به نام خدا

حلقه ها ی تکرار

در این مقاله خواهیم آموخت که چگونه از ساختارهایی که برای تکرار بخش های خاصی از برنامه وجود دارند، استفاده کنیم.

چرا ساختارهای تکرار؟

ابتدایی ترین کدهایی که در اولین مقاله ها نوشتیم به گونه ای بودند که همه شان به ترتیب از بالا به پایین و هر کدام فقط یک مرتبه اجرا می شدند. بعد از آن آموختیم که چطور با استفاده از ساختارهای تصمیم گیری، به برخی از کدهای برنامه اجازه ی اجرا شدن بدهیم و به بعضی دیگر این اجازه را ندهیم.

در این قسمت شما را با شرایطی آشنا می کنیم که قرار است کاری تا به صورت مکرر انجام بدهیم ؛ به طوریکه روح و قالب کار ما یکسان است. حالا در اینجا با بیان مثالی ساده سعی می کنیم تا ذهن شما را با علت استفاده از حلقه ها ، آشنا کنیم. فرض کنید می خواهیم برنامه ای داشته باشیم که تعداد خاصی عدد را از ورودی دریافت کند و حاصل جمع آن ها را به ما بدهد. در اینجا یک کار تکراری وجود دارد و آن گرفتن عدد و اعمال کردن آن در محاسبه ی مجموع کل می باشد. آیا این درست است که بخواهیم چندین بار کدی را بنویسیم که این کار را برای ما انجام بدهد؟ اگر بخواهیم این برنامه را طوری بنویسیم که برای ۱۰۰ عدد این کار را انجام بدهد چطور؟ قطعا این کار علاوه بر این که برای کسی که در حال نوشتن کد است، دشوار است، در پایان ما را با یک کد بسیار بزرگ مواجه می کند.

تا پایان این مقاله طرز کار با ساختار تکرارهای زیر را یاد می گیریم:

۱. while
۲. do – while
۳. for

ساختار تکرار while:

نمای کلی این ساختار به شکل روبه رو است:

```
while(boolean_expression){
    statements;
    ....
    ....
} // end of while loop
```

وقتی برنامه با کلمه ی کلیدی while مواجه می شود، ابتدا boolean_expression را ارزیابی می کند؛ در صورتی که این عبارت true ارزیابی شود، تمام کدهایی که در بلوک بعد از آن قرار دارند اجرا می شوند و بعد از اجرای کدهای موجود در این بلوک، مجددا عبارت boolean_expression ارزیابی می شود و باز هم در صورت true ارزیابی شدن این عبارت، بلوک بعد از آن اجرا می شود و این فرآیند تا زمانی ادامه پیدا می کند که عبارت boolean_expression به صورت false ارزیابی شود. چرا که در این مواقع کنترل برنامه به انتهای بلوک بعد از while منتقل می شود.

مثال) فرض کنید همانطوری که گفتیم می خواهیم برنامه ای را داشته باشیم که تعدادی عدد را از ما بگیرد و مجموع آن ها را محاسبه و چاپ کند. با فرض این که بخواهیم این کار را برای ۸ عدد انجام بدهیم؛ این کار را انجام می دهیم.

ابتدا مطابق آنچه که در مقاله های قبلی گفتیم، project جدیدی ایجاد کرده و کدی که در زیر مشاهده می کنید را ایجاد می کنیم: (می توانید این کد را عینا در netbeans IDE کپی کنید)

```
package whileloop;
import javax.swing.JOptionPane;
public class Main {
    public static void main(String[] args) {
        int counter=1;
        int sum=0;
        while(counter<=8){
            int num;
            num=Integer.parseInt(JOptionPane.showInputDialog("please insert
            nomber "+counter));
            sum+=num;
            counter++;
        } // end of while loop
        JOptionPane.showMessageDialog(null,"sum of your numbers is="+sum);
    } // end of main method
} // end of main class
```

در این مثال همانطوری که می بینید بلوک درون while با رنگ سبز مشخص شده است. اما نحوه ی کارکرد این برنامه به چه شکل است؟ خب ابتدا متغیر صحیحی به نام counter ایجاد می کنیم (تا تعداد دفعات دریافت اعداد را بشمارد) و آن را با ۱ مقدار دهی اولیه می کنیم. سپس متغیر صحیح دیگری به نام sum ایجاد می کنیم و آن را با صفر مقدار دهی اولیه می کنیم تا در هر بار اجرای حلقه ، حاصل جمع اعداد را درون آن ذخیره کنیم.

در برنامه ی فوق *while* چگونه کار می کند؟

به محض این که کنترل برنامه به while می رسد، ابتدا عبارت counter<=8 ارزیابی می شود و تا وقتیکه این عبارت true ارزیابی شود، بلوکی که به رنگ سبز نشان داده شده است، اجرا می شود و این کار تا زمانی ادامه پیدا می کند که عبارت فوق false ارزیابی شود.

در درون حلقه ی *while* فوق چه اتفاقی می افتد؟

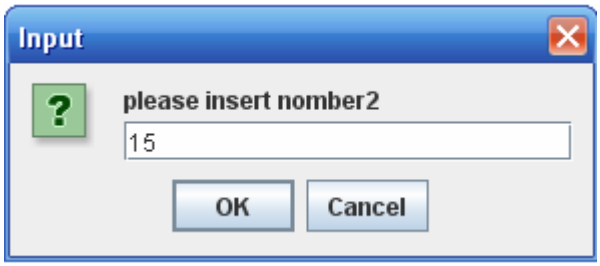
در اولین خط از این بلوک، یک متغیر صحیح به نام num تعریف کرده و در خط بعدی پس از آن که پیغام مناسبی به کاربر اعلام می کنیم، عددی از صفحه کلید خوانده وآن را به این متغیر نسبت می دهیم. در خط بعدی با نوشتن sum+=num ، مقداری را که در هر بار به متغیر num نسبت داده ایم را به مجموع اعداد قبلی اضافه می کنیم و در آخرین خط از این بلوک مقدار counter را یک واحد افزایش می دهیم.

مراحل اجرای این برنامه را با ورودی های فرضی ای که وارد کرده ام:

اولین باری که این حلقه اجرا می شود:

که در این بخش اولین عدد ۸ وارد شده است.

دومین مرتبه ای که این حلقه اجرا می شود:



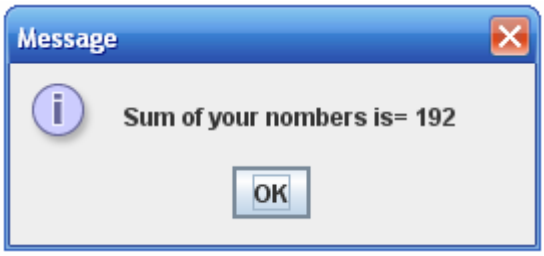
که در این بخش عدد ۱۵ به عنوان دومین عدد وارد شده است.

مرحله ی اجرای حلقه	مقدارنهایی متغیرcounter	مقدار عدد ورودی	مقدار نهایی متغیرsum
قبل از اجرا	۱	–	۰
۱	۲	۸	۸
۲	۳	۱۵	۲۳
۳	٤	۱۳	۳٦
٤	٥	۱۷	٥۳
٥	٦	۲٥	۷۸

و به همین ترتیب مراحل ادامه پیدا می کند تا با فرض وارد کردن داده ها طبق این جدول:

٦	٧	٦٩	١٤٧
٧	٨	٠	١٤٧
٨	٩	٤٥	١٩٢
بعد از اتمام حلقه	٩	–	١٩٢

پنجره ی محاوره ای زیر را به عنوان اعلام نتیجه ی نهایی این برنامه خواهیم داشت:



ساختار تکرار do-while:

این حلقه شباهت بسیار زیادی به حلقه ی while دارد. در حقیقت نمای کلی این ساختار به شکل زیر است:

```
do{
   statements;
   ....
   ....
   ....
}while(boolean_expression);
```

طریقه ی عملکرد این حلقه به این شکل است که ابتدا تمام بلوک محصور بین کلمه های کلیدی do و while اجرا می شوند و بعد از آن عبارت boolean_expression ارزیابی می شود؛ در صورتیکه این عبارت true ارزیابی شود کنترل برنامه یک بار دیگر به ابتدای بلوک رفته و کدهای محصور در این بلوک را اجرا می کند و این فرآیند تا زمانی اتفاق می افتد که به هنگام اتمام اجرای کدهای بلوک، عبارت boolean_expression به صورت false ارزیابی شود. چرا که در این زمان کنترل برنامه به بعد از این ساختار منتقل می شود.

مثال) برنامه ای را که در مثال قبل دنبال کردیم، این بار با ساختار do-while می نویسیم:

```
package whileloop;
import javax.swing.JOptionPane;
public class Main {
    public static void main(String[] args) {
        int counter=1;
        int sum=0;
        do{
            int num;
            num=Integer.parseInt(JOptionPane.showInputDialog("please insert
            nomber "+counter));
            sum+=num;
            counter++;
        } while(counter<=8); // end of do-while loop
        JOptionPane.showMessageDialog(null,"sum of your numbers is="+sum);
    } // end of main method
} // end of main class
```

جالب است بدانید که هر دوی این برنامه ها عملیات مد نظر ما را برای ۸ عدد انجام می دهند.

اما تفاوت حلقه های تکرار ***while*** و ***do-while*** در چیست؟

در حلقه ی while ابتدا عبارت boolean ارزیابی می شود و بعد از آن محتویات حلقه اجرا می شود. یعنی ممکن است که در اولین بار عبارت boolean به صورت false ارزیابی شده و محتویات بلوک حلقه اصلا اجرا نشوند. اما در حلقه های do-while ابتدا یک بار محتویات بلوک حلقه اجرا می شوند و بعد از آن عبارت boolean ارزیابی می شود. بنابر این می توان نتیجه گرفت که در حلقه ی do-while محتویات بلوک حداقل یک بار اجرا خواهند شد.

ساختار تکرار for:

این ساختار تکرار برای مواقعی که تعداد دفعات تکرار حلقه ی برنامه مان مشخص است، بسیار مفید است. در حقیقت نمای کلی این ساختار به شکل رو به رو است:

```
for(initialization expression;loop condition;step expression){
    statements;
    ....
    ....
}
```

در ساختار فوق ، در قسمت initialization expression ، متغیر حلقه را مقدار دهی اولیه می کنیم. در بخش loop condition شرایط ادامه ی حلقه را قرار می دهیم و در بخش step expression عملیات به روز رسانی متغیر حلقه را انجام می دهیم.

نحوه ی عملکرد حلقه ی ***for***:

وقتی کنترل برنامه به کلمه ی کلیدی for می رسد، ابتدا در بخش initialization متغیر حلقه را مقدار دهی اولیه می کند (مقدار دهی اولیه در حلقه ی for فقط قبل از اولین اجرای حلقه انجام می شود؛ یعنی این عمل فقط یک بار انجام می شود). سپس در قسمت loop condition عبارت بولین موجود را ارزیابی می کند و در صورت true ارزیابی شدن این عبارت ، محتویات بلوک این حلقه را اجرا کرده و مجددا به ابتدای حلقه می رود و متغیر حلقه را در بخش step expression به روز رسانی می کند . بعد از اولین اجرا و در سایر دفعات، فقط بخش loop condition مورد ارزیابی قرار می گیرد و تا موقعیکه این عبارت true ارزیابی شود، محتویات بلوک حلقه اجرا شده و متغیر حلقه به روز رسانی می شود؛ اما در صورتیکه این عبارت به صورت false ارزیابی شود، کنترل برنامه به بعد از ساختار for منتقل می شود.

مثال) فرض کنید همان برنامه ای که با ساختارهای قبلی مورد بررسی قرار دادیم ، این بار می خواهیم با ساختار for بنویسیم:

```
package whileloop;
import javax.swing.JOptionPane;
public class Main {
    public static void main(String[] args) {
        int counter=1;
        int sum=0;
        for(counter=1;counter<=8;counter++){
            int num;
            num=Integer.parseInt(JOptionPane.showInputDialog("please insert
            nomber "+counter));
            sum+=num;
        } // end of for loop
        JOptionPane.showMessageDialog(null,"sum of your numbers is="+sum);
    } // end of main method
```

```
} // end of main class
```

خروجی این برنامه نیز مانند سایر برنامه هایی است که پیش از این نوشتیم. ضمن اینکه هر سه برنامه ۸ عدد را از ورودی گرفته و حاصل جمع آن ها را به ما نشان می دهند.

در برنامه ی فوق وقتی کنترل برنامه به حلقه ی for می رسد، مقدار counter را برابر ۱ قرار می دهد. سپس شرط counter<=8 را ارزیابی می کند و چون این شرط درست ارزیابی می شود، محتویات بلوک (که به رنگ سبز مشخص شده اند) را اجرا می کند ؛ سپس به سراغ عبارت counter++ رفته و از این طریق با افزودن یک واحد به مقدار شمارنده ی حلقه ، آن را به روز رسانی می کند. این کار تا زمانی اتفاق می افتد که عبارت counter<=8 به صورت false ارزیابی شود.

نکته هایی پیرامون کار با حلقه ها:

- ✓ در تمام ساختارهای تکرار اگر فقط یک دستور دارید که می خواهید آن را درون حلقه قرار دهید، می توانید آن را بیرون از یک جفت آکولاد قرار بدهید اما این را به خاطر داشته باشید که استفاده از آکولاد ها ، خوانایی برنامه ی شما را افزایش می دهد.
- ✓ به یاد داشته باشید که همواره باید اجرای حلقه ی شما در شرایط خاصی متوقف شود. وگرنه برنامه ی شما بی نهایت بار اجرا می شود. مانند:

```
while(True){
    System.out.println("hello");
}
```

که بی نهایت مرتبه پیام hello را در خروجی چاپ خواهد کرد!!!

- یکی از متداول ترین اشتباهات این است که به هنگام استفاده از حلقه ی do-while گذاشتن سمیکالون ; بعد از عبارت مربوط به while از قلم می افتد:

```
do{
    ....
    ....
    ....
}while(boolean_expression);
```

- به یاد داشته باشید که تحت هر شرایطی باید از چه نوع ساختار تکرار استفاده کنید. مثلا همواره به یاد داشته باشید که محتویات بلوک حلقه ی do-while حداقل یک بار اجرا خواهند شد. مثلا دستور X++ در حلقه ی زیر هرگز اجرا نمی شود:

```
While(false){
   X++;
}
```

در حالیکه در حلقه ی زیر یک بار اجرا می شود:

```
do{
   x++;
}while(false);
```

***

تهیه شده در: www.java4every1.wordpress.com

ایمیل تماس با مدیریت وبلاگ: blogsofmine@gmail.com

مدیریت وبلاگ مثل همیشه منتظر انتقادات و پیشنهادات شما عزیزان جهت هرچه بهتر برگزار شدن فرآیند آموزش در وبلاگ می باشد.