**به نام خدا**

## آموزش پنجاه و دوم

اهداف آموزشی این قسمت عبارتند از:

۱. نحوه ایجاد یک Interface
۲. نحوه ایجاد یک متد در یک Interface
۳. نحوه اجرا کردن یک Interface در یک Class

پس از آشنایی با مفهوم Interface در آموزش قبل، در این آموزش قصد داریم ببینیم به چه شکل می توان یک Interface ایجاد کرد. برای این منظور پروژه ای تحت عنوان 52nd Session ایجاد کرده و فایلی در آن تحت عنوان ProgrammerOne ایجاد می کنیم:

```
public interface ProgrammerOne {
	public void designTheUI();
}
```

همانطور که در کد فوق ملاحظه می شود برای ساخت یک Interface می بایست از کلید واژه interface استفاده کنیم. سپس بایستی یک نام برای آن انتخاب کنیم که در مثال فوق نام ProgrammerOne به معنی "برنامه نویس یک" انتخاب شده است (اگر به خاطر داشته باشیم در آموزش گذشته مثالی زدیم مبنی بر این که فرض کنیم که عضو یک تیم برنامه نویسی هستیم که در آن هر برنامه نویس وظیفه ای خاص دارد).

کلید واژه public هم که پیش از کلید واژه interface قرار گرفته است حاکی از آن است که این Interface توسط هر کلاسی که در هر پکیجی قرار داشته باشد قابل استفاده است.

لازم به ذکر است همانطور که در آموزش پیش توضیح داده شد نام فایلی که حاوی این Interface است می بایست با نام انتخابی ما یکی باشد بنابراین نام این Interface معادل با ProgrammerOne.java است. سپس همانند ساخت یک کلاس از دو علامت {} استفاده کرده و متدهایی که نیاز داشته باشیم را میان آن دو تعریف می کنیم. به طور مثال در کد فوق یک متد تحت عنوان designTheUI به معنی "طراحی کردن رابط گرافیکی کاربر" تعریف کرده

ایم. اگر توجه کرده باشیم متدی که داخل Interface ایجاد کرده ایم بر خلاف متدهایی که داخل کلاس تعریف می کنیم دارای {} نیست و این مسئله از آنجا ناشی می شود که متدهای داخل یک Interface فقط دارای Signature هستند. به عبارت دیگر متدهای داخل Interface فقط به صورت انتزاعی تعریف می شوند سپس داخل کلاسی که قرار است آن Interface را اجرا کند وظایف متد را داخل دو علامت {} مرتبط با آن تعریف می کنیم.

جالب است بدانیم کلیه متدهایی که داخل یک Interface ایجاد می شوند public خواهند بود بنابراین نوشتن یا ننوشتن این کلید واژه برای متدهای داخل یک Interface تفاوتی ایجاد نخواهد کرد.

حال نیاز داریم کلاسی ایجاد کنیم که این وظیفه را دارا است تا Interface یی که ایجاد کرده ایم را "اجرا" کند. برای این منظور کلاسی تحت عنوان ActionClass به معنی "کلاس اجرایی" ایجاد می کنیم (به خاطر داشته باشیم که در حین ساخت این کلاس گزینه public static void main را تیک دار نماییم):

```
public class ActionClass {

    public static void main(String[] args) {

    }
}
```

در زبان برنامه نویسی جاوا هر زمانیکه بخواهیم به یک کلاس دستور دهیم که یک Interface را اجرا کند می بایست از کلید واژه implements به معنی "اجرا کردن" استفاده نماییم. در واقع همانطور که از کلید واژه extends به منظور وراثت داشتن استفاده می کنیم، کلید واژه implements را نوشته سپس نام Interface را که تمایل داریم کلاس ما آن را اجرا کند را می نویسیم:

```
public class ActionClass implements ProgrammerOne{

	public static void main(String[] args) {

	}
}
```

اکنون نیاز داریم تا متدی که به صورت انتزاعی در Interface خود ایجاد کردیم را فرا خوانده و وظیفه ای برای آن تعریف کنیم:

```
public class ActionClass implements ProgrammerOne {
	@Override
	public void designTheUI() {
		System.out.println("Programmer one has to design the UI");
	}

	public static void main(String[] args) {

	}
}
```

همانطور که در کد فوق ملاحظه می شود در داخل کلاس اصلی برنامه متدی تحت عنوان designTheUI را که داخل Interface خود ایجاد کرده ایم را فرا خوانده ایم. اگر توجه کرده باشیم ما داخل Interface فقط این متد را به صورت انتزاعی ایجاد کردیم و وظیفه ای برای آن تعریف نکردیم اما این در حالی است که داخل کلاس وظیفه آن را به صراحت مشخص می کنیم که حاکی از آن است که عبارت Programmer one had to design the UI به معنی "برنامه نویس شماره یک بایستی رابط گرافیکی را طراحی کند" را روی صفحه به نمایش در آورد. نکته ای که در اینجا حائز اهمیت است این است که پس از وارد کردن این متد داخل برنامه اکلیپس از ما ایراد خواهد گرفت که بایستی متد را Override کرد (برای آشنایی بیشتر با دستور Override به آموزش سی و سوم مراجعه نمایید). به طور خلاصه علت اجباری بودن Override این است که از ایجاد مشکلات احتمالی جلوگیری به عمل آید.

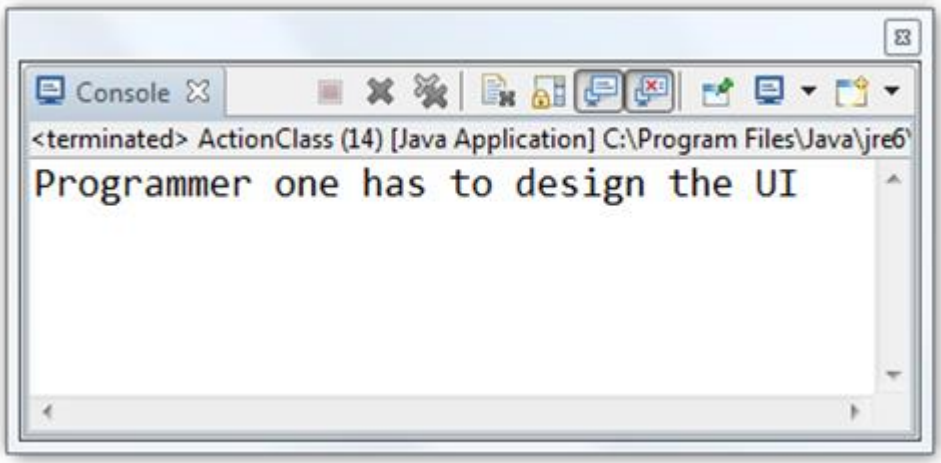
حال کد خود را به صورت زیر تکمیل می کنیم:

```
public class ActionClass implements ProgrammerOne {
	@Override
	public void designTheUI() {
		System.out.println("Programmer one has to design the UI");
	}

	public static void main(String[] args) {
		ActionClass test = new ActionClass();
		test.designTheUI();

	}
}
```

در واقع برای آنکه برنامه ما اجرا شود نیاز داریم تا به تکمیل متد main بپردازیم زیرا همانطور که قبلاً توضیح داده شده است این متد به منزله نقطه شروع برنامه خواهد بود. کاری که در متد main انجام داده ایم این است که یک شیئ تحت عنوان test از روی کلاس ActionClass ایجاد کرده ایم. در واقع برای این منظور می بایست اول نام کلاس خود را نوشته سپس نامی برای آن در نظر بگیریم و یک علامت مساوی قرار می دهیم. حال از آنجا که می خواهیم یک شیئ جدید ایجاد کنیم می بایست کلید واژه new را بنویسیم و سپس مجدد نام کلاسی که می خواهیم از روی آن یک شیئ جدید ایجاد کنیم را خواهیم نوشت و یک علامت ;() در انتها قرار می دهیم. از این پس به کلیه متدها، Interface ها و دیگر کلاس های مرتبط با این کلاس دسترسی خواهیم داشت. حال نام شیئ ساخته شده از روی کلاس ActionClass را نوشته یک نقطه قرار می دهیم و سپس نام متدی که در Interface ایجاد کردیم را می نویسیم. پس از اجرای برنامه خروجی زیر مشاهده خواهد شد:



همانطور که می بینیم با موفقیت توانستیم یک Interface ایجاد کنیم سپس آن را در کلاس خود اصطلاحاً implements کرده و در نهایت با ساخت یک Object از روی کلاس خود توانستیم برنامه ای که حاوی یک Interface و یک Class و یک Method بود را اجرا نماییم. در پایان این نکته را فراموش نکنیم که کلاسی که دست به اجرای یک Interface می زند می بایست کلیه متدهای موجود در آن Interface را نیز اجرا نماید.

پس از مطالعه این آموزش انتظار می رود بتوانیم به سؤالات زیر پاسخ بدهیم:

۱. به منظور ساخت یک Interface از چه کلید واژه ای استفاده می کنیم؟
۲. نحوه نامگذاری یک Interface به چه شکل است؟
۳. پسوند فایلی که حاوی یک Interface است چیست؟
۴. چگونه می توانیم یک Interface را در یک کلاس اجرا کنیم؟