آموزش زبان برنامه نویسی جاوا

(انتزاع) Abstraction

جلسه بیست و پنجم

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!!

Abstraction (انتزاع) فرایند پنهان کردن جزییات پیاده سازی برنامه و تنها نشان دادن قابلیت برنامه به کاربر (کسی که تنها استفاده کننده از برنامه می باشد) است.

خب تا اینجا یک تعریف رسمی از Abstraction یا انتزاع کردیم از آنجایی که ما با هم خیلی راحت هستیم رسمی بازی رو میزاریم کنار و راحت انتزاع یا Abstraction را با هم بررسی میکنیم ☺

من دیگه از اینجا به بعد بجای واژه Abstraction ،واژه انتزاع را به کار میبرم که هی نخوام زبان کیبرد رو پشت سرهم عوض کنم ☺

با مثال از دنیای واقعی شروع میکنم:

مثال ۱:انجام بازی در گوشی همراه

مثلا وقتی شما گوشی همراه در دست دارید و دارید بازی می کنید از فرایند و جزییات پشت بازی خبر ندارید!! مثلا وقتی در یک بازی جنگی با یک اسلحه شلیک میکنید و گلوله به سمت هدف پرتاب می شود اصلا مهم نیست که شما بدونید چطور در این بازی گلوله داره حرکت میکنه و چطور وقتی به هدف برخورد میکند شما امتیاز میگیرید!!! در هنگام بازی تنها شما با چند دکمه ای که روی گوشی شما هست کار میکنید و هیچ خبری از پشت کار ندارید!!! انتزاع هم همین مفهوم رو دنبال میکنه ،انتزاع هدفش اینه جزییات برنامه باید از دید کاربر پنهان شود و تنها قابلیت های برنامه در معرض دید کاربر قرار گیرد.

مثال ۲:فیلم سینمایی

وقتی شما یک فیلم سینمایی را تماشا میکنید و از آن لذت میبرید از جزییات پشت صحنه و کادر کارگردانی و فیلم برداری و .... یعنی از جزییات کار بی خبر هستید و همزمان تنها صحنه های خود فیلم را می بینید در حالی که جزییات اجرای فیلم را پنهان کرده اند.

مثال ۳:کامپیوتر

همزمان که دارم این اموزش را برای شما تهیه میکنم تنها از **قابلیت** کامپیوتر یعنی دکمه های کیبرد استفاده میکنم و از **جزییات** پردازش این دکمه هایی که فشار می دهم ، مانیتوری که کلمات وارد شده را نمایش می دهد و ده ها قطعه ای که در کامپیوتر وجود دارد و هر یک عمل متفاوتی را انجام میدهد بی خبر هستم.

انتزاع هم میگه ما باید کاری کنیم که جزییات برنامه پنهان شود و تنها قابلیت برنامه را برای کاربر به نمایش بگذاریم.

با این کار هم **امنیت** برنامه بالا می رود و هم کاربر آسوده تر است.چرا امنیت برنامه بالا می رود؟ دوباره میریم سراغ مثال چون که بهترین راه یادگیری فقط مثال است و دیگر هیچ ☺

به نظر شما امنیت کدام کیس کامپیوتر زیر بالاتر است؟!

الف)

ب)

خب مسلما کیس گزینه الف) دارای امنیت بیشتری است،چون کاربر به جزییات آن دسرسی ندارد اما کیس ب) دارای امنیت پایین تری هست چون حتی یک کاربر عادی هم می تواند به جزییات و قطعات درون آن دسرسی داشته و آن را دستکاری کند و الله اعلم که چه بر سر این کیس آورد ☺

پس انتزاع در شی گرایی یک مفهوم خوبیست و به ما کمک میکنه با پنهان سازی جزییات پیاده سازی برنامه و تنها نمایش قابلیت برنامه به کاربر امنیت برنامه بالا رود.

Abstraction در جاوا:

در جاوا ما کلاس مون رو میتوانیم انتزاع یا `abstraction` اعلام کنیم.

در جاوا هر کلاسی را می شود انتزاعی اعلام کرد اما بهتر است کلاس هایی را انتزاعی اعلام کنیم که مفهوم کلی داشته باشند یعنی کلاسی که مفهوم کل تر ،بزرگ تر داشته و می تواند نقش پدر کلاس های دیگر را بازی کند:

مثال: مثلا کلاس Animal (حیوان) یک مفهوم کلی را در بر دارد وقتی ما میگوییم حیوان منظورمان مشخص نیست کدام و چه نوع حیوانی،آیا منظور ما از حیوان گربه یا سگ یا مرغ یا خروس یا ماهی و .... است؟یا کلاس automobile (اتومبیل) نیز یک مفهوم کلی را میرساند و مشخص نیست منظور از اتومبیل ماشین شخصی است یا اتوبوس یا قطار یا هواپیما یا کشتی و....است پس کلاس هایی نظیر Animal و automobile و.... که مفهوم کلی را میرسانند و می توانند نقش کلاس پدر را بازی کنند را می توانیم انتزاعی یا abstract تعریف کنیم.

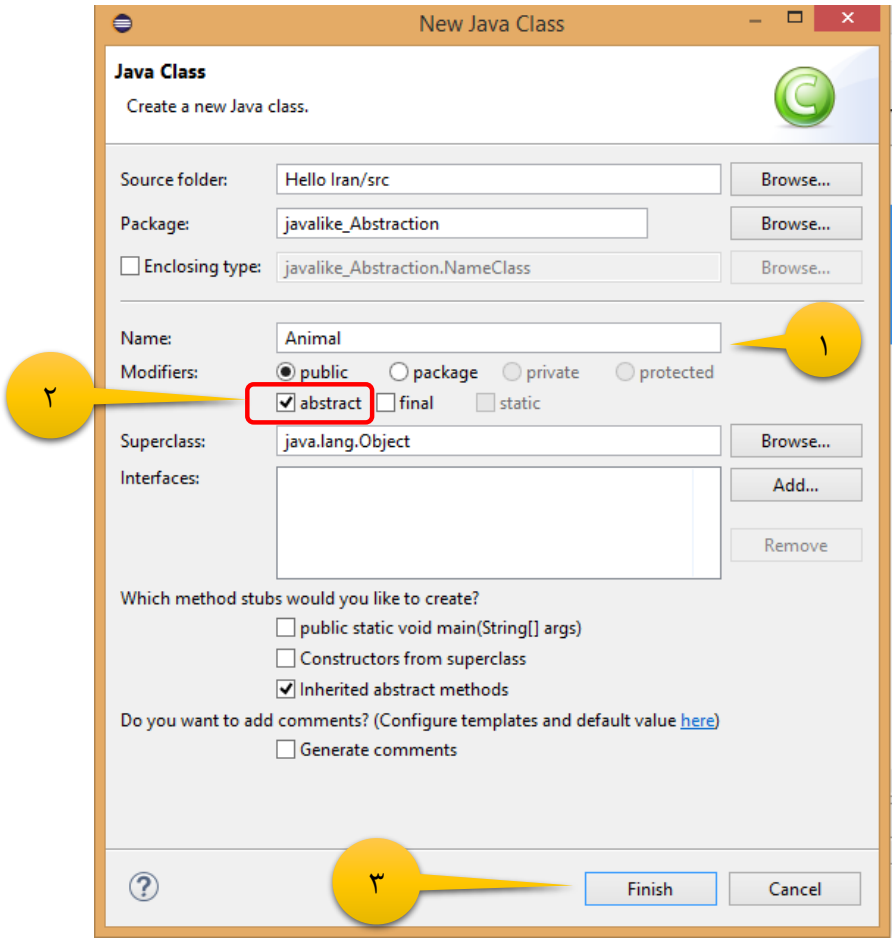
کلاس های انتزاعی (Abstract Class):

وقتی کلمه کلیدی **abstract** را قبل از کلمه class می آوریم کلاسمون تبدیل به یک کلاس انتزاعی (abstract) می شود.

نحوه نوشتن کلاس abstract:

```
public abstract class NameClass {


}
```

یا این که می توانید هنگام ایجاد کلاس جدید و تعیین نام برای کلاس تیک گزینه abstract را بزنید: تصویر(۱)



تصویر(۱)

نتیجه :

```
public abstract class Animal {

}
```

❖ یک کلاس انتزاعی (abstract) می تواند در بدنه خود یک متد از نوع انتزاعی (abstract) داشته باشد و بلعکس می تواند هیچ متد از نوع انتزاعی (abstract) در بدنه خود نداشته باشد.

❖ متد انتزاعی (abstract) به متدی می گویند که بدنه یا بلوک نداشته باشد!!!!یعنی تنها می تواند نوع،نام ،پارامتر داشته باشد و همچنین کلمه کلیدی abstract قبل از نوع متد آورده می شود:

```
abstract void setName(String name);
```

- اگر یک کلاس حداقل یک متد انتزاعی (abstract) داشت حتما باید کلاس از نوع انتزاعی (abstract) اعلام و تعریف شود:

```
class Animal extends Cat{

 abstract void setName(String name);
}
```

خطای کامپایل!!! ، زیرا این کلاس در بدنه خود یک متد انتزاعی (abstract) دارد اما کلاس از نوع انتزاعی (abstract) تعریف نشده است. شکل صحیح کلاس بالا بصورت زیر است:

```
abstract class Animal extends Cat{

    abstract void setName(String name);
}
```

- اگر یک کلاس از نوع انتزاعی (abstract) اعلام و تعریف کردیم دیگر نمی توان از آن نمونه و شی ایجاد کنیم.

```
package javalike_Abstraction;

abstract class Animal {

 abstract void setName(String name);

 public static void main(String[] args) {
      Animal    a    = new Animal();

 }
}
```

خطای کامپایل-زیرا از کلاس انتزاعی نمی شود شی ایجاد کرد

- خب وقتی ما یک کلاس را انتزاعی اعلام میکنیم و دیگر نمی توانیم شی از ان ایجاد کنیم خب چطور به ویژگی ها و رفتار های آن دسری پیدا کنیم؟؟؟؟؟؟!!!!!!!!!!!!!!!!!!

پاسخ: می توانیم از طریق کلاس دیگر کلاس انتزاعی را به ارث ببریم و متدهای انتزاعی آن را در کلاس فرزند پیاده سازی کنیم و از کلاس انتزاعی استفاده کنیم.

```
package javalike_Abstraction;

class Cat extends Animal {
	String name;

	void setName(String name) {
		this.name = name;

	}

}

abstract class Animal {

	abstract void setName(String name);

	public static void main(String[] args) {
		Cat c = new Cat();
		c.setName("makhmalak!");
		System.out.println(c.name);

	}
}
```

خروجی:

```
makhmalak!
```

- در این برنامه کلاس `Animal` انتزاعی تعریف شده است و دارای یک متد انتزاعی به نام `setName` می باشد.
- چون کلاس `Animal` انتزاعی تعریف شده نمی توان از آن شی ایجاد کرد.
- کلاس `Cat` کلاس `Animal` را به ارث برده است.از انجایی که کلاس `Animal` یک متد انتزاعی در بدنه خود دارد کلاس `Cat` بصورت اجباری باید متد انتزاعی کلاس `Animal` را در بدنه کلاس خود پیاده سازی کند.به همین خاطر متد `setName(String name)` در کلاس `Cat` پیاده سازی کرده ایم.در کلاس فرزند (`Cat`) می توانیم متد انتزاعی کلاس پدر (`Animal`) را بازنویسی کنیم و بدنه با توجه به دستور دلخواه براش تعریف کنیم.

- در پایان با شی ساختن از کلاس `Cat` متد `setName` را صدا زده و آن را مقداردهی می کنیم و در آخر مقدار متغیر `name` کلاس `Cat` را چاپ می کنیم.
- در اینجا کلاس `Animal` که انتزاعی می باشد کاملا پنهان مانده است و تنها نقشه ای بود که از روی آن کلاس های فرزند رفتار ها و متدهای انتزاعی کلاس پدر را بلاجبار باید در بدنه کلاس خود پیاده سازی کنند.

- کلاسی که یک کلاس انتزاعی را به ارث می برد، اگر کلاس انتزاعی دارای متدهای انتزاعی بود ،کلاس فرزند حتما باید متدهای انتزاعی را در بدنه کلاس خود پیاده سازی کنند.

```
class Cat extends Animal {

}

abstract class Animal {

 abstract void setName(String name);

}
```

خطای کامپایل!!!،زیرا کلاس `Cat` کلاس `Animal` که انتزاعی هست را به ارث برده اما متدهای انتزاعی آن را در بدنه خود پیاده سازی نکرده است.

- کلاس های فرزند می توانند برای متدهای انتزاعی کلاس پدر که در بدنه خود پیاده سازی کرده اند بدنه تعریف کنند و دستورات دلخواه خود را درون بدنه تعریف کنند.

```
class Cat extends Animal {
 String name;

 void setName(String name) {
       this.name = name;

 }

}

abstract class Animal {

 abstract void setName(String name);

}
```

مثال:

```
package javalike_Abstraction;

public class AbstractDemo {

	public static void main(String[] args) {
		Employee e = new Employee("Ali", "iran", 43); // Compile error

		e.mailCheck(); // Run-time error
	}
}

abstract class Employee {
	private String name;
	private String address;
	private int number;

	public Employee(String name, String address, int number) {
		System.out.println("Constructing an Employee");
		this.name = name;
		this.address = address;
		this.number = number;
	}

	public double computePay() {
		System.out.println("Inside Employee computePay");
		return 0.0;
	}

	public void mailCheck() {
		System.out.println("Mailing a check to " + this.name + " "
				+ this.address);
	}

}
```

خروجی:

```
خطای کامپایل و خطای زمان اجرا!!
```

- در این مثال کلاس `Employee` انتزاعی تعریف شده است اما هیچ متد انتزاعی در بدنه ان وجود ندارد.پس می توانیم یک کلاس انتزاعی بدون متد انتزاعی ایجاد کنیم.

```
Employee e = new Employee("Ali", "iran", 43);
```

- از آنجایی که کلاس Employee انتزاعی (abstract) هستش نمی توان از آن شی ایجاد کنیم و برنامه خطای کامپایل میده.

```
e.mailCheck();
```

- چون شی e از نوع کلاس انتزاعی Employee هست،و در کل نمی توانیم از کلاس انتزاعی شی و نمونه ایجاد کنیم در هنگام اجرای برنامه خطای می دهد.

یک نکته مهم : ما در انتزاع(Abstraction) همان طور که گفتیم نمی توانیم از یک کلاس انتزاعی شی و نمونه ایجاد کنیم، این رو بدانید که دلیل عدم ایجاد شی از یک کلاس انتزاعی این است که نمی توان از سازنده کلاس انتزاعی در ایجاد شی از کلاس انتزاعی استفاده کنیم.(تصویر(۲)

```
public abstract class A {

    public static void main(String[] args) {
        A a=new A();

    }

}
```

تصویر(۲)

همان طور که در تصویر(۲) مشاهده میکنید خطای کامپایل در سازنده کلاس انتزاعی A اتفاق افتاده است.به همین خاطر در مثال قبل در بخش e.mailCheck(); با شی e که از نوع کلاس انتزاعی Employee بود توانستیم متد mailcheck() را صدا بزنیم اما هنگام اجرا خطا داد.چون شی e از نوع Employee ایجاد شد اما سازنده کلاس Employee این شی را نتوانست بسازد!! فک کنم یکم پیچیده شد!!!!!!!

در کل میخوایم بگیم با شی ایجاد شده از کلاس انتزاعی می شود به متغیرها و متدهای کلاس انتزاعی دسرسی داشت اما در هنگام اجرای برنامه خطا خواهد داد پس در کل نمی توانیم از یک کلاس انتزاعی شی بسازیم ☺ برای روشن شدن این نکته مثال زیر را ببینید☺

```
package javalike_Abstraction;

public abstract class A {
    String name;
    int age;

    public void setname(String name) {
        System.out.println(name);

    }

    public static void main(String[] args) {
        A a = new A();

        a.setname("Javalike");
        a.age = 6;
        System.out.println(a.age + " " + a.name);

    }
}
```

خطای کامپایل

با وجود این که شی a از نوع کلاس انتزاعی A هستش اما با آن توانسته ایم به متغیرها و متدهای این کلاس انتزاعی دسرسی پیدا کنیم،اما این را بدانید که با وجود این که خطای کامپایل نداده اما در هنگام اجرای برنامه خطا میدهد

تصویر(۳)

خروجی مثال بالا:

```
Exception in thread "main" java.lang.Error: Unresolved compilation
problem:
    Cannot instantiate the type A

    at javalike_Abstraction.A.main(A.java:13)
```

همان طور که در خطای صورتی رنگ می بینید گفته نمی توانید از کلاس A شی و نمونه بسازید.

این نکته صرفا جهت یادگیری این موضوع بود که نگید چرا با شی کلاس انتزاعی می تونیم متغیرها و متدهاش رو صدا بزنیم در کل هیچ وقت نمی توانیم از کلاس انتزاعی شی بسازیم چون سازنده کلاس انتزاعی این اجازه را به ما نمیده!!! البته دلیل گفتن این نکته علاوه بر دلیل بالا دلیل دیگه هم داره ☺ ادامه رو بخون!!!!!

در نکته قبل گفتیم دلیل این که نمی توانیم از کلاس انتزاعی شی بسازیم اینه که سازنده کلاس انتزاعی این اجازه را به ما نمیده!!! خب سازنده اس دیگه کارش شی ساختنه!!!! گردنشم کلفته ☺ اما ما با یک راه حل این گردن کلفت رو حالی میکنیم:

راه حل ما این هست، مگه نگفتیم دلیل این که نمی توانیم یک شی از کلاس انتزاعی (abstract) ایجاد کنیم این هست که سازنده کلاس انتزاعی اجازه نمی دهد؟درسته؟خب ما بجای سازنده کلاس انتزاعی از سازنده دیگری استفاده می کنیم!!!! چطور؟!! می توانیم هنگام شی سازی از کلاس انتزاعی ،سازنده ان را از میان سازنده کلاس فرزندان آن انتخاب کنیم!!!!! (شی سازی به روش چندریختی)!!!! عجیبه نه؟!!! کد زیر رو ببین که بدون کد یادگیری هرگز ☺

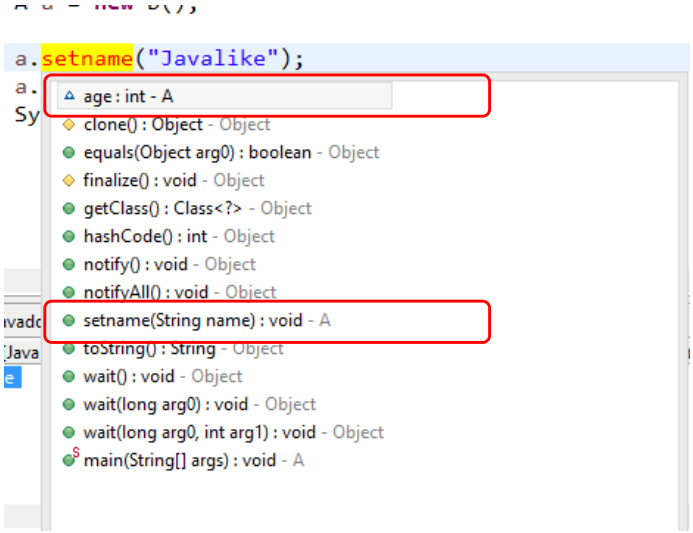
```
package javalike_Abstraction;

class B extends A {

	@Override
	public void setname(String name) {
		System.out.println("name is " + name);

	}

}

public abstract class A {

	int age;

	public abstract void setname(String name);

	public static void main(String[] args) {
		A a = new B();

		a.setname("Javalike");
		a.age = 6;
		System.out.println("age= " + a.age);

	}

}
```

خروجی:

```
name is Javalike
age= 6
```

حتما خیلی تعجب کردی نه؟!!! همان طور که گفتیم ما به ویژگی ها و رفتارهای کلاس انتزاعی (abstract) به دلیل این که نمی توانیم از کلاس انتزاعی شی بسازیم دسرسی نداریم و تنها از طریق کلاس دیگر که کلاس انتزاعی را به ارث میبرد می توانستیم با پیاده سازی متدهای انتزاعی در کلاس فرزند از ویژگی ها و متدهای کلاس انتزاعی استفاده کنیم، خب اینم یک راه دیگه است که از طریق شی سازی از کلاس انتزاعی و استفاده از سازنده کلاس فرزند می توانیم به ویژگی ها و رفتار های کلاس انتزاعی دسرسی پیدا کنیم.(این روش شی سازی همون روش چندریختی هستش)

سطح دسرسی شی a در مثال بالا که نوع اش از نوع کلاس انتزاعی و سازنده اش از کلاس فرزندش هست را در تصویر(۴) مشاهده کنید:



تصویر(۴)

همان طور که در تصویر(۴) مشاهده میکنید،شی a تنها به متغیرها و متدهای کلاس انتزاعی A دسرسی دارد.و چون متد setname را در کلاس فرزند بازنویسی (override) و پیاده سازی کرده ایم ،دستور متد setname در کلاس فرزند (B) اجرا می شود. (در کل چندریختی انجام شده است یعنی کلاس انتزاعی A به شکل کلاس فرزند خود B ظاهر شده است وگرنه هیچ وقت نمی توانیم مستقیم از یک کلاس انتزاعی شی بسازیم)

اگه هنوز براتون این مفهوم روشن نشده برید سراغ مثال بعد:

```java
package javalike_Abstraction;

abstract class Employee {

	String address;
	int number;

	abstract void daryagtHoghogh(double salary);
}

class Salary extends Employee {

	String name;

	public Salary(String name) {
		this.name = name;

	}

	void daryagtHoghogh(double salary) {
		System.out.println(name + " " + salary + "$ daryaft kard");
	}

}

public class Test {



	public static void main(String[] args) {



		Salary s = new Salary("hasan");
```

[M] Commented [۱]: کلاس ما درون این پکیج قرار دارد

[M] Commented [۲]: این کلاس انتزاعی اعلام شده است

[M] Commented [۳]: تعریف دو متغیر نمونه در بدنه کلاس انتزاعی

[M] Commented [٤]: دلیل بدنه نداشتن این متد این است که از نوع انتزاعی اعلام شده است

[M] Commented [٥]: تعریف کلاس Salary که کلاس Empoloyee را به ارث برده است.

[M] Commented [٦]: تعریف یک متغیر نمونه از نوع رشته در بدنه کلاس Salary

[M] Commented [۷]: سازنده کلاس که دارای یک پارامتر از نوع رشته است و پارامتر سازنده را درون متغیر نمونه کلاس میریزد

[M] Commented [۸]: این متد که در کلاس Empoloyee بصورت انتزاعی تعریف شده بود و چون کلاس Salary کلاس انتزاعی Empoloyee را به ارث برده است بلاجبار باید این متد را در کلاس خود پیاده سازی و باتوجه به نیاز بازنویسی یا override کنیم

[M] Commented [۹]: یک کلاس مجزا برای تست برنامه تعریف کرده ایم

[M] Commented [۱۰]: متد main که کاربرد آن اجرای دستورات برنامه می باشد.

[M] Commented [۱۱]: ساخت شی از کلاس Salary و مقدار دهی پارامتر سازنده ان

```
        Employee e = new Salary("ali");









        // ==================================================
        System.out.println("method  daryagtHoghogh "
                + "ra ba object s ke as noe class Salary hast seda zedeeim ");

        s.daryagtHoghogh(200000);

        System.out.println("method  daryagtHoghogh "
                + "ra ba object e ke as noe class Employee hast seda zedeeim ");

        e.daryagtHoghogh(50000);

    }
}
```

[M Commented [1]]: ساخت شی از کلاس انتزاعی به روش چندریختی-یعنی نوع شی از نوع کلاس پدر و سازنده ان از نوع سازنده کلاس فرزند که در اینجا نوع کلاس از نوع Empoloyee و سازنده ان از نوع کلاس Salary انتخاب شده است و همچنین سازنده کلاس Salary را مقداردهی کرده ایم.شی e که به این روش ایجاد شده تنها به تمام ویژگیها و رفتار های کلاس انتزاعی Empoloyee دسرسی دارد با این تفاوت که هنگام صدا زدن متدهایی که در کلاس فرزند خود پیاده سازی و بازنویسی (override) شده بجای اجرای دستورات متد پدر ،دستورات متد بازنویسی شده در کلاس فرزند اجرا می شود.

[M Commented [2]]: با شی s که از نوع کلاس Salary هست متد daryagtHoghogh را صدا زده و مقدار دهی کرده ایم

[M Commented [3]]: با شی e که از نوع کلاس Empoloyee و سازنده اش از نوع سازنده کلاس Salary است ، متد daryagtHoghogh را صدا زده و مقدار دهی کرده ایم

خروجی:

```
method  daryagtHoghogh ra ba object s ke as noe class Salary hast seda zedeeim
hasan 200000.0$ daryaft kard
method  daryagtHoghogh ra ba object e ke as noe class Employee hast seda zedeeim
ali 50000.0$ daryaft kard
```

- در این برنامه نیز به روش چندریختی از کلاس انتزاعی Employee شی ساختیم.یعنی نوع شی از کلاس Employee و سازنده از از نوع کلاس Salary

```
Employee e = new Salary("ali");
```

پس ما می توانیم از کلاس انتزاعی به روش چندریختی شی بسازیم و به ویژگی ها و متدهای کلاس انتزاعی دسرسی پیدا کنیم.

متدهای انتزاعی (Abstract Methods):

اگر هنگامی که کلاس فرزند کلاس پدر را به ارث می برد تمایل دارید متدهای خاصی از کلاس پدر در کلاس فرزند بلاجبار پیاده سازی شود در کلاس پدر متدهای مورد نظر را انتزاعی اعلام میکنیم.مثلا در دنیای واقعی پدر جعفر سفارش کرده اگر جعفر پراید

رو از من میخواد به ارث ببره باید پراید رو گازسوز کنه!!! پس جعفر در صورت به ارث بردن پراید مجبوره که پراید رو گازسوز کنه!!! ☺ پس اگر قصد داریم متدهای خاصی در کلاس فرزندمان بلاجبار پیاده سازی شود باید آن متدها را انتزاعی اعلام کنیم.

روش اعلام متدهای انتزاعی:

- با قرار دادن کلمه کلیدی `abstract` قبل از نوع متد می توانیم ان متد را انتزاعی اعلام کنیم:

```
abstract void daryagtHoghogh(double salary);
```

- یک متد انتزاعی همه بخش های متد عادی یعنی نوع ،نام،پارامتر را به جز **بدنه** دارا می باشد.
- در زیر یک کلاس انتزاعی که در بدنه ان یک متد انتزاعی تعریف شده است را مشاهده میکنید:

```
public abstract class Employee {
	private String name;
	private String address;
	private int number;

	public abstract double computePay();

}
```

❖ وقتی یک متد انتزاعی اعلام می شود دو پیامد دارد:

۱. کلاسی که این متد انتزاعی در ان قرار دارد باید انتزاعی اعلام شود.

۲. هر **کلاسی** که **کلاس انتزاعی** که دارای **متد انتزاعی** می باشد را به **ارث** ببرد **باید متدهای انتزاعی** آن **کلاس** را در **خود پیاده سازی و بازنویسی (override)** کند.

در زیر کلاس `Salay` کلاس انتزاعی `Empoloyee` را به ارث برده است،و از انجایی که کلاس `Empoloyee` دارای متد انتزاعی `computepay` می باشد،پس باید کلاس `Salary` متد `computepay` را در بدنه کلاس خود پیاده سازی و باتوجه به نیاز خود آن را بازنویسی یا `override` کند.

```
package javalike;

public class Salary extends Employee {


	public double computePay(double salary) {

		return salary / 52;
	}
```

```
}

abstract class Employee {

      public abstract double computePay(double salary);

}
```

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.