به نام خدا

آموزش چهلم

اهداف آموزشی این قسمت عبارتند از:

۱. معرفی Local Variable ها و Field ها در زبان جاوا
۲. تفاوت متغیرهای Local و Field
۳. معرفی Constructor در زبان جاوا

پس از آشنایی با مفهوم متد به طور کلی در آموزش های سی و هفتم، سی و هشتم و سی و نهم، حال نوبت به بررسی مفهوم Constructor ها در زبان برنامه نویسی جاوا می رسد اما پیش از آشنایی با Constructor ها لازم است تا تفاوت مابین Field Variable ها و Local Variable ها را درک کرده باشیم. برای همین منظور پروژه ای تحت عنوان 40th Session به معنی **"جلسه چهلم"** در محیط اکلیپس ایجاد کرده و کلاسی تحت عنوان Constructor در آن می سازیم. اکنون کد برنامه می بایست به صورت زیر باشد:

```
public class Constructor {

}
```

به طور کلی چنانچه ما متغیری را داخل یک متد تعریف کنیم، آن متغیر یک Local Variable خواهد بود. به عبارت دیگر فقط قابل دسترسی داخل آن متد بوده و متدهای دیگر نمی توانند آن متغیر را مورد استفاده قرار دهند. برای روشن شدن مطلب به مثال زیر توجه کنید:

```
public class Constructor {
	public void testMethodOne() {
		int number;
	}

}
```

در کد فوق ما یک متد تحت عنوان testMethodOne داریم که دارای متغیری از جنس int تحت عنوان number به معنی **"عدد"** است. در حقیقت متغیر int یک Local Variable یا

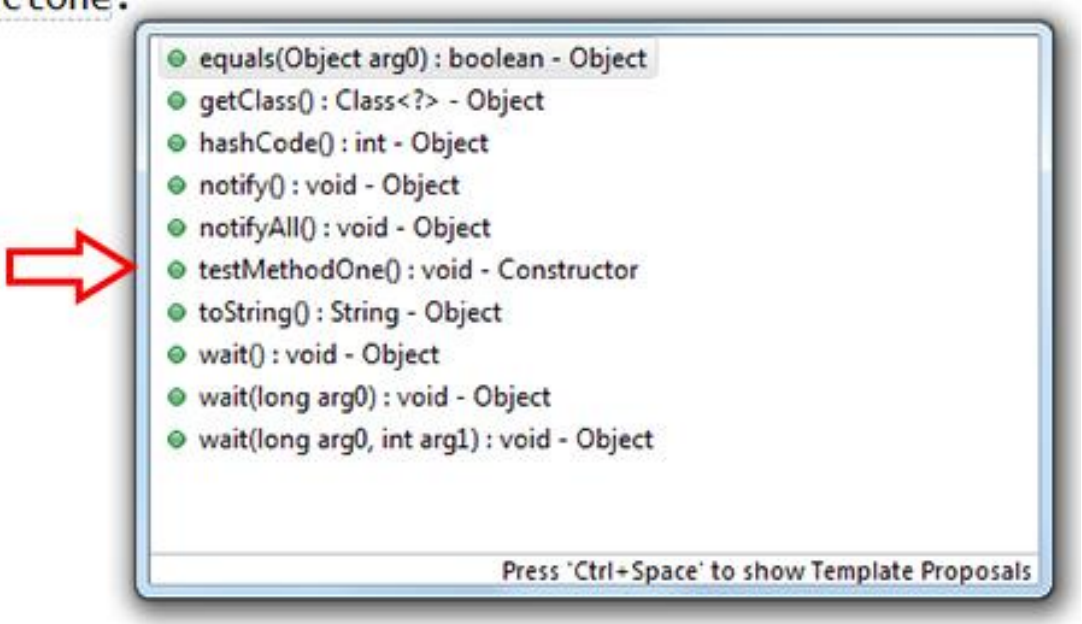
به عبارتی یک متغیر داخلی است و فقط متد testMethodOne به آن دسترسی خواهد داشت.
از سوی دیگر اگر یک کلاس جدید ایجاد کنیم که از کلاس Constructor ارث بری کند، سپس یک شیئ از روی کلاس Constructor در آن بسازیم، صرفا ما به خود متد testMethodOne دسترسی خواهیم داشت و نخواهیم توانست کنترل متغیر int داخل این متد را به صورت اختصاصی در دست بگیریم. برای روشن شدن این مطلب یک کلاس دیگر تحت عنوان ActionClass به معنی **"کلاس عملیاتی"** ایجاد می کنیم که از کلاس Constructor ارث بری می کند. سپس یک شیئ از روی کلاس Constructor تحت عنوان objectOne به معنی **"شیئ یک"** می سازیم. حال کد ما می بایست به صورت زیر باشد:

```
public class ActionClass extends Constructor {
      public static void main(String[] args) {
            Constructor objectOne = new Constructor();

      }
}
```

حال می خواهیم تا به متغیر داخل متد testMethodOne که داخل کلاس Constructor ایجاد کرده است دست یابیم. برای این منظور نام شیئ ایجاد شده از روی کلاس Constructor که همان objectOne است را نوشته سپس یک نقطه قرار می دهیم:

```
class ActionClass extends Constructor {
blic static void main(String[] args) {
  Constructor objectOne = new Constructor();
  objectOne.
```



همانطور که در تصویر فوق مشاهده می شود، به محض اینکه یک نقطه پس از نام شیئ ایجاد شده قرار می دهیم، نرم افزار اکلیپس به صورت خودکار نام چیزهایی که برای این شیئ قابل دسترسی هستند را برای ما داخل پنجره ای نمایش می دهد. همانطور که فلش قرمز رنگ نشان می دهد، از کلاس Constructor ما فقط به متد testMethodOne دسترسی داریم و به هیچ وجه متغیر int که داخل آن متد قرار داده ایم در لیست چیزهایی که به آن دسترسی داریم وجود نخواهد داشت.

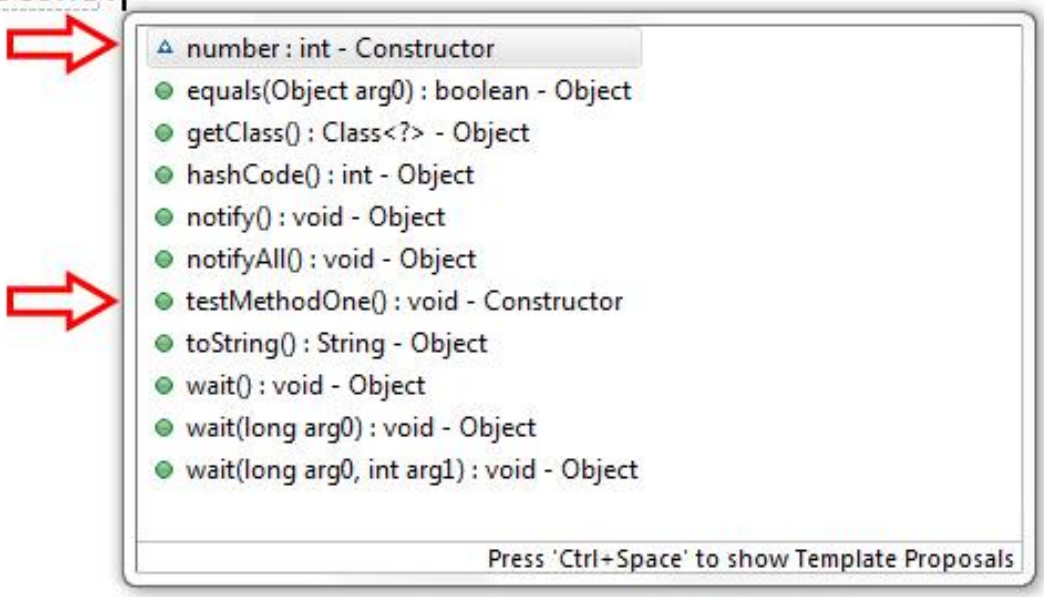
حال اگر متغیری را خارج از یک متد تعریف کنیم، این متغیر به مثابه یک Field Variable خواهد بود. به عبارت دیگر از دیگر بخش های برنامه نیز قابل دسترسی خواهد بود. برای روشن شدن مطالب فوق کد خود را به شکل زیر بازنویسی می کنیم:

```
public class Constructor {
      int number;
      public void testMethodOne() {

      }
}
```

در کد فوق ما یک متغیر از جنس int داریم که داخل خود کلاس تعریف شده است. به عبارت دیگر این متغیر برای کلیه متدهای داخل کلاس ما قابل دسترسی خواهند بود. برای روشن شدن این مطلب، مجدداً به کلاس دومی که ساختیم باز می گردیم. این بار به محض اینکه نام شیئ ساخته شده از روی کلاس Constructor را می نویسیم و یک نقطه قرار می دهیم، می بینیم که علاوه بر متد testMethodOne نرم افزار اکلیپس متغیری تحت عنوان number که در کلاس Constructor ساخته شده است را نیز نشان می دهد:

```
class ActionClass extends Constructor {
blic static void main(String[] args) {
 Constructor objectOne = new Constructor();
 objectOne.
```



در حقیقت این مسئله نشانگر این است که متغیرهایی از جنس Field یا به عبارت دیگر هر متغیری که در خود کلاس تعریف شود، برای دیگر بخش های برنامه نیز قابل دسترسی خواهد بود.

نکته دیگری که در مورد Field ها همواره می بایست مد نظر داشته باشیم این است که ما به هر تعداد که بخواهیم می توانیم از آن ها در دیگر جاهای برنامه خود استفاده کنیم و این در حالی

است که مقدار در نظر گرفته شده برای آن ها مخصوص همان اشیائی است که از آن Field استفاده می کنند. برای روش شدن مطلب به مثال زیر توجه کنید:

```
public class ActionClass extends Constructor {
	public static void main(String[] args) {
		Constructor objectOne = new Constructor();
		objectOne.number = 12;

		Constructor objectTwo = new Constructor();
		objectTwo.number = 20;

	}
}
```

همانطور که در کد فوق ملاحظه می شود، ما دو شیئ از روی کلاس Constructor ساخته ایم که شیئ اول objectOne و شیئ دوم objectTwo نام دارد. سپس از طریق شیئ اول به Field ساخته شده تحت عنوان number دسترسی پیدا کرده و مقدار آن را معادل با ۱۲ قرار می دهیم. در نهایت از طریق شیئ دوم هم به این Field دسترسی پیدا کرده و مقدار آن را معادل با ۲۰ قرار می دهیم. همانطور که ملاحظه می شود، این دو Value متفاوت با یکدیگر تناقضی ندارند چرا که در هر شیئ ساخته شده از روی کلاس Constructor به طور مجزا مورد استفاده قرار گرفته اند. شاید به همین دلیل باشد که نام دیگری که برای متغیرهای Field در نظر گرفته شده است Instance Variable است. به عبارت دیگر متغیری که در هر Instance یا "نمونه" ای از یک کلاس مورد استفاده قرار می گیرد.

اکنون به نظر می رسد با درک کامل تفاوت مابین Local Variable ها و Field ها بهتر بتوانیم به مبحث Constructor ها در زبان برنامه نویسی جاوا بپردازیم.

چنانچه بخواهیم به طور خلاصه بگوییم که Constructor چیست، بایستی گفت که یک Constructor همانند یک متد است با این تفاوت که نام آن دقیقاً با نام کلاس یکی بوده و از سوی دیگر به هیچ وجه نمی تواند داده ای را return کند. واقعیت دیگری که در مورد Constructor ها صدق می کند این است که یک Constructor می تواند هیچ پارامتری نداشته باشند، یک پارامتر داشته و یا حتی بیش از یک پارامتر داشته باشد. اکنون که با مقدمه ای کوتاه با

مفهوم Constructor ها آشنا شدیم خواهیم توانست در آموزش بعد درک بهتری از این ساختار داشته باشیم.

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

۱. تفاوت متغیرهایی از جنس Local یا Field چیست؟
۲. چگونه می توان یک متغیر Local و یک متغیر Field ایجاد کرد؟
۳. چرا اختصاص مقادیر متفاوت برای متغیرهای Field که در شیئ های مختلف مورد استفاده قرار می گیرند ایجاد تداخل نمی کند؟
۴. منظور از Constructor در زبان جاوا چیست؟