به نام خدا

آموزش سی و یکم

اهداف آموزشی این قسمت عبارتند از:

۱. آشنایی با مفهوم شیئ در زبان برنامه نویسی جاوا
۲. آشنایی با مزایای استفاده از شیئ ها در زبان جاوا
۳. آشنایی با اجزای تشکیل دهنده یک شیئ
۴. وظایف متدها در یک برنامه
۵. نحوه نامگذاری متدها
۶. آشنایی با نحوه ساخت یک کلاس جدید

در دنیای واقعی یک شیئ جدای از دیگر اشیاء می تواند وجود داشته باشد، با دیگر اشیاء تعامل داشته باشد و حتی چند شیئ با یکدیگر ادغام شده تا شیئی جدیدی را بوجود آورند. در واقع قسمت سی و یکم از آموزش زبان برنامه نویسی جاوا به منزله شروع کار با Object ها یا همان اشیاء در این زبان برنامه نویسی خواهد بود. از اینجای آموزش به بعد خواهیم دید که چه طور می توان یک Object با خصوصیات منحصر به فردی ایجاد کرده و همچنین خواهیم دید که به چه نحوی می توان آن شیئ را در دیگر جاهای برنامه خود مورد استفاده قرار داد. در حقیقت اگر یک برنامه نوشته شده در جاوا را به منزله تعدادی Object در نظر بگیریم که در کنار یکدیگر قرار گرفته اند تا یک Object بزرگ تر از خود را که کار خاصی را می تواند انجام دهد بسازد(یک برنامه)، به سهولت خواهیم توانست Object ها را در یک زبان برنامه نوسی شیئ گرا درک کنیم. در واقع با ساخت یک شیئ در برنامه خود، خواهیم توانست آن شیئ را به هر تعداد دفعه که بخواهیم مورد استفاده قرار دهیم و یا حتی چنانچه نیاز به تغییری در یکی از Attribute ها یا همان خصوصیات شیئ خاص داشته باشیم، صرفاً با اعمال تغییر در آن شیئ به خصوص تغییر اعمال شده در کل برنامه لحاظ خواهد شد. از سوی دیگر با ایجاد Object ها در یک برنامه کامپیوتری امکان Debug کردن یا مشکل یابی نیز برای ما راحت تر خواهد شد چرا که بجای درگیر کردن خود با کل پروژه، صرفاً می توانیم Object یی که به آن مشکوک هستیم را Debug کنیم.

در حقیقت یک شیئ در زبان برنامه نویسی شیئ گرا دارای دو چیز تحت عنوان Attribute و Behavior می باشد. به طور خلاصه Attribute ها عبارتند از ویژگی هایی که یک شیئ را از شیئ دیگری در برنامه متفاوت می سازند. منظور از Behavior ها همان کارهایی است که یک Object می تواند انجام دهد(به منظور آشنایی بیشتر با شیئ گرایی در زبان های برنامه نویسی، به آموزش سوم مراجعه نمایید). در زبان برنامه نویسی جاوا Object ها با استفاده از یک Class ایجاد می شوند. در حقیقت یک Class در برگیرنده کلیه Attribute ها و Behavior هایی است که یک Object خواهد داشت.

اکنون نیاز است پروژه ای تحت عنوان OOP به معنی **"برنامه نویسی شیئ گرایی"** ایجاد می کنیم سپس کلاسی به نام MyClass در آن می سازیم. در این مرحله کد ما می بایست به شکل زیر باشد:

```
public class MyClass {

}
```

نام این کلاس MyClass به معنی **"کلاس من"** می باشد. حال اگر یک Object از روی این کلاس ایجاد کنیم، آن Object هیچ کاری برای ما نخواهد کرد زیرا که دارای هیچ گونه Attribute و یا Behavior نمی باشد. معنی لغوی واژه public معادل است با **"عمومی"** و یا **"همگانی"** و در زبان برنامه نویسی جاوا نیز معنی این کلید واژه چیزی در حد همین معانی است. با قرار دادن کلید واژه public قبل از نام یک کلاس و یا حتی متد، این دستور را به کامپیوتر می دهیم که این کلاس یا متد قابل دسترسی توسط دیگر بخش های برنامه نیز خواهد بود(در آموزش سی و ششم با دیگر سطوح کنترل همچون private و protected نیز آشنا خواهیم شد). حال نیاز داریم تا یک Attribute برای شیئ خود تعریف کنیم. این کار را می توان از طریق انواع متغیرها که در جلسه آموزشی پنجم مورد بررسی قرار گرفتند انجام داد. از این رو، یک متغیر از جنس int در کلاس خود به اسم number به معنی **"شماره"** ایجاد کرده و Value اختصاص داده شده به آن را معادل با 1 در نظر می گیریم. حال نوبت به ایجاد یک Behavior برای کلاس خود می رسد و برای همین منظور از یک متد استفاده خواهیم کرد. به طور خلاصه وظیفه Method ها انجام یکسری کارها است که به آن ها محول می شود و در این برنامه وظیفه متدی

که ما می سازیم این است که روی صفحه مانیتور جمله ای را به نمایش در آورد. متد ما به شکل زیر خواهد بود:

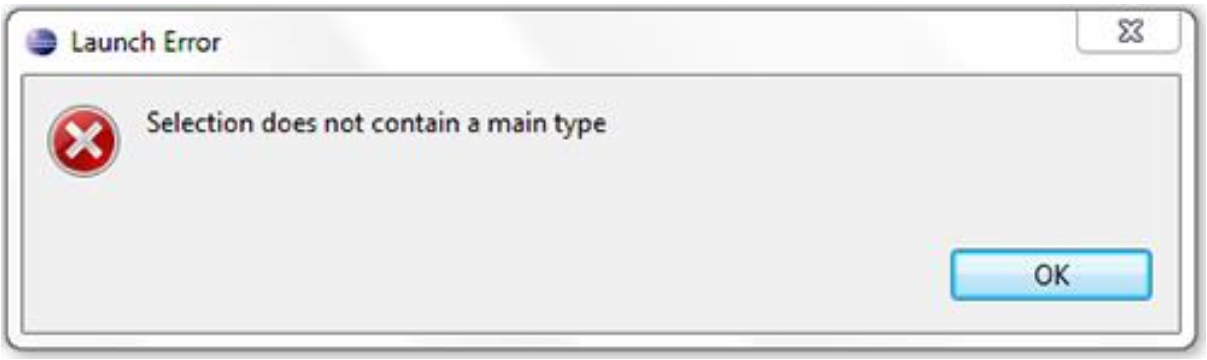
```
public void showSomething() {

	}
```

در واقع با قرار دادن کلید واژه public قبل از نام متد این دستور را به برنامه خود می دهیم که این متد از هر کجای برنامه قابل دسترس خواهد بود. کلید واژه void به این معنی است که این متد از یک سو قرار است دستوری انجام دهد و از سوی دیگر این متد قرار نیست که هیچ گونه Data یا داده ای را باز گرداند. به عبارت دیگر این متد return نخواهد داشت(در آموزش سی و نهم با مبحث return به طور مفصل آشنا خواهیم شد). سپس نام متد تحت عنوان showSomething با معنی **"چیزی را نشان بده"** قرار می دهیم. نکته ای که برنامه نویسان کل دنیا در نام گذاری متد ها همواره مد نظر دارند این است که این نام گذاری می بایست با یک فعل شروع شود چرا که وظیفه اصلی متد های انجام یکسری کار است پس بهتر است که نحوه نامگذاری آن ها هم به نحوی باشد که گویای وظیفه شان باشد. در این مثال از آنجا که این متد قرار است متنی را روی صفحه مانیتور به نمایش در آورد، پس نام آن را نیز با واژه show به معنی **"نشان دادن"** آغاز می کنیم. پس از نام متد علامت () را قرار داده سپس علامت { } را قرار می دهیم(هر کاری که متدها قرار است انجام دهند می بایست مابین دو علامت { } قرا گیرند و چنانچه نیاز است متد دارای Attribute های اضافی باشد می بایست داخل ( ) قرار گیرند که تحت عنوان پارامترهای آن متد شناخته می شوند). حال داخل دو علامت { } دستوری را می نویسیم که جمله **"این اولین متد ساخت من است"** را روی صفحه مانیتور به نمایش در آورد. در این مرحله کد ما به شکل زیر تکمیل خواهد شد:

```
public class MyClass {
	int number = 1;
	public void showSomething() {
	System.out.println("This is my method " + number + " created by
me.");
	}
}
```

در واقع دستوری که داخل System.out.println(); قرار گرفته است در وهله اول عبارت This is my method را روی صفحه مانیتور نوشته سپس مقدار متغیر برنامه تحت عنوان number را به آن اضافه کرده و در نهایت عبارت .created by me را به آخر string اضافه می کند. حال در محیط برنامه نویسی اکلیپس دکمه Run را برای اجرای برنامه می زنیم و خروجی برنامه همانند تصویر زیر خواهد بود:

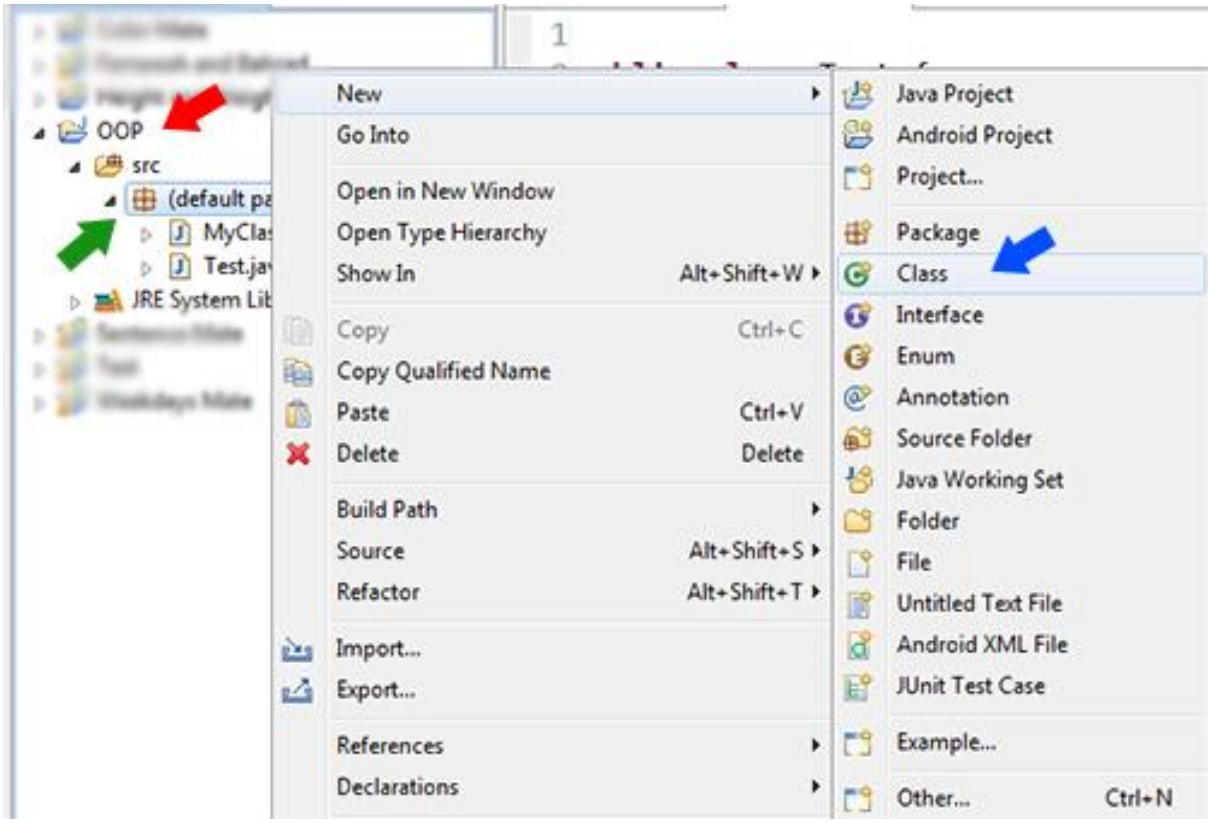


ظاهراً برنامه ما با مشکلی مواجه می باشد چرا که اکلیپس از اجرای برنامه خودداری کرده و از برنامه ما خطا می گیرد. عبارتی که در تصویر مشاهده می شود حاکی از آن است که **"کلاس ما دارای یک نقطه شروع نمی باشد"**. همانطورکه قبلا هم توضیح داده شد، یک برنامه جاوا برای آن بتواند اجرا شود نیاز به یک متد از جنس main دارد که در قسمت زیر قابل مشاهده است:

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

}
```

در واقع هرآنچه که در این متد قرار گیرد، به محض اجرای برنامه فرا خوانده خواهد شد. تا این جای کار ما یک کلاس ایجاد کرده ایم که می توانیم از روی آن یک Object یا شیئ ایجاد کنیم. برای این منظور نیاز داریم تا یک کلاس جدید ایجاد کرده که به منزله نقطه آغازین برنامه ما خواهد بود و در کلاس جدید خود از روی کلاس MyClass یک شیئ جدید ایجاد کرده و مورد استفاده قرار دهیم. برای ایجاد یک کلاس جدید همانند تصویر زیر عمل می کنیم:



در این تصویر فلش قرمز رنگ نشان دهنده نام پروژه ای است که در ابتدای هر آموزش ایجاد می کنیم که در این آموزش نام OOP به معنی **"برنامه نویسی شیئ گرایی"** در نظر گرفته شده است. فلش سبز رنگ نشانگر پکیجی است که در حین ایجاد یک کلاس جدید می توانیم ایجاد کنیم. به خاطر داشته باشیم که وظیفه هر پکیج این است که به ما کمک کند تا کلاس هایی را که نیاز است تا در یک گروه باشند را در کنار یکدیگر قرار دهیم. حال اگر در حین ساخت یک کلاس نام پکیجی در نظر گرفته نشود، اکلیپس به طور خودکار این کار را تحت عنوان default package به معنی **"پکیج پیش فرض"** انجام خواهد داد. در این مرحله روی نام پیکج کلیک راست کرده و از پنجره ای که باز می شود گزینه New را انتخاب کرده و از پنجره دوم گزینه Class را انتخاب می کنیم که در تصویر فوق با فلش آبی رنگ نشان داده شده است.(به منظور ساخت یک کلاس جدید از منوی File می توان گزینه New سپس گزینه Class را انتخاب کرد). حال نامی که برای کلاس خود در نظر می گیریم می تواند MySecondClass به معنی **"کلاس دوم من"** باشد. سپس پیش از زدن دکمه Finish گزینه public static void

main(String[] args) را تیک می زنیم. فایده این کار این است که دیگر نیازی نیست تا متد main را به طور دستی وارد کلاس جدید خود کنیم. پس از ایجاد کلاس جدید، کلاس به طور خودکار در پنجره ای جدید باز شده کد داخل آن به شکل زیر خواهد بود:

```
public class MySecondClass {
	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
	}
}
```

همانطور که قبلا توضیح داده شده است، کلیه Comment ها از داخل کد قابل حذف بوده و هیچ تاثیری در فرایند اجرای برنامه نخواهند داشت. این کلاس جدیدی که ایجاد کردیم بر خلاف کلاس اول خود، این قابلیت را دارا است که هر آنچه در آن قرار گیرد را اجرا کند چرا که داخل آن متد main گنجانده شده است. حال می خواهیم تا یک Object از روی کلاس قبلی خود در این کلاس ایجاد کنیم. برای این منظور کد خود را به شکل زیر تکمیل می کنیم:

```
public class MySecondClass {

	public static void main(String[] args) {
		MyClass newObject = new MyClass();
	}
}
```

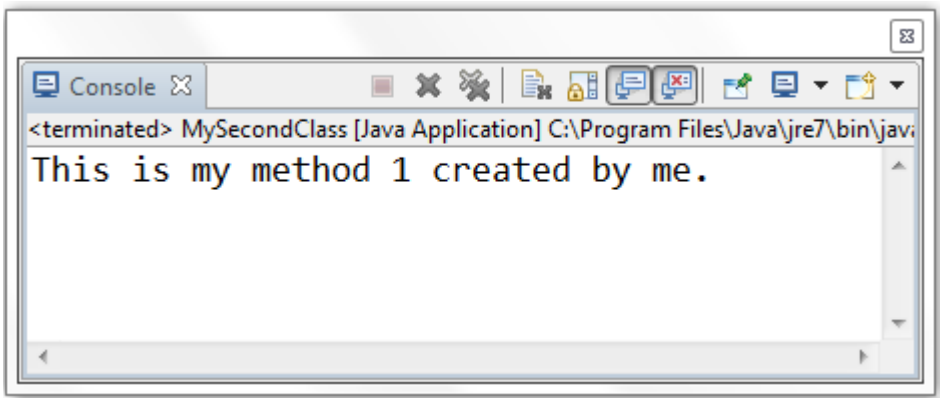
برای این کار نام کلاس قبلی خود را نوشته سپس برای این Object یی که می خواهیم بسازیم یک نام می بایست در نظر بگیریم. نامی که در اینجا می توان در نظر گرفت newObject به معنی "**شیئ جدید** " می باشد(لازم به ذکر است که هر نامی برای این شیئ می توان در نظر گرفت). سپس یک علامت مساوی قرار داده و از آنجا که قرار است یک شیئ جدید ایجاد کنیم از کلید واژه new استفاده خواهیم کرد و در نهایت مجدداً نام کلاسی که می خواهیم از روی آن یک شیئ درست کنیم را نوشته و یک علامت ;() هم در انتهای آن قرار می دهیم. اما برای آنکه این شیئ ساخته شده را بتوانیم مورد استفاده قرار دهیم، می بایست نام Object یی که ساخته ایم را نوشته سپس بعد از آن یک نقطه قرار داده و در نهایت نام متدی که در کلاس قبلی ایجاد کرده بودیم را بنویسیم که در این صورت کد ما به شکل زیر در خواهد آمد(به عبارت دیگر این متد

داخل کلاس است که می توان کاری مثل نمایش یک متن را عملی سازد که بدون فرا خواندن این متد شیئ ما هیچ کاری نخواد کرد):

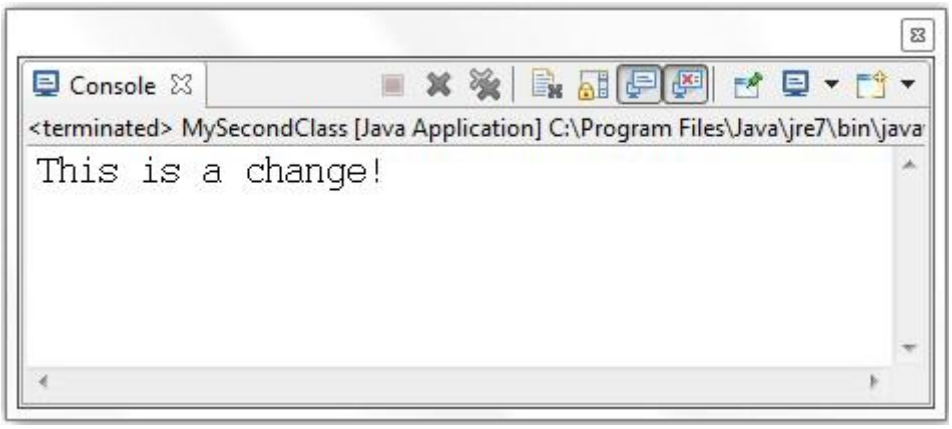
```
public class MySecondClass {

	public static void main(String[] args) {
		MyClass newObject = new MyClass();
		newObject.showSomething();
	}
}
```

تا اینجای کار ما توانسته ایم با موفقیت یک شیئ جدید بسازیم که کلیه Attribute ها و Behavior های خود را از کلاس MyClass به ارث خواهد برد. در این آموزش ما توانستیم یک Object جدید ایجاد کرده که در صورت اجرای برنامه، خروجی آن به شکل زیر خواهد بود:



حال اگر به Class اول خود بازگردیم و تغییری در متد آن ایجاد کنیم، این تغییر بلافاصله در Object یی که از روی آن کلاس ایجاد کردیم اعمال خواهد شد. به طور مثال عبارت داخل دستور System.out.println(); که در کلاس MyClass قرار دارد را به عبارت This is a change! به معنی **"این یک تغییر است"** تبدیل می کنیم. حال پس از Save کردن این تغییر اعمال شده مجدداً به Class دوم رفته و برنامه را اجرا می کنیم. خروجی برنامه به شکل زیر خواهد بود:



مشاهده می کنیم که به چه راحتی می توان در زبان برنامه نویسی جاوا یک Object ایجاد کرده و Attribute ها و Behavior های مرتبط با آن را در لحظه تغییر داد و تغییرات لحاظ شده را در کل برنامه خود دید.

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

۱. مزایای استفاده از Object ها یا همان اشیاء در یک برنامه چیست؟
۲. تفاوت Attribute ها و Behavior های یک شیئ چیست؟
۳. نحوه نامگذاری شیئ های ساخته شده از روی یک کلاس به چه شکل است؟
۴. چرا Compiler در حین اجرای MyClass به مشکل خورد؟
۵. منظور از کلید واژه های new و public چیست؟
۶. چرا در نامگذاری متدها اول از یک فعل استفاده می شود؟
۷. وظیفه متد main چیست؟

در قسمت سی و دوم پیرامون Inheritance یا به **ارث بردن** در زبان برنامه نویسی جاوا صحبت خواهیم کرد.