

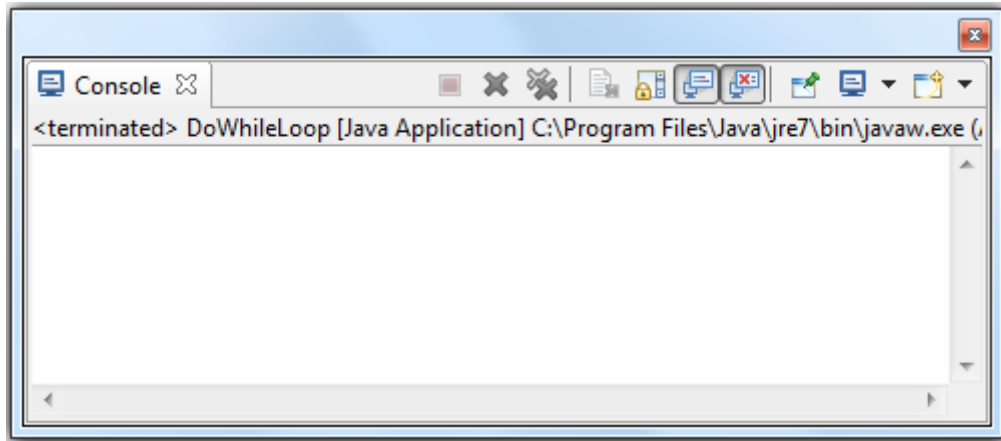
به نام خدا

آموزش بیست و چهارم

پس از آشنایی با نحوه عملکرد Loop هایی از جنس for و while در دو قسمت آموزشی پیشین، حال نوبت به بررسی نوع دیگری از Loop ها تحت عنوان do-while می رسد. وجه تمایز Loop یی از جنس do-while با Loop یی از جنس while در این است که در do-while حتی اگر جواب به شرط داخل پرانتز while() معادل با false باشد، باز هم دستور System.out.println(); حداقل یک بار اجرا می شود و این در حالی است که اگر ما Loop یی از جنس while داشته باشیم و پاسخ به شرط داخل پرانتز while() معادل با false باشد هیچ نوع خروجی از برنامه نباید انتظار داشته باشیم. برای روشن تر شدن این وجه تمایز پروژه ای جدید تحت عنوان 24th Session به معنی "جلسه بیست و چهارم" به همراه کلاسی تحت عنوان DoWhileLoop ایجاد می کنیم و آن را به شکل زیر تکمیل می کنیم:

```
public class DoWhileLoop {  
    public static void main(String[] args) {  
        int number = 12;  
        while (number <= 10) {  
            System.out.println("My Number Is: " + number);  
            number++;  
        }  
    }  
}
```

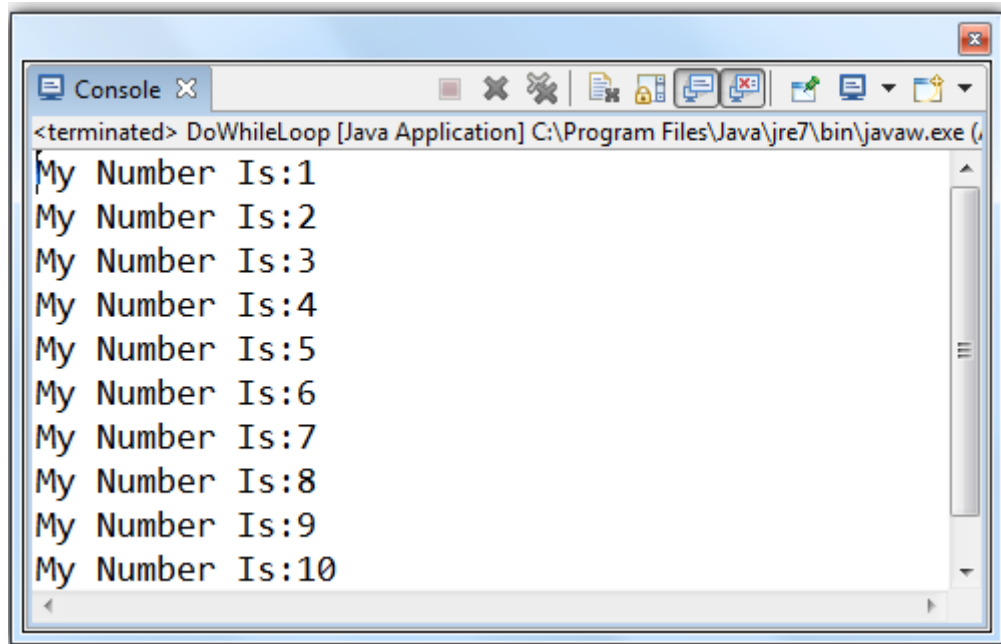
در این کد به برنامه طراحی شده دستور داده ایم مادامیکه مقدار متغیر number کوچکتر یا مساوی با عدد ده است این Loop ادامه پیدا کند اما نکته در اینجا است که مقدار اولیه متغیر معادل با عدد دوازده است و از آنجا که جواب به شرط داخل پرانتز از ابتدا false است، برنامه در پنجره Console هیچ چیز نمایش نخواهد داد و خروجی برنامه به شکل زیر خواهد بود:



حال کد فوق را می خواهیم به Loop یی از جنس do-while تبدیل نماییم. برای این منظور کد خود را به شکل زیر بازنویسی می کنیم:

```
public class DoWhileLoop {  
    public static void main(String[] args) {  
        int number = 1;  
        do {  
            System.out.println("My Number Is:" + number);  
            number++;  
        } while (number <= 10);  
    }  
}
```

در این کد یک متغیر از جنس int تحت عنوان number به مقدار Value اولیه یک تعریف کرده ایم. سپس یک Block به شکل { } ایجاد می کنیم (همانطور که در آموزش های پیشین گفته شد block به یک قسمت کد در برنامه نویسی گفته می شود). حال مابین دو علامت { } مرتبط با do دستوری که می خواهیم روی صفحه مانیتور به نمایش در آید را می نویسیم و سپس دستور می دهیم که در هر بار Loop شدن یک واحد به متغیر number اضافه شود. پس از پایان Block مرتبط با do حال نوبت به نوشتن شرط Loop است که به صورت while(number<=10); می نویسیم. به عبارت دیگر به برنامه دستور می دهیم که مادامیکه مقدار متغیر number کوچکتر یا حداکثر مساوی با عدد ده است این کار ادامه یابد. پس از اجرای برنامه خروجی زیر مشاهده خواهد شد:

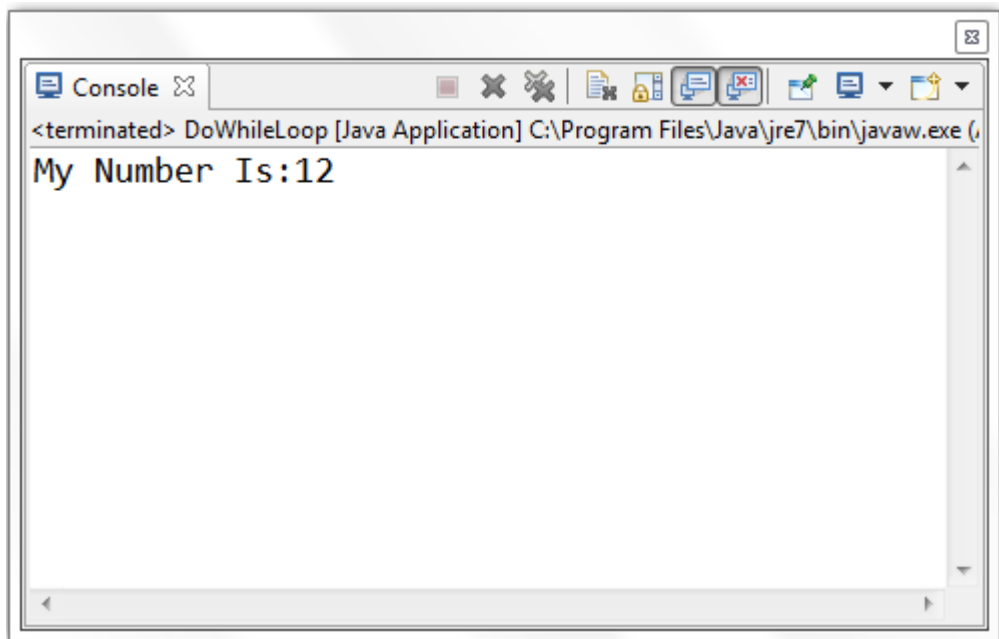


```
<terminated> DoWhileLoop [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (/
My Number Is:1
My Number Is:2
My Number Is:3
My Number Is:4
My Number Is:5
My Number Is:6
My Number Is:7
My Number Is:8
My Number Is:9
My Number Is:10
```

می بینیم که در برنامه فوق Loop یی از جنس do-while عملکردی همانند Loop یی از جنس while دارد. حال با تغییری که در کد خود خواهیم داد وجه تمایز این دو نوع Loop به خوبی مشخص خواهد شد. برای همین منظور کد خود را به شکل زیر بازنویسی می کنیم:

```
public class DoWhileLoop {
    public static void main(String[] args) {
        int number = 12;
        do {
            System.out.println("My Number Is:" + number);
            number++;
        } while (number <= 10);
    }
}
```

در واقع تنها تغییری که در کد انجام داده ایم این است که مقدار اولیه متغیر number را دوازده تعیین کرده ایم. حال برنامه را اجرا می کنیم:



اگر ما در برنامه خود از Loop یی از جنس while استفاده کرده بودیم، برنامه به ما هیچ خروجی نمی داد اما از آنجا که ما از یک do-while استفاده کرده ایم، علیرغم اینکه پاسخ به شرط `while(number <= 10);` معادل با false است اما Loop یک بار اجرا می شود (به این دلیل پاسخ false است که مقدار اولیه متغیر number معادل با عدد دوازده است). اما از آنجا که دستور `System.out.println();` در Block مرتبط با do نوشته شده است، برنامه ما حداقل یک بار این دستور را اجرا خواهد کرد چرا که محل قرار گرفتن این دستور در do است و پیش از آنکه برنامه به شرط قرار گرفته داخل پرانتز برسد، این دستور یک بار اجرا شده است.

خارج شدن از یک Loop

به طور معمول زمانی که جواب به شرط قرار گرفته در Loop معادل با false شود، برنامه به طور خود کار از Loop خارج خواهد شد. اما زمان هایی برای ما پیش می آید که خودمان عمداً می خواهیم برنامه از Loop خارج شود که برای این منظور از دستور `break` استفاده می کنیم. برای روشن شدن این مطلب می توانیم در همان پروژه قبلی که ایجاد کرده بودیم یک کلاس جدید تحت عنوان `ExitALoop` به معنی "خروج از یک Loop" ایجاد کنیم و یک Loop از جنس while در آن بسازیم. کد ما به شکل زیر خواهد بود:

دوره آموزش جاوا

کلیه حقوق متعلق به وب سایت نردبان است.

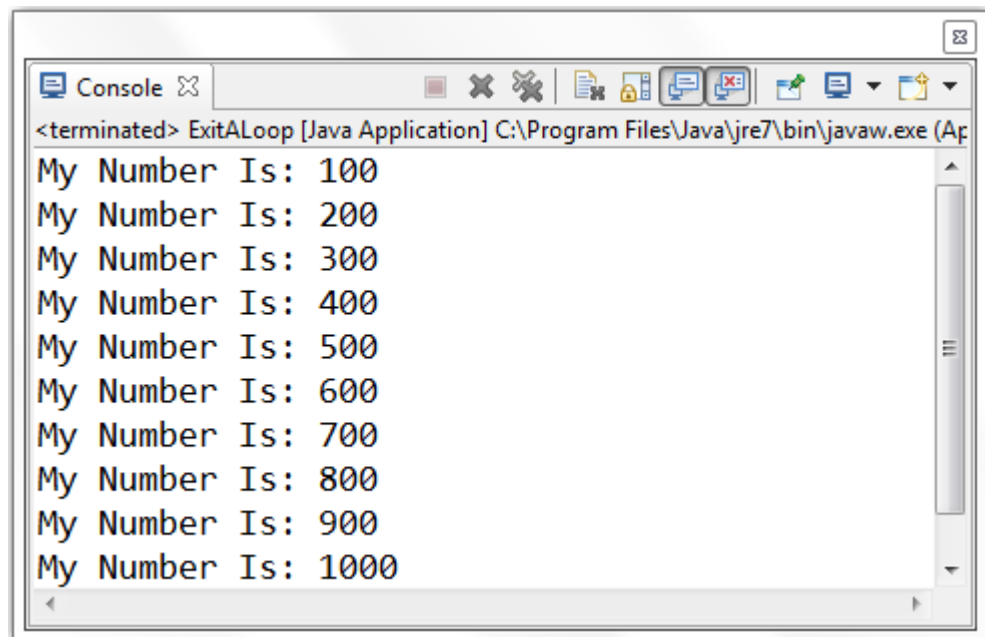
مدرس: بهزاد مرادی

```

public class ExitALoop {
    public static void main(String[] args) {
        int number = 0;
        while (number < 1000) {
            number += 100;
            System.out.println("My Number Is: " + number);
        }
    }
}

```

حال پس از اجرای برنامه، خروجی برنامه به شکل زیر خواهد بود:



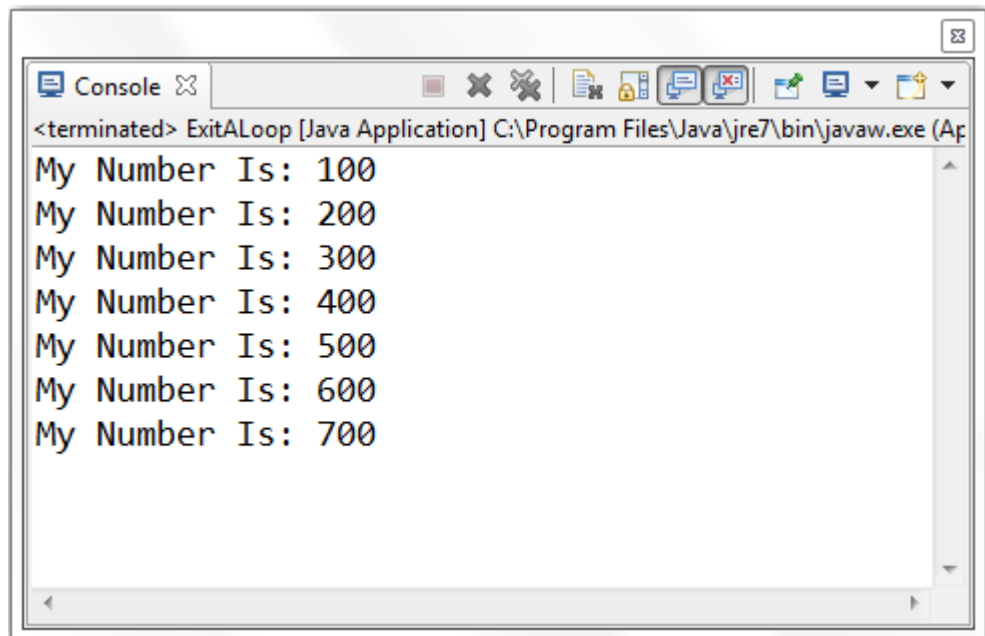
در کد فوق ما دستور داده ایم که مقدار اولیه متغیر برابر با صفر باشد و مادامیکه متغیر کوچکتر از هزار است Loop تکرار شده و هر دفعه به میزان صد واحد به آن اضافه شود. حال فرض کنیم که می خواهیم Loop ما پس از آنکه مقدار متغیر به عدد هفتصد رسید متوقف شده و برنامه از Loop خارج شود. برای این منظور کد خود را به شکل زیر تکمیل می کنیم:

```

public class ExitALoop {
    public static void main(String[] args) {
        int number = 0;
        while (number < 1000) {
            number += 100;
            System.out.println("My Number Is: " + number);
            if (number == 700) {
                break;
            }
        }
    }
}

```

کاری که در کد فوق انجام داده ایم این است که یک شرط داخل Block مرتبط با while از جنس if گذاشته ایم مبنی بر اینکه اگر مقدار متغیر number معادل با عدد هفتصد بود برنامه می بایست با استفاده از دستور break از داخل Loop خارج شده و بالتبع Loop ادامه نخواهد یافت (لازم به ذکر است که در زبان برنامه نویسی جاوا علامت = برای اختصاص دانه مقداری به یک متغیر و یا کلاس است و اگر بخواهیم مساوی بودن چیزی را بسنجیم بایستی از دو علامت مساوی پشت سر هم استفاده کنیم). حال خروجی برنامه به شکل زیر خواهد بود:



```

<terminated> ExitALoop [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Ap
My Number Is: 100
My Number Is: 200
My Number Is: 300
My Number Is: 400
My Number Is: 500
My Number Is: 600
My Number Is: 700

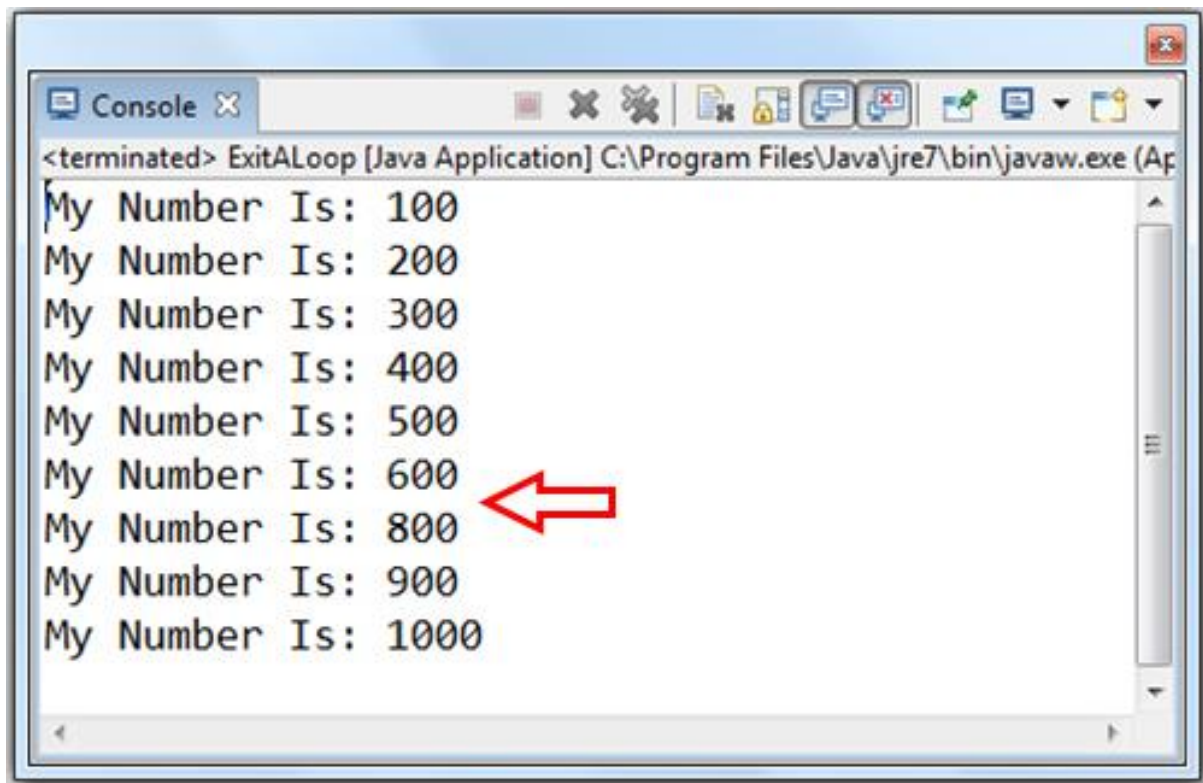
```

ادامه دادن یک Loop یا استفاده از دستور continue;

زمان هایی در برنامه نویسی برای ما پیش خواهد آمد که نیاز داریم اگر مقدار متغیر به مقدار خاصی رسید، به برنامه دستور دهیم که کار خود را ادامه دهد که برای این منظور از دستور continue استفاده می کنیم. برای روشن شدن مطلب کد خود را به شکل زیر بازنویسی می کنیم:

```
public class ExitALoop {
    public static void main(String[] args) {
        int number = 0;
        while (number < 1000) {
            number += 100;
            if (number == 700) {
                continue;
            }
            System.out.println("My Number Is: " + number);
        }
    }
}
```

در کد فوق دستور داده ایم مادامیکه مقدار متغیر number کوچکتر از هزار است، تعداد صد واحد به مقدار متغیر اضافه شود و در نهایت در بخش Console به نمایش در آید. اضافه بر این شرطی گذاشته ایم مبنی بر اینکه اگر مقدار متغیر برابر با عدد هفتصد شد Loop ما ادامه یابد. به عبارت دیگر زمانی که Loop به عدد هفتصد رسید به جای آنکه برنامه ادامه یابد و عبارت My Number Is: 700 در پنجره Console به نمایش در آید، برنامه به ابتدای Loop رفته و کار خود را "ادامه" می دهد. در واقع خروجی برنامه به شکل زیر خواهد بود:



```
<terminated> ExitALoop [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Ap
My Number Is: 100
My Number Is: 200
My Number Is: 300
My Number Is: 400
My Number Is: 500
My Number Is: 600
My Number Is: 800
My Number Is: 900
My Number Is: 1000
```

همانطور که مشاهده می شود عدد هفتصد در خروجی برنامه وجود ندارد چرا که خود به برنامه دستور داده ایم به محض اینکه مقدار متغیر معادل با عدد هفتصد شد به جای نمایش آن به ابتدای Loop رفته و کار خود را ادامه دهد.

در آموزش آینده با نحوه استفاده از دو متغیر داخل یک دستور Loop از جنس for تحت پروژه ای عملی به اسم Helping My Sibling به معنی "کمک به خواهر/برادرم" آشنا خواهیم شد.