به نام خدا

آموزش هفتم

در این جلسه پیش از تکمیل مبحث String ها در زبان برنامه نویسی جاوا می خواهیم یکی از خصوصیات کلیدی محیط برنامه نویسی اکلیپس را مورد بررسی قرار دهیم. این خصیصه نام های مختلفی همچون Auto Complete, Code Assist, Code Completion, Content Assist دارد. به طور خلاصه اگر بخواهیم در حین کدنویسی از خود محیط برنامه نویسی اکلیپس کمک بگیریم می توانیم از ترکیب کلید های Ctrl و Space استفاده کنیم. با این کار زمانیکه ما در حین نوشتن چیزی مثلاً نام یک متغییر که قبلاً ایجاد کرده ایم هستیم، با هم زمان زدن کلید هایی که در بالا گفته شد از محیط برنامه نویسی اکلیپس می خواهیم تا کد را برای ما تکیمل کند. برای روشن شدن مطلب به مثال زیر توجه کنید:

```
public class Test {
	public static void main(String[] args) {
		String myFistName;
	}
}
```

ما یک String تحت عنوان myFirstName ایجاد کرده ایم که فاقد هر گونه Value یا مقدار است. حال در خط بعد می خواهیم که مقداری برای آن درج کنیم. از این رو شروع به نوشتن نام String خود می کنیم به این صورت که می نویسیم my . حال در این مرحله اصلاً نیازی نیست که کل نام را به صورت دستی وارد کنیم بلکه می توانیم پس از نوشتن دو کاراکتر my کلید های Ctrl و Space را به صورت هم زمان فشار دهیم که در این صورت نام myFirstName به صورت خودکار نوشته می شود. از مزایای استفاده از این ویژگی محیط برنامه نویسی اکلیپس می توان گفت که نه تنها زمان

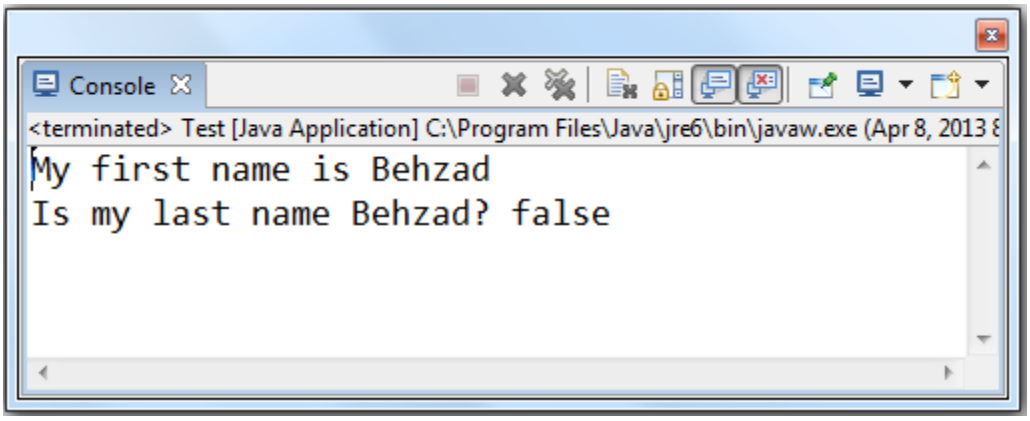
کد نویسی را کاهش داده بلکه میزان خطاهای تایپی نیز به حداقل می رساند. در واقع این ویژگی اکلیپس صرفاً برای تکمیل کردن نام ها و ... نیست بلکه جایی که این قابلیت خیلی زیاد به برنامه نویسان کمک می کند برای تکمیل کد است. در حقیقت از آنجا که قسمت های تشکیل دهنده API زبان برنامه نویسی جاوا بسیار گسترده است(بیش از ۴۰۰۰ مورد)، به سادگی می توان از این ویژگی برای تکمیل کد خود استفاده کنیم به این صورت که صرفاً نیاز است ابتدای نام Method و یا Class مد نظر و یا هر چیزی را به خاطر داشته باشیم و با زدن کلید های Ctrl و Space محیط برنامه نویسی اکلیپس پیشنهادات خود را برای تکمیل کد به ما می دهد و ما به سادگی می توانیم از میان پیشنهادات ارائه شده گزینه مد نظر خود را انتخاب کنیم.

حال که با این ویژگی ارزشمند اکلیپس آشنا شدیم می توانیم به مبحث تکمیلی String ها در زبان برنامه نویسی جاوا بپردازیم. در واقع زمان هایی برای برنامه نویسان جاوا پیش می آید که نیاز دارند تا کلاسی از جنس String را با یکدیگر مقایسه کنند که در این صورت از Method خاصی تحت عنوان equal استفاده می کنند. در واقع در زبان برنامه نویسی جاوا Method راهی برای به انجام رساندن کاری است و کاری که در اینجا مد نظر ما است این است که ببینیم آیا مقدار دو کلاس از جنس String یکی است یا خیر. برای روشن تر شدن مطلب به مثال زیر توجه کنید:

```
public class Test {

    public static void main(String[] args) {
    String name = "Behzad";
    String lastName = "Moradi";
    System.out.println("My first name is " + name);
   System.out.println("Is my last name Behzad: "
   +lastName.equals(name));

    }
}
```

در اینجا ما دو کلاس از جنس String داریم که دارای مقادیر Behzad و Moradi هستند. حال در اولین System.out.println(); ما نیاز داریم تا جمله My name is Behzad به نمایش در آید، برای همین منظور با قرار دادن علامت + و نام String که در برگیرنده مقدار "Behzad" است به راحتی این جمله را در بخش Console به نمایش در می آوریم. در مرحله بعدی ما می خواهیم تا مقدار دو String را با یکدیگر مقایسه کنیم. برای این منظور نام String که می خواهیم با String دیگر مقایسه شود را نوشته سپس یک Method تحت عنوان equals(). به آن اضافه کرده و داخل پرانتز این Method نام String یی را می نویسیم که می خواهیم String قبلی را با آن مقایسه کنیم. به طور خلاصه به صورت lastName.equals(name) می نویسیم.

حال چنانچه برنامه را اجرا کنیم، خروجی زیر را مشاهده خواهیم کرد:

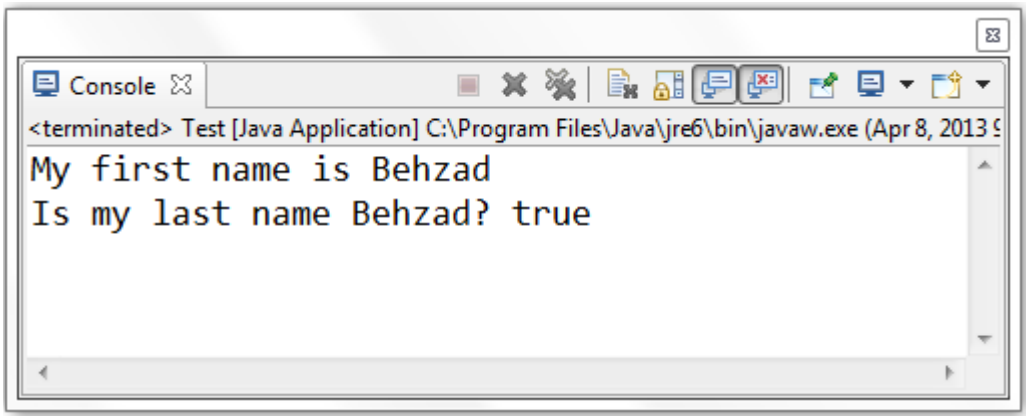


در واقع خروجی Method ما چیزی از جنس boolean خواهد بود که مقدار آن یا true است و یا false و در اینجا از آن رو که مقادیر متغیرهای ما با یکدیگر متفاوت هستند مقدار false به نمایش در خواهد آمد. حال اگر بخواهیم جواب true شود به راحتی می توانیم مقدار متغییر lastName را همانند مقدار متغییر name کنیم و

مشاهده می کنیم که جواب true خواهد بود. در واقع کد ما به شکل زیر تغییر پیدا خواهد کرد:

```
public class Test {

	public static void main(String[] args) {
	String name = "Behzad";
	String lastName = "Behzad";
	System.out.println("My first name is " + name);
	System.out.println("Is my last name Behzad: "
	+lastName.equals(name));

	}
}
```

و در نهایت خروجی برنامه ما پس از اجرا به شکل زیر خواهد بود:

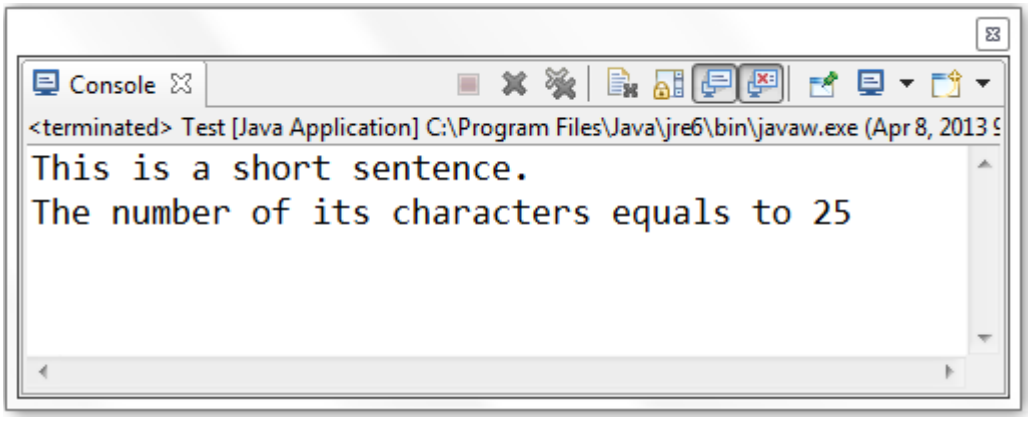


گاهی اوقات برای ما در برنامه نویسی پیش می آید که نیاز داریم تا تعداد کاراکتر های یک String را بشماریم که برای این منظور می توانیم از Method یی تحت عنوان lenght() استفاده کنیم. برای روشن شدن مطلب به مثال زیر توجه کنید:

```
public class Test {

	public static void main(String[] args) {
		String sentence = "This is a short sentence.";

		System.out.println(sentence);
		System.out.println("The number of its characters equals to "
				+ sentence.length());

	}
}
```

در مثال فوق ما یک کلاس از جنس String داریم که نام آن sentence است و مقدار آن جمله This is a short sentence. می باشد. حال در Statement اول ما متغییر خود را به نمایش در می آوریم اما در Statement دوم جمله The number of its characters equals to را به علاوه Method یی می کنیم که تعداد کاراکتر های String ما را بشمارد. برای همین منظور نام String خود را نوشته و نام Method خود که در اینجا lenght() است که قرار است تعداد کاراکتر ها را بشمارد ضمیمه آن می کنیم.(توجه داشته باشیم که این Method حتی جاهای خالی، نقطه، کاما و غیره را نیز در String می شمارد.)

خروجی برنامه به شکل زیر خواهد بود:



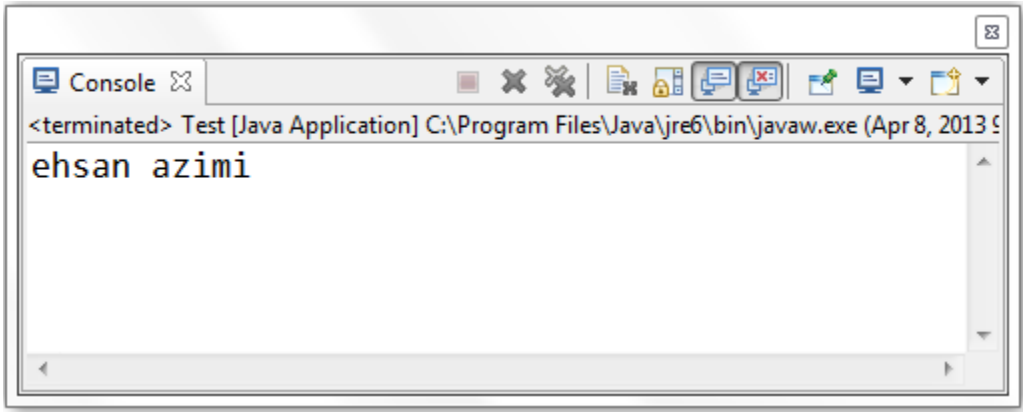
اگر توجه کرده باشید تعداد کاراکتر های جمله ما ۲۰ عدد است اما خروجی برنامه عدد ۲۵ را به نمایش در می آورد و این از آنجا ناشی می شود که Method یی که استفاده کردیم نقطه و فضاهای خالی را نیز شمرده است.

گاهی اوقات ما نیاز داریم تا کاراکتر های یک String را به حروف بزرگ و یا کوچک تبدیل کنیم. از این رو می توانیم از Method های toUpperCase. و toLowerCase. به ترتیب برای بزرگ کردن و کوچک کردن حروف استفاده کنیم.

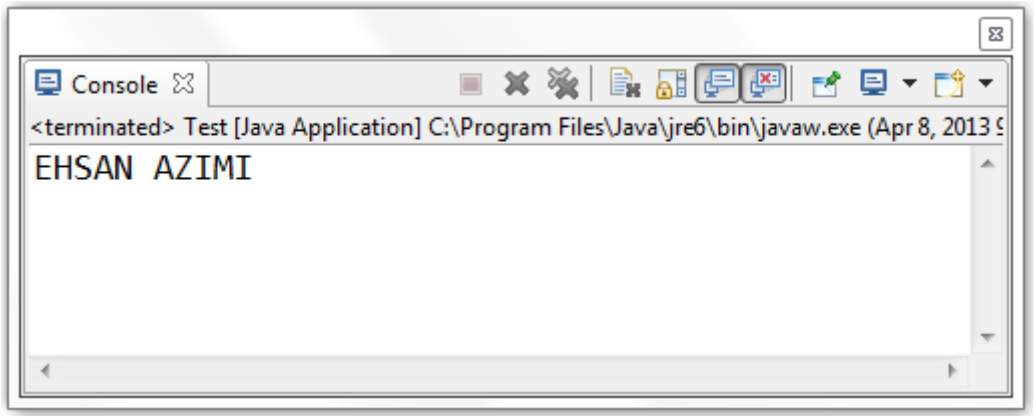
در حقیقت یکی از کاربردهای این Method این است که گاهی اوقات ما نیاز داریم تا کلیه ورودی ها را به صورت حروف بزرگ و یا حروف کوچک ذخیره سازیم. برای این منظور از این Method استفاده می کنیم تا این کار به صورت خودکار انجام شود. (در آموزش های آتی بیشتر از این Method استفاده خواهی کرد). حال فرض کنیم مقدار متغییر name که در مثال های قبلی استفاده کردیم معادل با EhSan AzimI باشد و ما نیاز داریم تا بی دقتی صورت گرفته در حین وارد کردن این متغییر را جبران کنیم. برای این منظور به صورت زیر عمل می کنیم:

```
public class Test {
    public static void main(String[] args) {
        String name = "EhSan AzimI";
        System.out.println(name.toLowerCase());
    }
}
```

در حقیقت با نوشتن نام متغییر و ضمیمه کردن Method یی تحت عنوان toLowerCase. کلیه حروف متغییر را به حروف کوچک تبدیل می کنیم.



عکس این قضیه هم صادق است. به عبارت دیگر برای تبدیل کلیه حروف از Method دیگری تحت عنوان toUpperCase. استفاده می کنیم و خروجی برنامه ما به شکل زیر خواهد بود:



پس از آشنایی با مفاهیم اصلی متغییر ها در این جلسه نوبت به درک مفاهیم Operator ها یا همان اعمال اصلی ریاضیاتی مابین متغییر ها می رسد که در آموزش آینده به تفصیل مورد بحث و بررسی قرار خواهد گرفت.