آموزش زبان برنامه نویسی جاوا

اینترفیس (Interface)

جلسه بیست و هفتم

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!!

اینترفیس چیست؟

اینترفیس مانند یک کلاس بنظر می رسد اما یک کلاس نیست!!! یک اینترفیس مثل کلاس می تواند متغیر و متد داشته باشد با این تفاوت که متدهای درون یک اینترفیس مثل متدهای کلاس abstract (انتزاعی) بدنه ندارند.یعنی متدهای یک اینترفیس دارای نوع ، نام ، پارامتر هستند اما بدنه ندارند.

در اینترفیس متغیرها می توانند public، static ، default،final اعلام و تعریف شوند.

تمامی متدهای درون اینترفیس بدون بدنه هستند.به عبارتی می توان گفت که اینترفیس ها مجموعه ای از متدهای abstract یا انتزاعی هستند.

نحوه پیاده سازی یک اینترفیس:

هنگام تعریف یک اینترفیس ابتدا از کلمه کلیدی **interface** استفاده می کنیم. بعد از کلمه کلیدی **interface** یک **نام** برای آن انتخاب کرده و با باز و بسته کردن **آکولاد** {} **بدنه** آن را تشکیل می دهیم.

- می توانیم ابتدای کلمه کلیدی **interface** از کلمه کلیدی public نیز استفاده کنیم.
- بهتر است نام اینترفیس با حروف بزرگ شروع شود.
- روش تعریف اینترفیس شبیه به روش تعریف یک کلاس است با این تفاوت که بجای کلمه کلیدی class از کلمه کلیدی interface استفاده کرده ایم.

پس یک بار دیگر روش ایجاد یک اینترفیس را گام به گام مرور می کنیم:

۱) استفاده از کلمه کلیدی public

```
public
```

۲) استفاده از کلمه کلیدی interface

```
public interface
```

۳) تعیین نام برای اینترفیس

```
public interface Name
```

۴) استفاده از دو بلوک آکولاد باز و بسته به عنوان بدنه اینترفیس

```
public interface Name {

}
```

در زیر یک اینترفیس (interface) با نام Animal داریم:

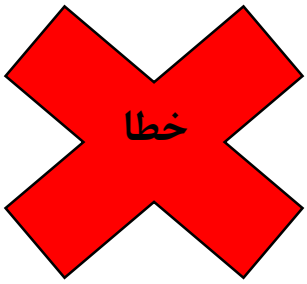
```
public interface Animal {

}
```

حالا در زیر یک اینترفیس (interface) با نام Animal داریم که دو متد بدون بدنه در آن تعریف شده اند:

```
package interface_javalike;

public interface Animal {

	public void sleep();

	public void eat();

}
```

در مثال زیر با خطای کامپایل برخورد می کنیم،زیرا در اینترفیس متدهای ما حق داشتن بدنه را ندارند:

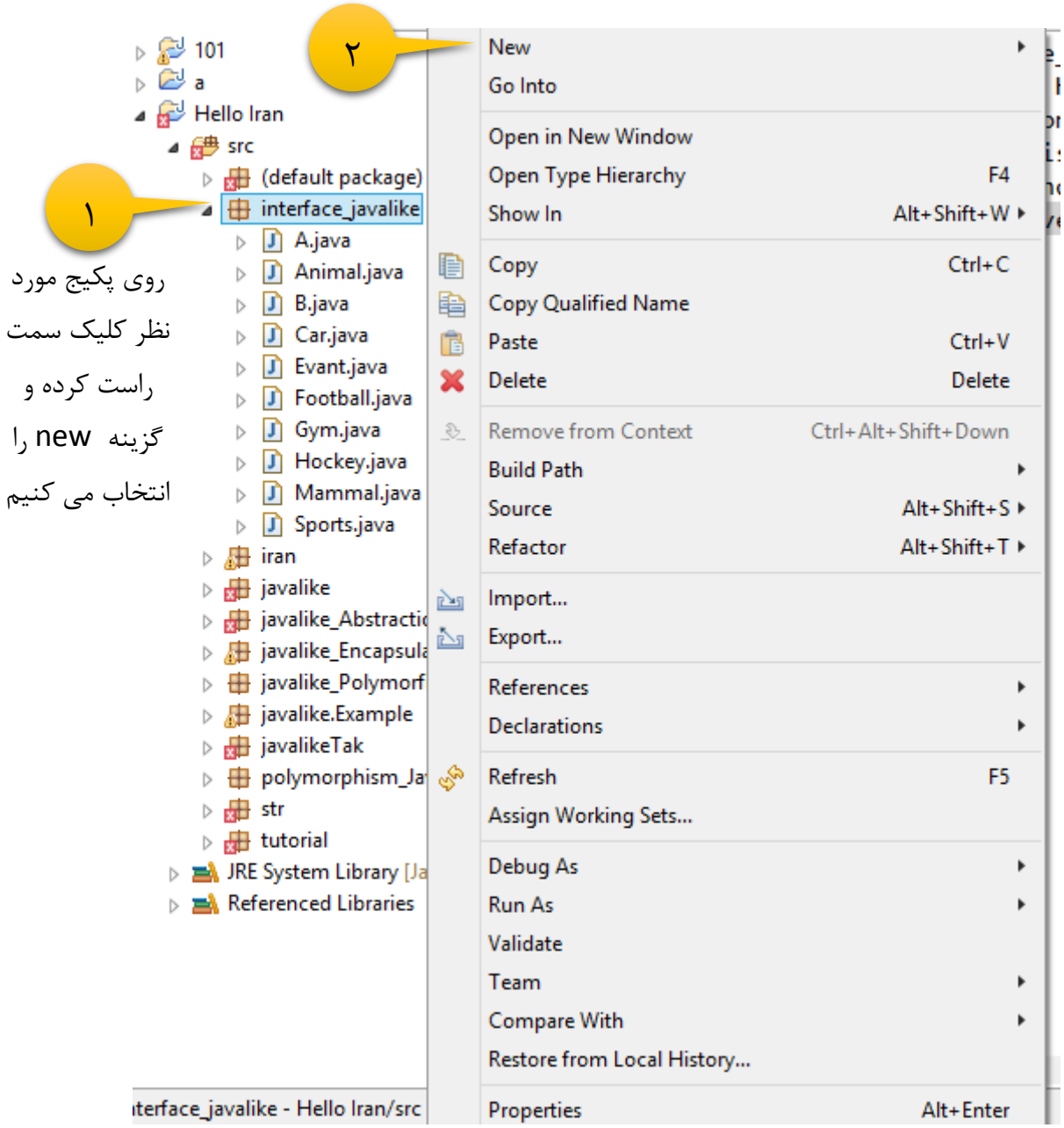
```
package interface_javalike;

public interface Animal {

	public void sleep(){


	}

	public void eat();

}
```



شما می توانید مانند کلاس یک اینترفیس را در نرم افزار ایکلیپس با چند کلیک ایجاد کنید:

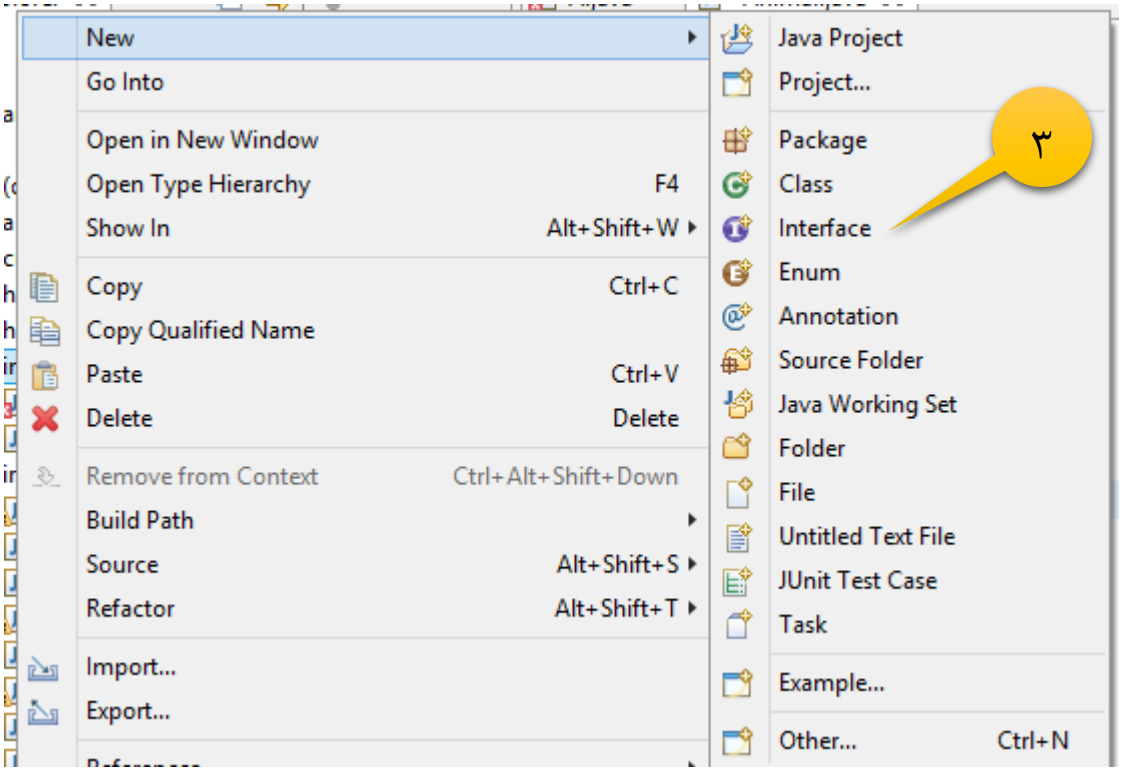
برای ایجاد یک اینترفیس در Eclipse به روش زیر عمل می کنیم:

ابتدا روی پکیج مورد نظر کلیک سمت راست کرده و گزینه new را انتخاب می کنیم: تصویر(۱)



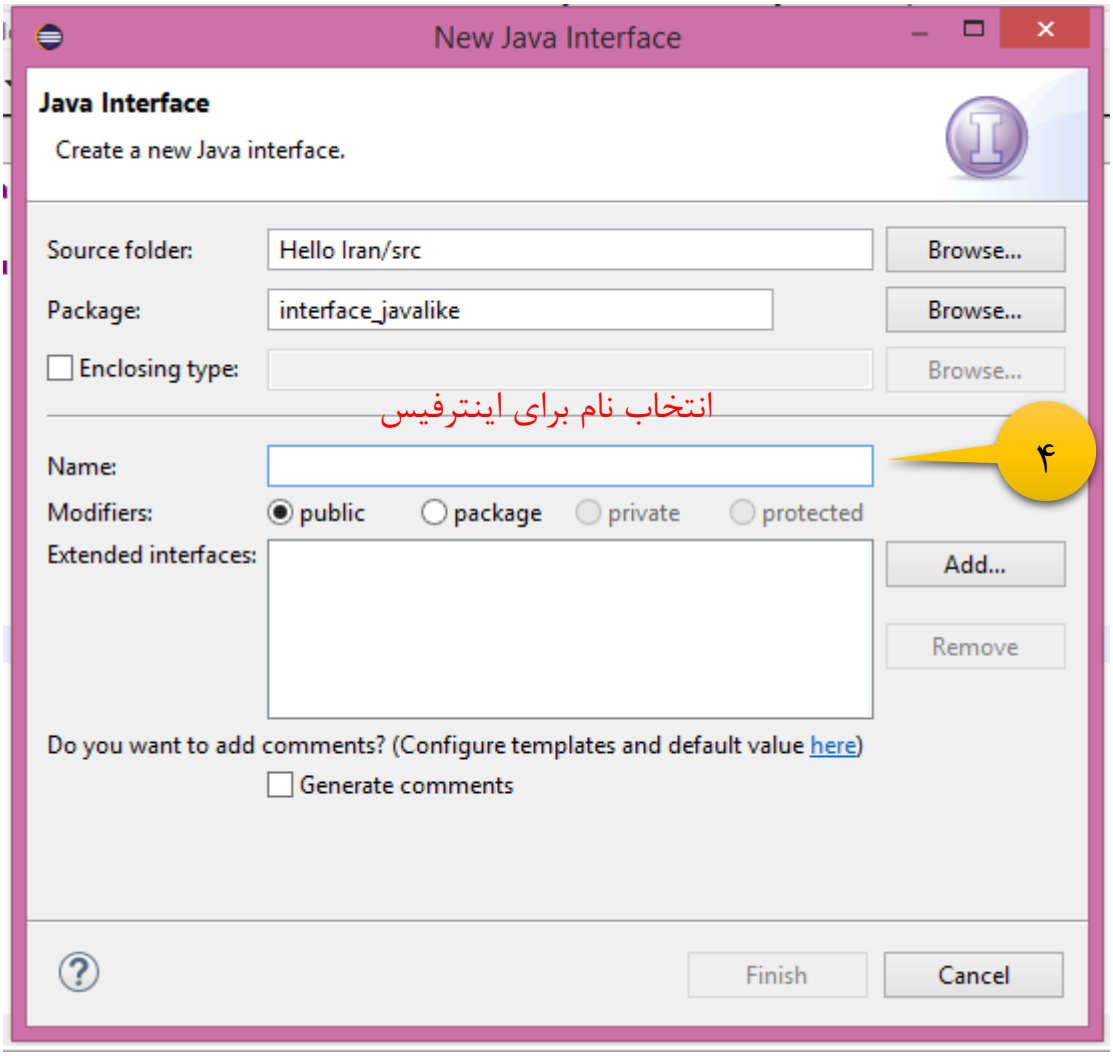
تصویر(۱)

بعد از انتخاب گزینه new زیر شاخه جدیدی که باز می شود گزینه `interface` را انتخاب می کنیم: تصویر(۲)



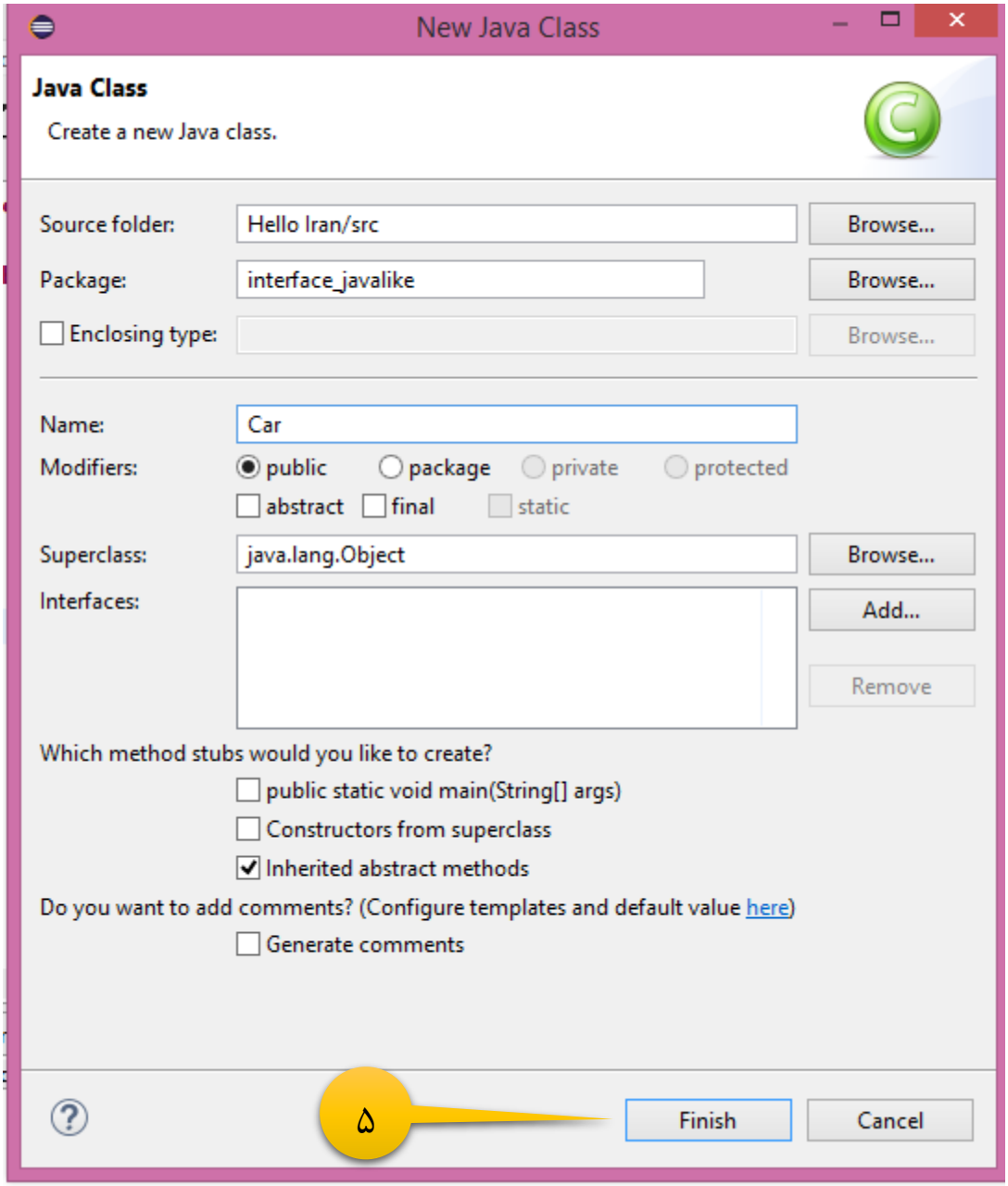
تصویر(۲)

بعد از انتخاب `interface` پنجره جدیدی تحت عنوان `New Jave Interface` باز می شود.که در این پنجره در بخش `Name` آن یک نام برای اینترفیس انتخاب کرده و دکمه `Finish` را می زنیم. تصویر(۳)و(۴)



تصویر(۳)

که در اینجا نام `Car` را برای اینترفیس انتخاب کرده ایم و سایر گزینه ها را به حالت پیشفرض قرار داده و در پایان دکمه `Finish` را می زنیم: تصویر (۴)



تصویر(۴)

نتیجه این مراحل در ایکلیپس ایجاد یک `interface` با نام `Car` می باشد: تصویر (۶)

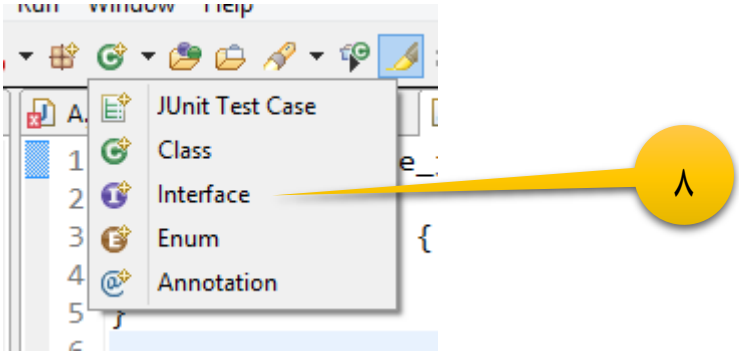
```
1  package interface_javalike;
2
3  public class Car {
4
5  }
6
```

تصویر(۶)

راه دوم و سریع تر این هست که از میان ابزارهای زیر منوبار گزینه ای که در تصویر نشان داده ایم را انتخاب کرده و مانند مراحل قبل اینترفیس را نام گذاری کرده و دکمه `finish` را بزنیم:تصویر(۷) و(۸)و

تصویر(۷)

فلش مشکی رنگ رو به پایین کنار این دایره سبز رنگ را انتخاب می کنیم



بین گزینه های زیر شاخه باز شده گزینه interface را انتخاب میکنیم

تصویر(۸)

سایر مراحل بعدی نیز مانند روش اول می باشد.

- پس کلا روش ایجاد یک اینترفیس شبیه به روش ایجاد یک کلاس هست با این تفاوت که کلمه کلیدی آنها متفاوت است.

خصوصیاتی که یک اینترفیس باید داشته باشد:

۱. تمامی متدهای درون اینترفیس بدون بدنه و انتزاعی هستند و نیازی نیست که با کلمه کلیدی `abstract` این متدها را انتزاعی اعلام کنید.
۲. بهتر است که متدهای درون اینترفیس را `public` اعلام کنیم.

```
public interface Animal {

	public void sleep();

	public void eat();
	public void run();

}
```

تا اینجا با نحوه ایجاد اینترفیس در جاوا آشنا شدیم.حالا اینترفیس چه استفاده هایی داره؟! ما در کلاس های خود می توانیم از اینترفیس ها استفاده کنیم.به عبارتی می توانیم اینترفیس ها را در کلاس خود **پیاده سازی** کنیم.خب چطوری؟!!! اصلا اینترفیس چه کاربردی در کلاس هامون داره؟!!!

اینترفیس حکم یک نقشه را برای کلاس ما دارد همان طور که در نقشه ساختمانی که روی کاغذ رسم شده است حاوی اطلاعات ساختمانی هست که قرار است پیاده سازی و ایجاد شود و پیمانکاران و مهندسان و.... بلااجبار باید مو به مو فونداسیون و تعداد اتاق ها و سالن پذیرایی و... را اجرا کنند.

اینترفیس نیز حکم نقشه ای را دارد که اگر به کلاسی بدهیم آن کلاس موظف است مو به مو تمامی رفتارهایی که اینترفیس دارد را در بدنه خود پیاده سازی کند.

خب تا اینجا فهمیدیم که در کلاس هامون از اینترفیس استفاده می کنیم.حالا نحوه استفاده کلاس از اینترفیس را بررسی می کنیم:

پیاده سازی اینترفیس در کلاس:

وقتی که یک کلاس یک اینترفیس را در خود پیاده سازی میکند، می تونیم تصور کنیم که کلاس قراردادی را به امضا رسانده است که موافقت کرده است که تمام رفتارها و متدهای اینترفیس را در بدنه خود اجرا کند.

پس اگر یک کلاس قصد داشت یک اینترفیس را در خود پیاده سازی کند بلاجبار باید تمامی متدهای اینترفیس را در بدنه خود پیاده سازی کند.

یک کلاس با استفاده از کلمه کلیدی **`implements`** می تواند یک اینترفیس (`interface`) را در خود پیاده سازی کند.

چطوری؟! به این صورت که هنگام تعریف کلاس ، بعد از نام کلاس از کلمه کلیدی **implements** استفاده کرده و بعد از کلمه کلیدی **implements** نام اینترفیس مورد نظر را می آوریم.

نحوه نوشتن پیاده سازی اینترفیس در کلاس:

```
public class NameClass implements NameInterface {


	}
```

نحوه استفاده از اینترفیس شبیه به ارث بردن یک کلاس هست با این تفاوت که بجای کلمه کلیدی extends از کلمه کلیدی implements استفاده می کنیم .

همان طور که گفتیم بعد از پیاده سازی یک اینترفیس در یک کلاس تمامی متدهای اینترفیس باید در کلاس بلاجبار پیاده سازی شود.

مثال:

اینترفیس B که دو متد انتزاعی(تمامی متدهای درون اینترفیس انتزاعی هستند) بدون بدنه در آن تعریف کرده ایم:

```
package interface_javalike;

public interface B {

	public void print(String name);

	public int sum(int a, int b);

}
```

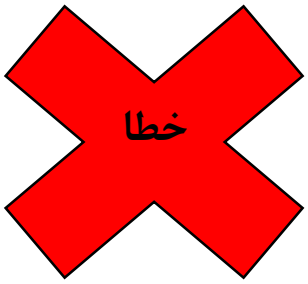
حال کلاس A اینترفیس B را پیاده سازی کرده است:

```
package interface_javalike;
public class A implements B {

	public void print(String name) {

	}
	public int sum(int a, int b) {

		return 0;
	}
}
```

همان طور که مشاهده می کنید تمامی متدهای اینترفیس B در کلاس A پیاده سازی و بازنویسی شده است.

به مثال زیر توجه کنید:

```
package interface_javalike;

public class A implements B {

	public void print(String name) {

	}

}
```



خطای کامپایل داده خواهد شد ،زیرا کلاس A همه متدهای اینترفیس B را پیاده سازی نکرده است.

نکته مهم : اگر یک کلاس دوست ندارد تمامی متدهای یک اینترفیس را در خود پیاده سازی کند کافیست که کلاس خودش را از نوع کلاس `abstract` اعلان کند.

حالا مثال قبل رو بصورت زیر تغییر می دهیم:

اینترفیس B :

```
package interface_javalike;

public interface B {

	public void print(String name);

	public int sum(int a, int b);

}
```

کلاس A که از نوع `abstract` هستش اینترفیس B را پیاده سازی یا به اصطلاح `implements` کرده است:

```
package interface_javalike;

public abstract class A implements B {

	public void print(String name) {

	}

}
```

نکته ای که مثال بالا داشت این هست که با وجود این که کلاس A تمامی متدهای اینترفیسB را پیاده سازی نکرده بود اما هیچ خطای کامپایلی داده نشده است ،دلیلش این هست که کلاس A از نوع abstract بوده است.پس یک کلاس از نوع abstract می تواند به دلخواه همه یا بخشی از متدهای یک اینترفیس را در خود پیاده سازی کند.

نکته مهم: در مثال هایی که اینترفیس را جدا و کلاس را جدا در یک پکیج تعریف کرده ایم ، شما نیز برای تست این مثال ها ابتدا اینترفیس ها را در پکیج مورد نظر خود در Eclipse ایجاد کنید و سپس کلاس ها را در همان پکیج بصورت جدا تعریف و ایجاد کنید.و در پایان برنامه را اجرا کنید.

مثال :

اینترفیس Animal :

```
package interface_javalike;

interface Animal {
	public void eat();

	public void travel();
}
```

نکته مهم: فایل اینترفیس ها نیز مانند فایل کلاس ها با فرمت java هستند.مثلا برای مثال بالا فایل اینترفیس Animal بصورت زیر است:

Animal.java

کلاس Mammal :

```
package interface_javalike;

public class Mammal implements Animal {

	public void eat() {
		System.out.println("Mammal eats");
	}

	public void travel() {
		System.out.println("Mammal travels");
	}

	public int speed() {
```

```
        return 100;
    }

    public static void main(String args[]) {
        Mammal m = new Mammal();
        m.eat();
        m.travel();
    }
}
```

خروجی:

```
Mammal eats
Mammal travels
```

- کلاس Mammal (پستاندار) اینترفیس Animal (حیوان) را پیاده سازی یا implements کرده است.
- بعد از پیاده سازی اینترفیس Animal درون کلاس Mammal بلاجبار تمامی متدهای اینترفیس Animal درون کلاس Mammal پیاده سازی شده است.و باتوجه به نیاز آن متدها را بازنویسی یا override کرده ایم.
- در متد main کلاس Mammal ، از کلاس Mammal شی ساخته و از طریق شی ایجاد شده متدهای eat و travel را صدا زده ایم.

نکات مهمی که در پیاده سازی اینترفیس در یک کلاس باید رعایت کنیم:

نکته: به متغیر محلی درون پرانتز جلوی یک متد پارامتر یا امضای متد گفته می شود.

۱) هنگام بازنویسی یا override متدهای اینترفیس درون یک کلاس باید امضا یا پارامتر و نوع متد بازنویسی شده درون کلاس با متد درون اینترفیس یکسان باشد.
۲) اگر یک کلاس از نوع abstract باشد نیاز نیست همه متدهای اینترفیس را در خود پیاده سازی کند.

نکته مهم: در کل مفاهیمی که کلی تر و انتزاعی هستند را بصورت interface تعریف می کنیم.

قوانین پیاده سازی یک اینترفیس:

۱) یک کلاس می تواند همزمان یک یا چندین interface را پیاده سازی یا implements کند.
۲) یک کلاس تنها یک کلاس را می تواند extend یا به ارث ببرید اما می تواند تعداد زیادی interface را پیاده سازی یا implements کند.

۳) یک اینترفیس می تواند ، اینترفیس دیگر را به ارث ببرد شبیه به روشی که یک کلاس ، کلاسی دیگر را به ارث می برد.

چگونه یک اینترفیس ، اینترفیس دیگر را به ارث می برد:

یک اینترفیس می تواند ، اینترفیس دیگر را به ارث ببرد شبیه به روشی که یک کلاس ، کلاسی دیگر را به ارث می برد.با استفاده از کلمه کلیدی **extends** یک اینترفیس می تواند اینترفیس دیگر را به ارث ببرد.با این کار اینترفیس فرزند می تواند تمامی متدهای اینترفیس پدر خود را به ارث ببرد. و وقتی یک کلاس قصد داشت از اینترفیس فرزند استفاده کند باید علاوه بر متدهای اینترفیس فرزند ،متدهای اینترفیس پدر آن را نیز در بدنه خود پیاده سازی کند.

مثال:

اینترفیس Sport که دارای دو متد انتزاعی می باشد.

```
package interface_javalike;

public interface Sports {

   public void setHomeTeam(String name);

   public void setVisitingTeam(String name);
}
```

اینترفیس Football اینترفیس Sport را به ارث برده است.با این کار اینترفیس Football علاوه بر متدهای خود متدهای پدرش را نیز دربر دارد.یعنی مجموع متدهای اینترفیس Football شش عدد می شود که دوتای آن مربوط به اینترفیش پدر و چهار تای آن به اینترفیس Football تعلق دارد.پس هر کلاسی که اینترفیس Football را در خود پیاده سازی کرد باید شش متد آن اینترفیس را در خود پیاده سازی کند.

```
package interface_javalike;

public interface Football extends Sports {

      public void homeTeamScored(int points);

      public void visitingTeamScored(int points);

      public void endOfQuarter(int quarter);
}
```

اینترفیس Hockey اینترفیس Sport را به ارث برده است.با این کار اینترفیس Hockey علاوه بر متدهای خود متدهای پدرش را نیز دربر دارد.و شبیه به اینترفیس Football می باشد

```
package interface_javalike;

public interface Hockey extends Sports {

   public void homeGoalScored();

   public void visitingGoalScored();

   public void endOfPeriod(int period);

   public void overtimePeriod(int ot);
}
```

مثال : در مثال زیر اینترفیس Football اینترفیس Sport را به ارث برده است، پس اینترفیس Football علاوه بر متدهای خود متدهای اینترفیس پدرش یعنی Sport را دربر دارد و هرکلاسی که اینترفیس Football را پیاده سازی کرد باید علاوه بر متد های اینترفیس Football متدهای اینترفیس پدر Football یعنی Sport را نیز در بدنه خود پیاده سازی کند.

خب در این مثال کلاس Gym اینترفیس Football را به ارث برده است.پس کلاس Gym باید تمامی متدهای اینترفیس Football و اینترفیس پدر Football را در بدنه خود پیاده سازی کند.

اینترفیس Sport :

```
package interface_javalike;

public interface Sports {

      public void setHomeTeam(String name);

      public void setVisitingTeam(String name);
}
```

اینترفیس Football :

```
package interface_javalike;

public interface Football extends Sports {

      public void homeTeamScored(int points);
```

```
	public void visitingTeamScored(int points);

	public void endOfQuarter(int quarter);
}
```

کلاس Gym :

```
package interface_javalike;

public class Gym implements Football {

	public static void main(String[] args) {

	}

	public void setHomeTeam(String name) {

	}

	public void setVisitingTeam(String name) {

	}

	public void homeTeamScored(int points) {

	}

	public void visitingTeamScored(int points) {

	}

	public void endOfQuarter(int quarter) {

	}

}
```

همان طور که مشاهده می کنید تمامی متدهای اینترفیس Football و اینترفیس پدر Football یعنی Sport در کلاس Gym پیاده سازی شده است.

وراثت چندگانه در اینترفیس:

وراثت چندگانه در مورد کلاس ها این بود که یک کلاس بیشتر از یک کلاس را می تواند به ارث ببرد که در جاوا همچین قانونی وجود ندارد یعنی یک کلاس تنها یک کلاس دیگر را می تواند به ارث ببرد.

اما اینترفیس انگار گردنش کلفت تر از کلاس هاست!!!! ☺ وراثت چندگانه در مورد اینترفیس ها جواب میده و مشکلی پیش نمیاد یعنی یک اینترفیس می تواند یک یا چندیدن اینترفیس را به ارث ببرد.

برای مثال اینترفیس Hockey در مثال زیر علاوه بر اینترفیس Sport اینترفیس Event را نیز به ارث برده است:

```
package interface_javalike;

public interface Sports {
	public void setHomeTeam(String name);

	public void setVisitingTeam(String name);
}
```

```
package interface_javalike;

public interface Evant {

	public void checkCollision();

	public void checkScore();

	public void run();

	public void stop();

}
```

```
package interface_javalike;
public interface Hockey extends Sports , Evant{
   public void homeGoalScored();
   public void visitingGoalScored();
   public void endOfPeriod(int period);
   public void overtimePeriod(int ot);
}
```

همان طور که مشاهده میکنید ، اینترفیس Hockey دو اینترفیس Sports و Evant را به ارث برده است.

پس نتیجه می گیریم که یک اینترفیس می تواند چندین اینترفیس دیگر را به ارث ببرد.

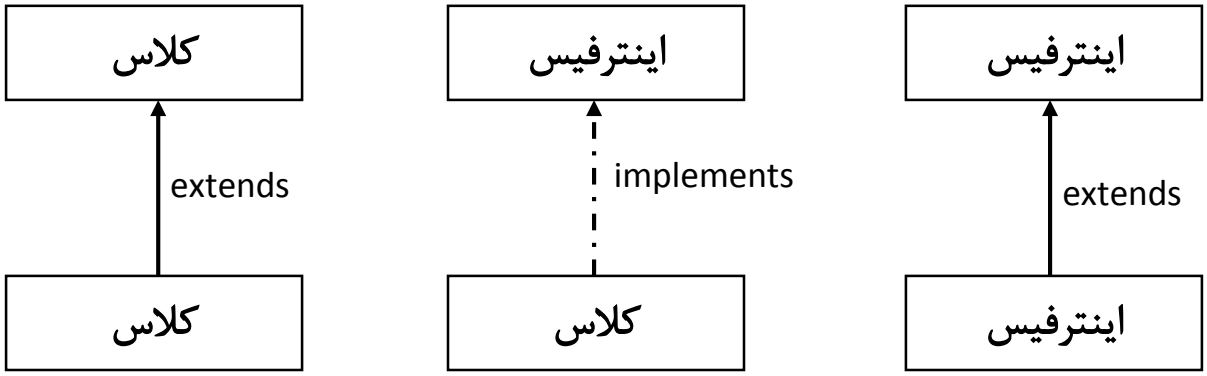
درک بهتر رابطه بین کلاس ها و اینترفیس:

در زیر با اشکال نشان داده ایم که

۱) یک کلاس ،کلاسی دیگر را به ارث برده است.

۲) یک کلاس یک اینترفیس را پیاده سازی یا `implements` کرده است.

۳) یک اینترفیس ،اینترفیسی دیگر را به ارث برده است.



(۱) (۲) (۳)

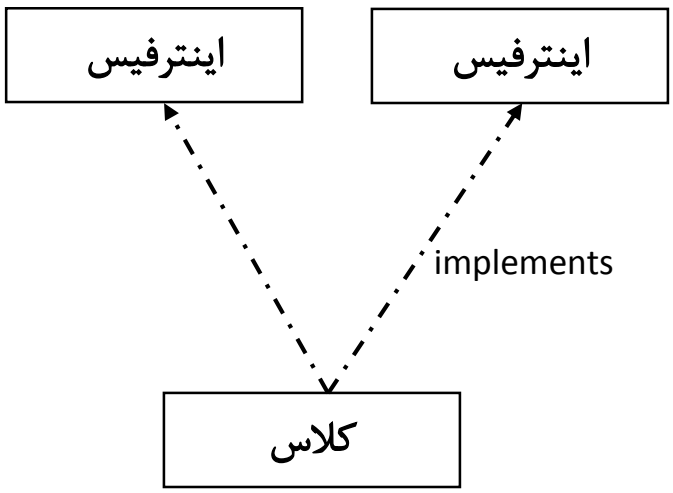
- وقتی یک کلاس ، کلاسی دگر را `extends` می کند با فلش ساده نشان می دهیم شکل (۱)
- وقتی یک کلاس یک اینترفیس را `implements` می کند با فلش خط چین دار نشان می دهیم شکل (۲)

درک بهتر وراثت چندگانه توسط اینترفیس در جاوا:

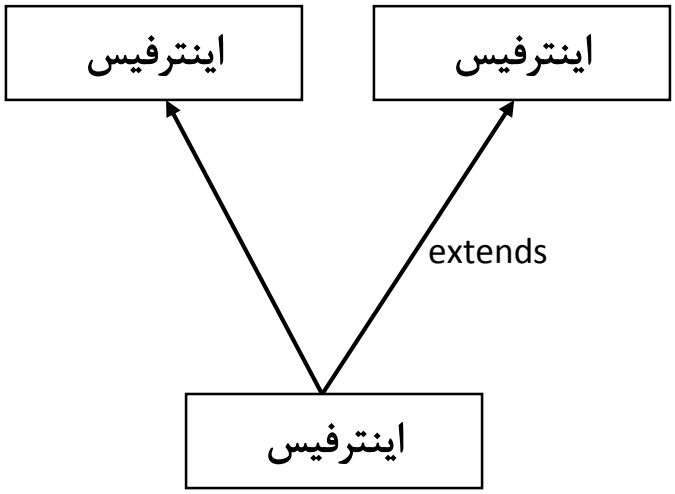
اگر یک کلاس چندین اینترفیس را `implements` کند یا یک اینترفیس چندین اینترفیس را به ارث ببرد به این اعمال وراثت چندگانه در جاوا می گوییم.همان طور که گفتیم در جاوا نمی شود یک کلاس چندین کلاس را به ارث ببرد اما بجای آن یک کلاس می تواند چندین اینترفیس را پیاده سازی کند. و همچنین یک اینترفیس می تواند چندین اینترفیس را به ارث ببرد.

۱) یک کلاس می تواند بیشتر از یک اینترفیس را پیاده سازی یا `implements` کند.

۲) یک اینترفیس می تواند بیشتر از یک اینترفیس را به ارث ببرد.



(۱)



(۲)

وراثت چندگانه در جاوا به کمک اینترفیس

خب مبحث اینترفیس هم تمام شد که بصورت جامع و کامل براتون توضیح دادم،برای درک مفاهیم در کنار هر مفهوم یک مثال ساده و کوتاه زده ایم،درسته کاربردی نبوده! چون هدف از مثال های ساده تنها روشن کردن مفاهیم بوده است.بعد از این که تمامی جلسات تمام که نه!! چون جاوا تمامی نداره ☺ به یک نقطه مناسبی رسیدیم برمیگردیم و از اول اول جاوا ، شروع به حل مثال از ساده به دشوار و کاربردی می کنیم.پس آموزش های ما را دنبال کنیم ☺

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.