به نام خدا

آموزش سی و هشتم

اهداف آموزشی این قسمت عبارتند از:

۱. معرفی پارامتر در متدهای جاوا
۲.

پس از آشنایی با نحوه ساخت یک متد از پایه، در این قسمت از آموزش قصد داریم تا به بررسی این مسئله بپردازیم که به چه نحوی می توان داده هایی از هر جنس که بخواهیم را در حین ساخت یک متد جدید به عنوان پارامتری برای آن متد در نظر گرفت.

پیش از شروع آموزش، یک پروژه جدید تحت عنوان 38^{th} Session به معنی **"جلسه سی و هشتم"** ایجاد کرده و کلاسی به نام MainClass به معنی **"کلاس اصلی"** در آن ایجاد می کنیم.

حال برای روشن شدن این مطلب نیاز به یک سناریو در دنیای واقعی داریم. فرض کنیم که پدر یا مادرمان از ما می خواهد که " برو به سوپر مارکت". در واقع ما بایستی کاری انجام دهیم. پس می توان این کار را به کاری که یک متد قرار است انجام دهد تشبیه کنیم. به عبارت دیگر اگر این درخواست پدر یا مادر خود را بخواهیم به یک متد در زبان برنامه نویسی جاوا تبدیل کنیم، متدی به شکل goToSuperMarket(); خواهیم داشت. حال در همان لحظه که آماده شده تا به سوپر مارکت برویم، از ما درخواست می شود که "راستی نون هم بخر". در این جا خرید نان به منزله اطلاعاتی اضافی است که در لحظه به دستور "برو به سوپر مارکت" الحاق می شود.

حال در این صورت متد فوق به شکل goToSuperMarket(bread); نوشته خواهد شد. در واقع خرید نان به عنوان پارامتری است که در لحظه به متد **"برو به سوپر مارکت"** اضافه شده و می بایست داخل پرانتز متد ما قرار گیرد. اکنون اگر والدین ما تصمیمشان عوض شود و به جای خرید نان از ما بخواهند که "راستی پنیر بخر" این پارامتر به پنیر تغییر کرده و متد ما به شکل goToSuperMarket(cheese); تغییر پیدا خواهد کرد. حال والدین ما ممکن است از ما

بخواهند که هم نان بخریم و هم پنیر که در این صورت متد ما به شکل goToSuperMarket(bread, cheese); تغییر پیدا خواهد کرد.

اکنون ببینیم که استفاده از پارامترها در یک متد چه فایده ای دارد. در واقع با اضافه کردن یک پارامتر یا لیستی از پارامترها به یک متد، خواهیم توانست کارایی متد خود را افزایش دهیم. به عبارت دیگر به جای آنکه هر دفعه چیز یکسانی را دریافت کنیم(مثلا هر دفعه نان بگیریم)، می توانیم یک بار برای خرید نان به سوپر مارکت رفته، بار دیگر برای خرید پنیر به سوپر مارکت رفته و یا حتی بار دیگری برای خرید بستنی به سوپر مارکت برویم. در واقع زمانیکه ما متد خود را می نویسیم درهمان لحظه می توانیم تصمیم بگیریم که چه چیزی از سوپر مارکت نیاز داریم تا بخریم.

برای روشن تر شدن توضیحات فوق، قصد داریم تا این مفاهیم را در قالب یک پروژه جاوا بیان کنیم. برای این منظور کلاسی را که در ابتدای این آموزش ایجاد کردیم را به شکل زیر ویرایش می کنیم:

```
public class MainClass {
	String mySentence = "Dear son go to supermarket";
	public void goToSuperMarket() {
		System.out.println(mySentence);
	}
}
```

در کد فوق ما یک کلاس از جنس String تحت عنوان mySentence به معنی **"جمله من"** داریم که Value در نظر گرفته شده برای آن عبارت Dear son go to supermarket به معنی **"پسر عزیزم برو به سوپر مارکت"** می باشد. سپس از متدی از جنس public تحت عنوان goToSuperMarket به معنی **"به سوپر مارکت برو"** ساخته ایم که قرار است هیچ گونه return یی نداشته باشد چرا که از جنس void است(در قسمت سی و نهم بیشتر پیرامون مبحث return صحبت خواهیم کرد). کاری که این متد قرار است انجام دهد این است که عبارت قرار گرفته در کلاس String را روی صفحه مانیتور به نمایش در آورد. اکنون می خواهیم پارامتر "یه چیزی هم بخر" را به متد خود اضافه کنیم. برای همین منظور کد فوق را به شکل زیر تکمیل می کنیم:

```
public class MainClass {
	String mySentence = "Dear son go to supermarket";
	public void goToSuperMarket(String something) {
		System.out.println(mySentence);
	}
}
```

همانطور که مشاهده می شود، داخل پرانتز متد خود یک پارامتر از جنس کلاس String تحت عنوان something به معنی "یه چیزی" اضافه کرده ایم. کاری که این پارامتر قرار است انجام دهد این است که در همان لحظه که متد فرا خوانده می شود کلاسی از جنس String تحت عنوان something به معنی **"یه چیزی"** را هم به عنوان پارامتر خود ارسال کند. حال برای آنکه این پارامتر هم در صفحه مانیتور به نمایش در آید، نیاز داریم تا آن را نیز در دستور System.out.println(); بگنجانیم. برای این منظور کد فوق را به شکل زیر تکمیل می کنیم:

```
public class MainClass {
	String mySentence = "Dear son go to supermarket";
	public void goToSuperMarket(String something) {
		System.out.println(mySentence + something);
	}
}
```

در این صورت، به محض اینکه این متد فرا خوانده شود هر آنچه مقداری که برای کلاس String تحت عنوان something در نظر گرفته شود نیز در پنجره Console به نمایش در خواهد آمد.

حال می بایست این متد را فرا بخوانیم، برای این منظور یک کلاس جدید تحت عنوان ActionClass به معنی **"کلاس عملیاتی"** ایجاد کرده و از آنجا که این کلاس جدید قرار است به منزله نقطه شروع برنامه باشد پس نیاز داریم تا در حین ساخت آن گزینه public static void main را تیک بزنیم. حال کد ما می بایست به صورت زیر باشد:

```
public class ActionClass {

	public static void main(String[] args) {

	}
}
```

اکنون در متد main در کلاس جدیدی که ایجاد کردیم یک شیئ از روی کلاس قبلی به صورت زیر می سازیم:

```
public class ActionClass {

	public static void main(String[] args) {
		MainClass myShopping = new MainClass();

	}
}
```

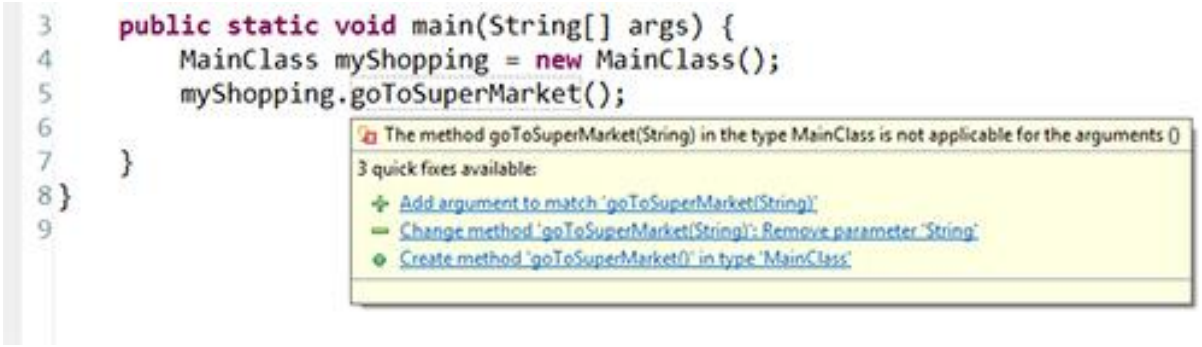
در کد فوق ما نام کلاسی که می خواهیم از روی آن یک شیئ ایجاد کنیم را نوشته سپس نامی برای آن انتخاب کرده که در این مثال نام myShopping به معنی **"خرید من"** در نظر گرفته شده است. سپس یک علامت مساوی قرار داده کلید واژه new را می نویسیم و مجدداً نام کلاسی که می خواهیم شیئ مان از روی آن ساخته شود را می نویسیم و در نهایت یک علامت ;() قرار می دهیم. شیئ ما ساخته شد. اکنون می خواهیم متدی که در کلاسی که از روی آن یک شیئ ساختیم را فرا بخوانیم. برای این منظور کد خود را به شکل زیر تکمیل می کنیم:

```
public class ActionClass {

	public static void main(String[] args) {
		MainClass myShopping = new MainClass();
		myShopping.goToSuperMarket();

	}
}
```

برای این منظور نام شیئ که جدیداً ساخته را نوشته سپس یک نقطه قرار داده و سپس نام متدی که در کلاس اصلی خود ساخته بودیم را می نویسیم و در نهایت علامت ;() را در انتهای نام متد قرار می دهیم. به محض اینکه برنامه را Save کنیم، محیط برنامه نویسی اکلیپس از ما خطایی خواهد گرفت که در تصویر زیر قابل مشاهده است:



در واقع از آنجا که ما در متد خود یک پارامتر از جنس کلاس String تعریف کرده ایم ولیکن هیچ گونه مقداری برای آن پارامتر که از جنس کلاس String بود در نظر نگرفته ایم، همانطور که در باکس زرد رنگ در اولین خط آبی رنگ مشاهده می شود، اکلیپس به ما پیشنهاد می دهد که برای رفع این مشکل می بایست پارامتر از جنس String به متد goToSuperMarket اضافه کنیم.

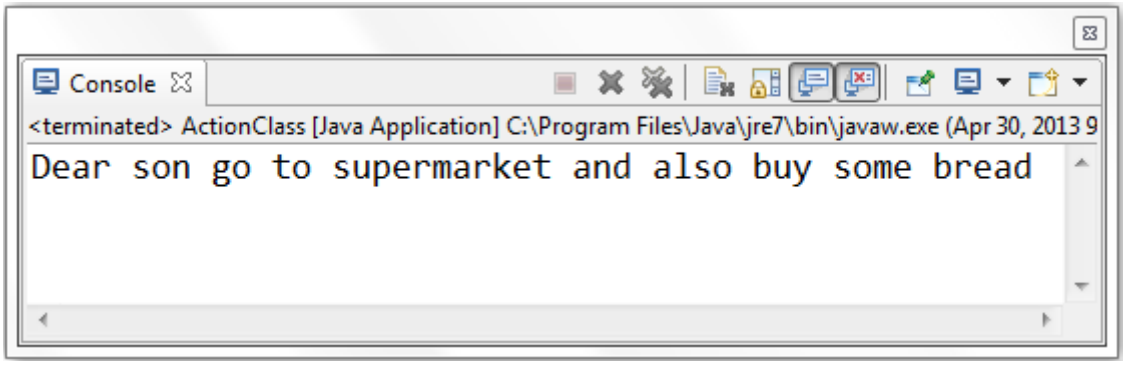
از آنجایی که این پارامتر از جنس کلاس String است و همانطور که می دانیم مقادیر این کلاس می بایست داخل دو علامت " " قرار گیرند، در متد فرا خوانده شده در شیئ ساخته شده صرفاً عبارتی را داخل دو علامت " " قرار داده و داخل پرانتز به شکل زیر می نویسیم:

```
public class ActionClass {

	public static void main(String[] args) {
		MainClass myShopping = new MainClass();
		myShopping.goToSuperMarket(" and also buy some bread");

	}
}
```

همانطور که مشاهده می شود عبارت and also buy some bread به معنی "**و همچنین مقداری نان بخر**" داخل پرانتز نوشته شده است. حال برنامه خود را اجرا می کنیم(اگر توجه کنیم می بینیم که پیش از کلمه and یک فاصله قرار داده ایم. علت اینکار این است که می خواهیم دو جمله به یکدیگر نچسبند):



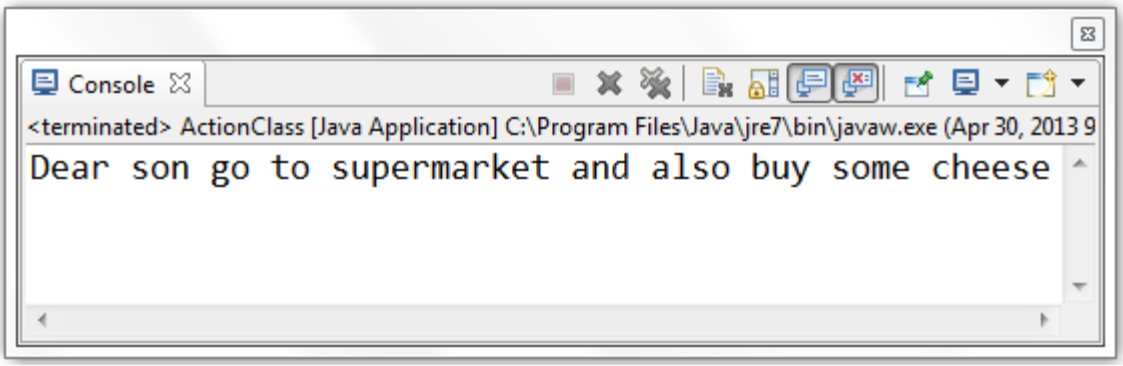
در تصویر فوق مشاهده می شود که String اول تحت عنوان mySentence در بخش اول به صورت Dear son go to supermarket به نمایش در آمده است. سپس پارامتر خود که از جنس String تحت عنوان something بود و در متدمان تعریف کردیم نیز به عبارت قبلی اضافه شده است. در واقع خوبی این پارامتر در اینجا است که فعلاً مقدار آن and also buy some bread می باشد اما این در حالی است که در لحظه پدر یا مادر ما می تواند نظرش را عوض کرده و از ما چیزی دیگر بخواهند که بخریم. به کد زیر توجه کنید:

```
public class ActionClass {

	public static void main(String[] args) {
		MainClass myShopping = new MainClass();
		myShopping.goToSuperMarket(" and also buy some cheese");

	}
}
```

در کد فوق مقدار String را به جمله and also buy some cheese به معنی "**و همچنین مقداری پنیر بخر**" تغییر دادیم. حال مجدد برنامه را اجرا می کنیم:



می بینیم که در لحظه متد ما update شده و جمله جدید را نمایش می دهد.

نکته ای که در اینجا حائز اهمیت است این است که نوع پارامتر تعیین شده برای متد بسیار تعیین کننده است که چه نوع داده ای به همراه آن متد در زمانیکه فرا خوانده می شود ارسال شود. در مثال فوق ما نوع داده را String قرار داده ایم. حال اگر نوع این داده متغیری از جنس int بود، می بایست داده ای از جنس عدد صحیح برای پارامتر متد خود در نظر بگیریم. نکته ای که در اینجا حائز اهمیت است این است که اگر ما بیش از یک پارامتر را برای متد خود در نظر بگیریم، حتما بایستی به همان ترتیبی که پارامترها را قرار داده ایم، مقادیر آن ها را وارد کنیم. به طور مثال اگر پارامتر اول ما از جنس String است و پارامتر دوم ما از جنس int است پس می بایست داده اولیه ما در حین فرا خواندن متد از جنس string بوده و داده دوم ما یک عدد صحیح باشد.

در قسمت سی و نهم پیرامون مبحث return در زمانیکه متد ما پس از اجرا داده ای را به همراه خود باز می گرداند صحبت خواهیم کرد.

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

۱. خوبی استفاده از پارامترها در برنامه نویسی جاوا چیست؟
۲. یک پارامتر به چه شکل ایجاد می شود؟
۳. اگر برای متدی یک پارامتر در نظر بگیریم ولی در حین استفاده از آن پارامتری برای متد تعریف نکنیم چه اتفاقی می افتد؟
۴. اگر بخواهیم بیش از یک پارامتر برای متدها در نظر بگیریم، به چه شکل می توانیم این کار را انجام دهیم؟