makeSound();
            a2.eat();

      }
}
```



خروجی:

```
animal  eat
animal  sounds
miu miu
animal  eat
```

شماره های بالایی که مشخص شده را بصورت زیر بررسی میکنیم:

۱) متد makeSound() درون کلاس Animal که دستور چاپ پیام "animal sounds" را اجرا میکند پیاده سازی شده است.

۲) متد eat که دستور چاپ پیام "animal eat" را اجرا میکند در کلاس Animal پیاده سازی شده است.

۳) متد makeSound() کلاس Animal (پدر) درون کلاس Cat (فرزند) override شده است.

۴) یک شی به نام a1 به روش عادی از کلاس Animal ایجاد کرده ایم.شی a1 به تمام ویژگی ها و رفتارهای کلاس Animal دسترسی دارد.

۵) با شی a1 متد eat() کلاس animal را صدا زده ایم و پیام animal eat در خروجی چاپ شده است.

۶) با شی a1 متد makeSound() کلاس animal را صدا زده ایم و پیام animal sounds در خروجی چاپ شده است.همان طور که مشاهده میکنید دقیقا دستور درون این متد در کلاس Animal اجرا شده است.

۷) یک شی به نام a2 به روش چندریختی از کلاس Animal ایجاد کرده ایم.با این تفاوت که شی a2 نوعش از کلاس Animal (کلاس پدر) و سازنده اش ،سازنده کلاس Cat فرزند می باشد.شی a2 نیز تنها به تمام متغیرها و متدهای کلاس Animal دسرسی دارد چون نوعش از نوع کلاس Animal می باشد.اما وقتی میخوایم با شی a2 متدهایی از کلاس Animal (کلاس پدر) که درون کلاس Cat (کلاس فرزند) override شده است را صدا بزنیم برنامه بجای اجرای دستور درون متد کلاس Animal (کلاس پدر) ،دستور درون متد override شده کلاس Cat (کلاس فرزند) را اجرا میکند.شماره ۸ این مفهوم را کامل میکند.

۸) مثلا در اینجا با شی a2 متد makeSound() که درون کلاس Animal قرار دارد را صدا زده ایم.برنامه چون میبینه این متد درون کلاس Cat (فرزند) نیز override شده است ، به جای اجرای دستور این متد در کلاس Animal (پدر) ، دستور override شده این متد در کلاس Cat (فرزند) را اجرا میکند. و در خروجی به جای پیام "animal sounds" ، پیام "miu miu" چاپ می شود.بازم تکرار میکنم شی a2 تنها به متغیرها و متدهای کلاس Animal (کلاس پدر) دسرسی دارد و هیچ دسرسی به کلاس Cat نداردا!! شبیه طرقه که صدای قناری می دهد اما ما چیزی به نام قناری نمیبینیم.

۹) در اینجا با شی a2 متد eat() که در درون کلاس Animal قرار دارد صدا زده ایم.

۱۰) ما هیچ وقت نمی تونیم با شی a2 متد sleap() درون کلاس Cat را صدا بزنیم چون شی a2 تنها به متدها و متغیرهای کلاس Animal دسرسی دارد.طبق تصویر(۱) شی a2 تنها به ویژگی ها و رفتارهای کلاس Animal دسرسی دارد و تنها متدهایی از کلاس Animal (کلاس پدر) که درون کلاس Cat (کلاس فرزند) overrid شده است

ب

تصویر(۱)

حالا که جزبه جز چندریختی رو گفتم چندریختی رو به زبان ساده براتون در چند خط میگم!!!!!

در چند ریختی :

۱) چندریختی یعنی یک کلاس به شکل کلاس های دیگر ظاهر شود.
۲) اگر یک کلاس میخواهد به شکل کلاس های دیگر ظاهر شود باید رابطه IS-A بین کلاس ها برقرار باشد یعنی کلاسمون باید پدر کلاس هایی باشد که قراره به شکل انها ظاهر شود.(ارث بری)
۳) حالا رفتارهای مشترک (متدهای مشترک) بین کلاس های پدر و فرزندان را override می کنیم. (Overriding)
۴) به شیوه چندریختی از کلاس پدر شی می سازیم،یعنی نوع شی از نوع کلاس پدر و سازنده از نوع کلاس فرزند.
۵) حالا با شی ساخته شده متدهایی از کلاس پدر که در کلاس فرزند override کردیم را صدا میزنیم،با این کار برنامه با وجود این که متد کلاس پدر را صدا زده ایم اما دستور این متد که در کلاس فرزند override شده است اجرا می شود.

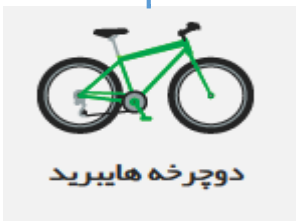
- در کل در چندریختی شی کلاس فرزند در کالبد شی کلاس پدر پنهان شده است و یک حالت انتزاعی را دارد به همین خاطر طرف حساب ما شی کلاس پدر هست و از طریق شی کلاس پدر به متدهایی که در کلاس فرزند override شده است دسرسی پیدا میکنیم.
- این آموزش را مکمل آموزش چندریختی جلسه قبل ارائه کردم امیدوارم این مفهوم چغر ☺ برای شما روشن شده باشد اگه هنوز چندریختی برای شما واضع نشده است بهترین راه حل کردن مثال و ایجاد تغییر در مثال و اجرا کردن آن می باشد.بهترین راه یادگیری باز کردن برنامه Eclipse و کدزدن عملی می باشد.

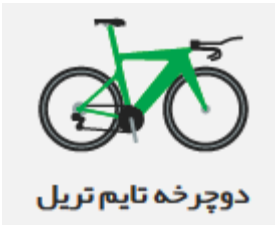
چون که منبع مناسب مثال در نت نیست در زیر سری مثال با جواب در زمینه چندریختی طراحی کرده ام:

مثال دوچرخه را میخواهیم بررسی کنیم:در زیر ۵ نوع دوچرخه زیر شاخه دوچرخه به معنای عمومی و کل می باشد. بین همه دوچرخه ها عمل راندن یا run() مشترک هست.پس در کلاس دوچرخه که پدر همه دوچرخه ها هستش متد run() را پیاده سازی میکنیم. و این متد را در دوچرخه های فرزند override میکنیم.بعد با استفاده از شی سازی به روش چندریختی کلاس دوچرخه (پدر) به شکل پنج دوچرخه فرزند ظاهر می شود:

دوچرخه

extends






```
package polymorphism_JavaPro;

class Bike {
	int speed;

	public void run() {
		speed = 50;
		System.out.println("speed bike is " + speed);
	}
}

class RacingBike extends Bike {
	public void run() {
		speed = 160;
		System.out.println("speed RacingBike is " + speed);
	}
}
```

```
class TimeTrialBike extends Bike {
	public void run() {
		speed = 70;
		System.out.println("speed TimeTrialBike is " + speed);
	}
}

class TouringBike extends Bike {
	public void run() {
		speed = 100;
		System.out.println("speed TouringBike is " + speed);
	}
}

class Hybrid_CityBike extends Bike {
	public void run() {
		speed = 45;
		System.out.println("speed Hybrid_CityBike is " + speed);
	}
}

class MountainBike extends Bike {
	public void run() {
		speed = 130;
		System.out.println("speed MountainBike is " + speed);
	}
}

public class Maintest extends Bike {

	public static void main(String[] args) {
//=================================================
		Bike RacingBike = new RacingBike();		//Polymorphism
		Bike TimeTrialBike = new TimeTrialBike();
		Bike TouringBike = new TouringBike();
		Bike Hybrid_CityBike = new Hybrid_CityBike();
		Bike MountainBike = new MountainBike();
		RacingBike.run();
		TimeTrialBike.run();
		TouringBike.run();
		Hybrid_CityBike.run();
		MountainBike.run();
//=============================================================
	}
}
```

خروجی:

```
speed RacingBike is 160
speed TimeTrialBike is 70
speed TouringBike is 100
speed Hybrid_CityBike is 45
speed MountainBike is 130
```

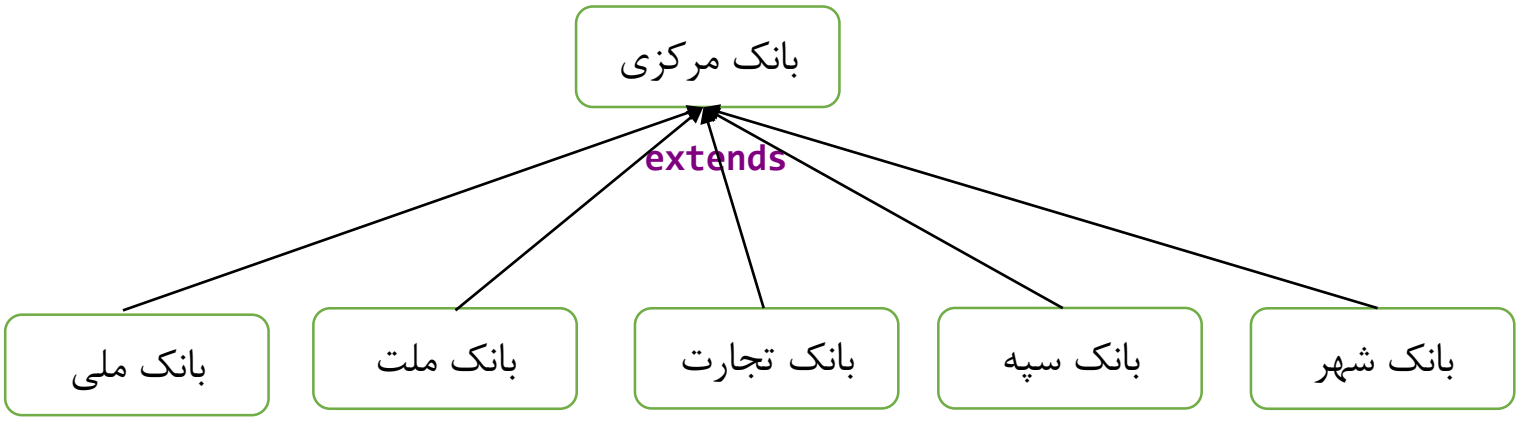
- در اینجا کلاس Bike (پدر) از طریق چندریختی به شکل فرزندان خود ظاهر شده است.

- یعنی اول کلاس های RacingBike، TimeTrialBike، TouringBike ، Hybrid_CityBike ، MountainBikeکلاس Bike را به ارث برده اند.(رابطه IS-A برقرار است) بعد متد run کلاس Bike (پدر) در کلاس های فرزندان یعنی RacingBike، TimeTrialBike، TouringBike ، Hybrid_CityBike ، MountainBike بازنویسی یا override شده است (Overriding) و در پایان هنگام شی سازی از کلاس پدر سازنده ها را از نوع کلاسهای فرزندان انتخاب کردیم که شی حاصل علاوه بر این که به تمام ویژگیها و رفتارهای کلاس پدر دسرسی دارد ، وقتی متدهای پدر که در کلاس فرزند بازنویسی یا override شده را صدا میزنیم دستورات متد پدر به دستورات بازنویسی شده ای(override شده) که در کلاس فرزندان قرار دارد تغییر میکند.

- پس در چند ریختی :
  ۱. ابتدا مبحث ارث بری
  ۲. بعد مبحث overriding
  ۳. ایجاد شی ای که نوعش از نوع کلاس پدر و سازنده اش از نوع سازنده کلاس فرزند.
  ۴. صدا زدن متدهای override شده و تغییر دستور درون بدنه متد پدر به دستور درون بدنه متد فرزند

مثال بعدی در زمینه بانک ها هستش: کلاس **بانک مرکزی** پدر کلاس های **بانک ملی،بانک ملت،بانک تجارت،بانک سپه،بانک شهر** می باشد.در اینجا میخوایم بررسی کنیم که چطور بانک مرکز به شکل بانک ملی،بانک ملت،بانک تجارت،بانک سپه،بانک شهر ظاهر می شود.

هر بانکی از نظر سود آوری یک رتبه ای دارد پس یک متد مشترک (getRateOfInterest()) بین همه بانک ها وجود دارد که رتبه سودآوری خود را به ما برمیگرداند.مثلا رتبه سودآوری بانک ملی ۱۲ ،بانک ملت ۱۵،بانک تجارت ۱۸،بانک سپه ۶،بانک شهر ۹ می باشد. (اینا همش بصورت فرضی هست حالا نیاین بگین رتبه سود آوری بانک فلان از بانک فلان بیشتره ☺)



۱. شرط اول را بررسی میکنیم،آیا رابطه IS-A بین بانک مرکزی و بانک ملی،بانک ملت،بانک تجارت،بانک سپه،بانک شهر وجود دارد؟ بله چون مثلا بانک ملی زیر مجموعه بانک مرکزی می باشد و..... ،همچنین و بانک ملی،بانک ملت،بانک تجارت،بانک سپه،بانک شهر بانک مرکزی را به ارث برده اند.

۲. حالا متد getRateOfInterest() را درون کلاس های فرزند override میکنیم.

۳. بعد به روش چندریختی از کلاس پدر شی میسازیم و با اون شی متد getRateOfInterest() را صدا میزنیم.

```
package polymorphism_JavaPro;

class CenterBank {

	int getRateOfInterest() {
		return 0;
	}
}

class MelyBank extends CenterBank {

	int getRateOfInterest() {
		return 12;
	}
}

class MelatBank extends CenterBank {

	int getRateOfInterest() {
		return 15;
```

```
	}
}

class TejaratBank extends CenterBank {

	int getRateOfInterest() {
		return 18;
	}
}

class SepahBank extends CenterBank {

	int getRateOfInterest() {
		return 6;
	}
}

class ShahrBank extends CenterBank {

	int getRateOfInterest() {
		return 9;
	}
}

public class TestBank {

	public static void main(String[] args) {
		//
==========================================================================
		CenterBank melyBank = new MelyBank();
		CenterBank melatBank = new MelatBank();
		CenterBank sepahBank = new SepahBank();
		CenterBank tejaratBank = new TejaratBank();
		CenterBank shahrBank = new ShahrBank();
		//
==========================================================================

		System.out.println("Rate melyBank is " +
melyBank.getRateOfInterest());
		System.out
				.println("Rate melatBank is " +
melatBank.getRateOfInterest());
		System.out
				.println("Rate sepahBank is " +
sepahBank.getRateOfInterest());
```

```
System.out.println("Rate tejaratBank is "
                        + tejaratBank.getRateOfInterest());
            System.out
                        .println("Rate shahrBank is " +
shahrBank.getRateOfInterest());
            //
==========================================================================
      }

}
```

خروجی:

```
Rate melyBank is 12
Rate melatBank is 15
Rate sepahBank is 6
Rate tejaratBank is 18
Rate shahrBank is 9
```

امیدوارم تونسته باشم مفهموم را جا بیاندازم ،تا میتونید مثال عملی کار کنید و کد بزنید که مفهومش براتون روشن بشه.

پیروز و موفق باشید

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.