بسم الله الرّحمن الرّحیم

دستورات شاخه ای

break , continue , return

در مقاله های قبلی راجع به یکسری از ساختارهای کنترلی توضیح دادیم. حالا در این مقاله سعی می کنیم تا شما را با سه دستوری که در عنوان این مقاله مشخص کرده ایم، آشنا کنیم.

ساختارهای کنترلی ای که تاحالا معرفی کرده بودیم، چگونگی اجرای کدهای برنامه هایمان را کنترل می کردند. اما در بعضی از مواقع لازم است تا بتوانیم یکسری کنترل های خاص دیگری را بر روی برنامه هایمان اعمال کنیم؛ به گونه ای که مثلا شاید لازم باشد کنترل برنامه از حلقه خارج شود ، از یک متد (همان مفهوم تابع در زبان های C++ و C#) خارج شود و یا اجرای یک حلقه را از سر بگیرد.

برای هر کدام از موارد مذکور دستورات خاصی وجود دارد که در این مقاله راجع به آن ها توضیح داده و مثال می آوریم.

دستور break:

این دستور به منظور خارج شدن از ساختار switch (که در مقاله های قبلی راجع به آن توضیح داده شد و همانطوری که به یاد دارید از این کلمه استفاده کرده بودیم) و همچنین پایان دادن به اجرای حلقه ها به کار می رود. و خود این دستور به دو دسته ی برچسب دار و بدون برچسب تقسیم می شود.

دستور break بدون برچسب :

این دستور درون حلقه ها و به تنهایی به کار می رود؛ به گونه ای که وقتی کنترل برنامه به این دستور می رسد، از ساختار درونی ترین حلقه خارج می شود. دقیقا مثل این است که در همان موقع اجرای حلقه پایان یافته باشد (منتها بدون اعمال هیچ چیز اضافه تری). برای درک بهتر فرض کنیم در همان مسئله ای که در مقاله ی حلقه ها بیان کردیم (برنامه ای که ۸ عدد را از ما بگیرد و حاصل جمع آن را به ما اعلام کند) می خواهیم در صورتی که عددی منفی وارد برنامه مان شد، برنامه دیگر هیچ عددی را دریافت نکند و در همان جا حاصل نهایی را اعلام کند.

به کدهای این برنامه دقت کنید:

```
package breakstatement;
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));
            if(num<0) break; // in this situation, program return final result
            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 حالت فوق، همان طور که گفته شد وقتی کنترل برنامه به کلمه ی کلیدی break (که با رنگ آبی مشخص شده است) می رسد؛ از حلقه ی while فوق (که داخلی

ترین حلقه نسبت به break است) خارج شده و برنامه اعداد دیگر را از کاربر در یافت نمی کند. به عنوان مثال اگر سومین عدد وارد شده منفی باشد، برنامه فقط حاصل جمع دو عدد اول وارد شده را برمی گرداند.

دستور break برچسب دار:

همانطوری که در مورد دستور break برچسب دار توضیح دادیم، این دستور باعث می شود که کنترل برنامه از درونی ترین حلقه ای که break در آن قرار دارد، خارج شود. حالا در مواقعی هست که شما می خواهید از حلقه ای خارج شوید که بیرونی تر از حلقه ای است که عبارت break شما در آن جا قرار دارد. در این مواقع برای قسمتی که می خواهید کنترل برنامه به آن جا ارجاع داده شود، یک برچسب (label) تعریف کرده، سپس عبارت break را به همراه نام همان برچسب مورد استفاده قرار می دهید:

```
break label_name;
```

شاید یک مقدار سخت باشد اما این مثال به راحتی می تواند درک این مسئله را برای شما راحت کند.

مثال) فرض کنید می خواهید برنامه ای داشته باشید که از شما جدول ضرب می پرسد. شاید یک مقدار هیجان انگیز باشد که این برنامه را برای یک دانش آموزی که تازه جدول ضرب یاد گرفته است اجرا کنید. می خواهیم برنامه را طوری طراحی کنیم که تمام جدول ضرب اعداد یک رقمی را از کاربر سوال کند و به محض اینکه کاربر

جوابی را اشتباه وارد کرد، کنترل برنامه با اعلام پیغامی به کاربر اعلام کند که او این مسابقه را باخته است و برنامه نیز خاتمه پیدا کند ولی در صورتی که کاربر به تمام سوالات پاسخ صحیح بدهد، به او پیام تبریک بگوید.

قطعه کد زیر این برنامه را برای شما پیاده سازی می کند. (می توانید به همان شکلی که در مقاله های قبلی عنوان کردیم، بعد از ایجاد یک project جدید این قطعه کد را در محیط netbeans IDE کپی کرده و آن را اجرا کنید. چون این همان کدی است که خود من از محیط netbeans کپی کرده ام و در اینجا قرار داده ام. اگر نمی دانید که چطور می توانید project جدیدی درست کنید و کد برنامه تان را در آن قرار دهید، به وبلاگ www.java4every1.wordpress.com مراجعه کنید و مقاله ی "اولین برنامه تان را در جاوا بنویسید" را مطالعه کنید.)

```
package labledbreakstatement;
import javax.swing.JOptionPane;
/**
 *
 * @author Javad
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int answer;
        String finalMessage="Congratulation! you won";
        wrongAnswerOccur:
            for(int i=1;i<10;i++)
                for(int j=1;j<10;j++){
                    answer=Integer.parseInt(JOptionPane.showInputDialog(i+"*"+j+" ?"));
                    if (answer!=i*j){
                        finalMessage="Sorry you lost";
                        break wrongAnswerOccur;
                    }
                }
```

```
        JOptionPane.showMessageDialog(null, finalMessage);
    }

}
```

و اما دستور break wrongAnswerOccur در برنامه ی فوق ، چگونه کار می کند؟ همانطوری که می دانید اگر در برنامه ی فوق به جای استفاده از break برچسب دار، آن را بدون برچسب استفاده کنیم، کنترل برنامه فقط از درونی ترین حلقه ی for (که در کد بالا به رنگ قرمز مشخص شده است) خارج می شود. اما با قرار دادن برچسبی به نام wrongAnswerOccur برای حلقه ی for بیرونی تر (که به رنگ سبز مشخص شده است) و به کار بردن عبارت break wrongAnswerOccur باعث می شویم که به محض وقوع اشتباه در جواب کاربر، کنترل برنامه از حلقه ی for بیرونی تر (که خود شامل یک حلقه ی for درونی تر است) خارج می شود.

به شما پیشنهاد می کنم که حتما کد هایی که به عنوان مثال در اختیارتان قرار می گیرد، اجرا کنید و کدها را دستکاری کنید تا بتوانید مطالب بیشتری یاد بگیرید.)

دستور continue:

در مواقعی لازم است که کنترل برنامه ی خود را بنا به دلیلی از یکی از حلقه های برنامه خارج کرده و از پردازش باقیمانده ی کدها جلوگیری کنیم. منتها با این تفاوت نسبت به دستور break که این دستور کنترل برنامه را به گام دیگر حلقه منتقل کنیم و این در حالی است که دستور break به طور کامل اجرای حلقه را متوقف می کند. در این مواقع دستور continue به ما کمک می کند تا این کار را انجام بدهیم.

این دستور نیز مانند دستور break خود به دو دسته ی برچسب دار و بدون برچسب تقسیم می شود.

دستور continue بدون برچسب:

با به کار بردن این دستور، باعث می شویم که کنترل برنامه به انتهای درونی ترین حلقه ای که عبارت continue در آن قرار دارد، منتقل شود و مجددا عبارت بولین آن حلقه مورد ارزیابی قرار می گیرد. در حقیقت با استفاده از این دستور باعث می شویم که از عبارت continue به بعد ارزیابی نشده و گام دیگر حلقه شروع شود.

به عنوان مثالی برای این بخش فرض کنید می خواهیم برنامه ای داشته باشیم که تعدادی عدد (مثلا ۵) عدد را از کاربر دریافت کرده و تعداد اعداد منفی وارد شده را شمرده و در نهایت به کاربر اعلام کند.

برنامه ی زیر این کار را برای ما انجام می دهد:

```
package unlabeledcontinuestatement;
import javax.swing.JOptionPane;
public class Main {
   public static void main(String[] args) {
      int counterOfNegativeNumbers=0;
      for(int counter=0;counter<5;counter++){
         int newnum=Integer.parseInt(JOptionPane.showInputDialog("enter your next num:"));
         if(newnum>=0) continue;
         counterOfNegativeNumbers++;
      }
      JOptionPane.showMessageDialog(null,counterOfNegativeNumbers);
   }

}
```

باز هم به شما پیشنهاد می کنم که این برنامه ها را به عنوان کد برنامه هایتان اجرا کنید و سعی کنید تا آنجا که می توانید این کدها را دستکاری کنید تا به نتایج بیشتری برسید.

اما continue در برنامه ی فوق چطور کار می کند؟ در برنامه ی مذکور، به محض اینکه برنامه با دستور continue مواجه می شود، از انجام باقیمانده ی دستورات حلقه ی درونی صرف نظر کرده و در حالی به انتهای درونی ترین حلقه ی نسبت به خودش می رود که عبارت شرط آن حلقه counter<5 را بررسی می کند. برای بیان بهتر طریقه ی عملکرد برنامه ی فوق به رنگ های مختلف کدها دقت کنید (آبی: قسمت های مربوط به عبارت continue و حلقه ای که با این عبارت در ارتباط است. قرمز: کدهایی که در صورت اجرا شدن دستور continue اجرا نخواهند شد. نارنجی: شرط حلقه است که به محض رسیدن برنامه به دستور continue ، مورد ارزیابی قرار خواهد گرفت.)

دستور continue برچسب دار:

این دستور دقیقا مانند continue بدون برچسب عمل می کند؛ منتها با این تفاوت که نسبت به مکان تعریف شدن برچسب خود، به اجرای یک حلقه ی خارجی تر خاتمه می دهد.

به عنوان مثال فرض کنید می خواهیم برنامه ای را که برای پرسیدن جدول ضرب نوشته بودیم؛ به گونه ای تغییر بدهیم که جدول ضرب اعداد یک رقمی را از کاربر سوال کند و به محض اینکه کاربر جواب اشتباهی را وارد کند، جواب درست را به

کاربر اعلام کرده و از پرسیدن باقیمانده ی جدول ضرب در مورد عددی که پاسخ اشتباه در آن رخ داده است، امتناع کند. سپس به سراغ پرسیدن جدول ضرب از عدد بعدی شود. مثلا اگر کاربر در سری سوالات ضرب عدد ۲، جواب ۲*۷ را اشتباه وارد کند، کنترل برنامه ادامه ی سوالات را از ۳*۱ شروع کند. و در نهایت به کاربر اعلام کند که در جواب دادن به ضرب چند عدد اشتباه کرده است.

کدهای زیر این برنامه را برای شما درست خواهند کرد:

```
package labledbreakstatement;
import javax.swing.JOptionPane;
public class Main {
    public static void main(String[] args) {
        // TODO code application logic here
        int answer;
        int mistakesCounter=0;
        String finalMessage="You could finish this exercise";
        wrongAnswerOccur:
            for(int i=1;i<10;i++)
                for(int j=1;j<10;j++){
                    answer=Integer.parseInt(JOptionPane.showInputDialog(i+"*"+j+" ?"));
                    if (answer!=i*j){
                        JOptionPane.showMessageDialog(null,"Your answer was wrong! press OK to continue");
                        mistakesCounter++;
                        continue wrongAnswerOccur;
                    }
                }
        JOptionPane.showMessageDialog(null, finalMessage+" You had "+mistakesCounter+" mistake(s)");
    }
}
```

و اما دستور continue wrongAnswerOccur در برنامه ی فوق چگونه عمل می کند؟ همانطوری که مشاهده می کنید برچسب wrongAnswerOccur برای حلقه ی for بیرونی قرار گرفته است و به محض اینکه کنترل برنامه به دستور continue wrongAnswerOccur برسد، به اجرای حلقه ی for بیرونی خاتمه داده و با به روز رسانی متغیر این حلقه (i++) ، شرط آن را ارزیابی می کند (i<10).

می توانید بعد از ایجاد یک project جدید، کد برنامه ی فوق را در محیط netbeans یا حتی notepad کپی کنید و همانطوری که در مقاله های قبلی به شما آموزش داده ایم، آن را اجرا کنید.

دستور return:

متدها در جاوا یکی از پرکاربرد ترین ابزار هستند. و دستور return یکی از پرکاربرد ترین دستورات در متدهاست. این دستور که خیلی ساده کار می کند ، باعث می شود که کنترل برنامه از متدی که در آن هستیم ، خارج شده و به بخشی برود که این متد در آن جا فراخوانی شده است. دستور return خود به دوشکل ساده و به همراه آرگومان به کار می رود.

شکل ساده ی این دستور برای مواقعی مورد استفاده قرار می گیرد که نوع داده ی برگشت داده شده توسط متد، void باشد. در این مواقع این دستور را به این شکل می آوریم:

```
return;
```

شکل به همراه آرگومان این متد، به این ترتیب مورد استفاده قرار می گیرد که کلمه ی کلیدی return را به همراه مقداری می آوریم که قرار است به عنوان نتیجه ی متد محسوب شود. بدیهی است که آرگومان return در این مواقع باید با نوع داده ی برگشت داده شده توسط متد، همخوانی داشته باشد. در این مواقع این دستور را به این شکل می آوریم:

```
return argument;
return number;
```

```
return "hello";
return 1542;
return 13.25;
return 'a';
```

إن شاء الله در مقاله های بعدی و در قسمت متدها ، بیشتر در مورد return توضیح خواهیم داد.

***

تهیه شده در: www.java4every1.wordpress.com

ایمیل تماس با مدیریت وبلاگ: blogsofmine@gmail.com

در پایان مثل همیشه پذیرای انتقادات و پیشنهادات شما عزیزان جهت هرچه بهتر برگزار شدن فرآیند آموزش در وبلاگ هستم.