آموزش حلقه for در جاوا

همانطور که قبلا ذکر کردیم، برنامه نویسی که اکنون در حال انجام آن هستید یک برنامه نویسی پی در پی می باشد. این به این معناست که جریان صعودی می باشد، از بالا به پایین، با هر خط از کد که اجرا شده است، مگر اینکه شما درخواست دیگری از جاوا بکنید.

در بخش قبل مشاهده کردید که یک روش برای اینکه به جاوا بگویید هر خط از کد را اجرا نکند، استفاده از عبارت IF برای بخش های خاموش کد می باشد.

راه دیگر برای به هم ریختن جریان از بالا به پایین، استفاده از loop ها می باشد. یک برنامه نویسی loop آن برنامه نویسی می باشد که مجددا به عقب بازمی گردد. اگر بازگشت مجدد به عقب اجبار باشد، شما می توانید خطوط را به طور مکرر اجرا کنید.

به عنوان یک مثال فرض کنید که می خواهید اعداد از یک تا 10 را با هم جمع کنید. شما این کار را در جاوا به راحتی می توانید انجام دهید، مانند زیر:

```
int addition = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;
```

اما اگر بخواهید اعداد را از یک تا 1000 با هم جمع کنید، واقعا نمی خواهید از این متود استفاده کنید. در عوض می توانید از loop استفاده کنید تا مکررا وارد خطوط کد شوید تا اینکه به 1000 برسید. سپس می توانید از loop خارج شده و به راه خود ادامه دهید.

Java For Loops

با For Loops یکی از متداول ترین انواع loop ها آغاز خواهیم کرد. به نظر می رسد قسمت For از For Loops معنای خود را از دست داده باشد. اما می توانید آن را به این شکل در نظر بگیرید loop :برای یک مجموعه عدد از دفعات .
ساختار For Loop مانند زیر می باشد:

```
for ( start_value; end_value; increment_number ) {

//YOUR_CODE_HERE

}
```

بنابراین بعد از لغت for با حروف کوچک، یک جفت آکولاد دارید. در داخل این آکولادها سه چیز لازم می باشد: مقدار آغازین برایloop ، مقدار پایانی برایloop ، و روشی برای به دست آوردن یک تعداد به تعداد دیگر. این فرایند افزایش عدد (increment number) نامیده می شود و معمولا 1 می باشد. اما لازم نیست اینطور باشد. اگر تمایل داشته باشید می توانید تا تعداد 10 نیز بالا بروید.

بعد از پرانتزها نیاز به یک جفت آکولاد دارید. آکولادها برای جداسازی آن بخشی از کد استفاده می شوند که می خواهید به طور مکرر اجرا شود. یک مثال می تواند این مورد را شفاف سازد.

برای این مورد یک پروژه ی جدید آغاز کنید. پروژه و گروه را می توانید به دلخواه خود نامگذاری کنید. (ما پروژه ی خود را "loops" و گروه را "ForLoops"نامیدیم.) اکنون کد زیر را اضافه کنید:

```
package loops;


public class ForLoops {

    public static void main(String[] args) {

        int loopVal;
        int end_value = 11;

        for (loopVal = 0; loopVal < end_value; loopVal++) {

            System.out.println("Loop Value = " + loopVal);
        }
    }

}
```

ما کار را با تنظیم یک مقدار صحیح آغاز کردیم که loopVal نامیده ایم. خط بعدی یک متغیر صحیح دیگر را تنظیم می کند. این متغیر برای مقدار نهایی loopاستفاده می شود و به 11 تنظیم شده است. آنچه ما تصمیم داریم انجام دهیم چاپ اعداد از 0 تا 10 می باشد.

در داخل پرانتزهای for loop خط زیر را مشاهده می کنید:

```
loopVal =0; loopVal < end_value; loopVal++
```

اولین قسمت به جاوا می گوید که در چه شماره ای looping را آغاز کنید. در اینجا ما یک مقدار صفر را به متغیر loopVal اختصاص می دهیم. این اولین عدد در loop می باشد. بخش بعد از برخی منطق های شرطی استفاده می کند:

```
loopVal < end_value
```

این عبارت می گوید که loopVal کمتر از end_value باشد for loop .به چرخش خود ادامه خواهد داد در حالیکه مقدار داخل متغیر loopVal کمتر از متغیری به نام end_value می باشد. تا زمانیکه کمتر بودن loopVal از end_value درست باشد، جاوا looping را روی کد در بین آکولادها حفظ خواهد کرد.

بخش نهایی بین پرانتزهای for loop مانند زیر می باشد:

loopVal++

آنچه در اینجا انجام می دهیم این است که به جاوا می گوییم چگونه از مقدار آغازین در loopVal به عدد بعدی در ترتیب حرکت کند. می خواهیم از 0 تا 10بشمریم. عدد بعد از 0 عدد 1 می باشد .loopVal++ راه کوتاه بیان عبارت "افزودن 1 به مقدار در متغیر" می باشد.

به جای بیان loopVal++ می توانیم عبارت زیر را نیز بنویسیم:

```
loopVal = loopVal + 1
```

در سمت راست علامت تساوی loopVal + 1 را داریم. بنابراین جاوا 1 را به هر چیزی که در متغیر loopVal ذخیره شده، اضافه می کند. زمانی که 1 را به مقدار اضافه کرده است، نتیجه را در داخل متغیر در سمت چپ تساوی ذخیره خواهد کرد. این مجددا متغیر loopVal می باشد. نتیجه این است که افزوده شدن 1 به loopValادامه می یابد. این فرایند افزایش متغیر نامیده می شود. نماد تندنویسی متغیر ++ بسیار متداول می باشد:

```
int some_number = 0;
some_number++;
```

وقتی که کد بالا اجرا شود، مقدار some_number مقدار 1 خواهد بود. این راه کوتاه بیان زیر می باشد:

```
int some_number = 0;
some_number = some_number + 1;
```

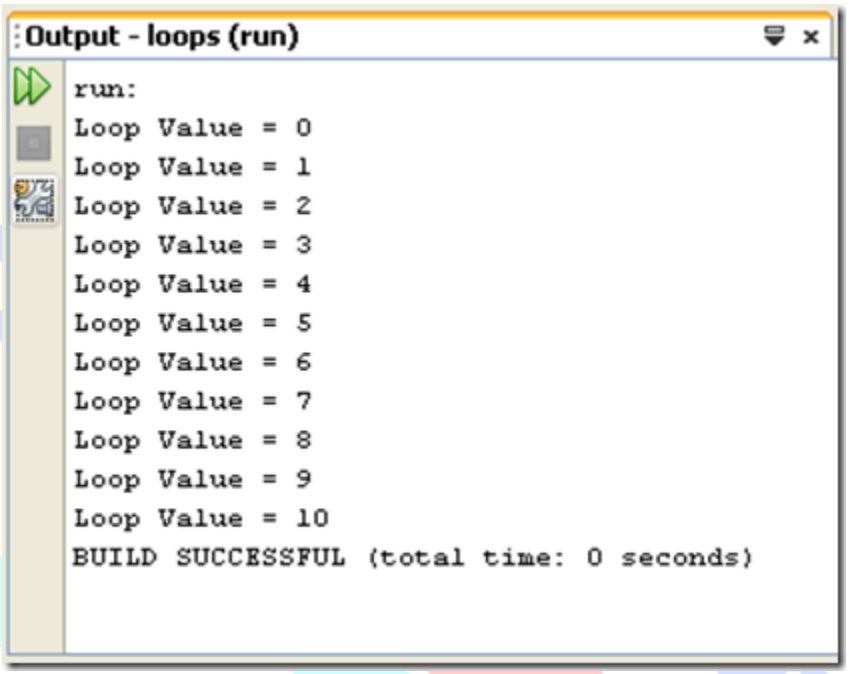
برای پوشاندن آن، for loopدر حال بیان مورد زیر می باشد:

Loop Start value: 0 Keep Looping While : مقدار آغازین کمتر از 11 می باشد How to advance to the .
end value : افزودن 1 را به مقدار نهایی حفظ می کند.

در داخل پرانتزهای for loop عبارت زیر دیده می شود:

```
System.out.println("Loop Value = " + loopVal);
```

هر چیزی که در داخل متغیر loopValمی باشد، همراه با متن چاپ خواهد شد:



بنابراین برنامه را در یک loop گیر انداخته ایم و آن را مجبور کرده ایم به طور مکرر اجرا شود. با هر اجرای loop مقدار 1 به loopValاضافه می شود .به چرخش خود ادامه خواهد داد، در حالیکه مقدار در داخل loopVal کمتر از مقدار در داخل end_value می باشد. هر چیزی که در داخل آکولادهای loop می باشد کدی است که پشت سر هم اجرا خواهد شد. و این تمام نکته ی loop می باشد: اجرای کد آکولاد به طور پیوسته.

ئر اینجا کدی را می بینید که اعداد را از 1 تا 10 به هم اضافه می کند. آن را امتحان کنید:

```
public static void main(String[] args) {

    int loopVal;
    int end_value = 11;
    int addition = 0;

    for (loopVal = 1; loopVal < end_value; loopVal++) {

        addition = addition + loopVal;
    }

    System.out.println("Total = " + addition);
}
```

پاسخی که در پنجره ی Output دریافت می کنید 55 می باشد. کد نیز کم و بیش همان کد مربوط به loop قبلی می باشد. ما همان دو متغیر loopVal و end_valueرا تنظیم شده در بالا داریم. همچنین یک متغیر صحیح سومی هم داریم که additionنامیده می شود. این مقدار مجموع 1 تا 10 را حفظ خواهد کرد.

بین پرانتزهای for loop تقریبا همان مقدار آخرین بار می باشد: ما در حال loopingهستیم در حالیکه loopVal کمتر از end_valueمی باشد؛ و در حال افزودن 1 به متغیر loopVal می باشیم. تنها تفاوت این است که اکنون مقدار آغازین 1 می باشد.

بین آکولادها فقط یک خط کد وجود دارد:

```
addition = addition + loopVal;
```

این خط مجزا اعداد 1 تا 10 را با هم جمع می کند. اگر در مورد چگونگی کار کردن آن گیج شده اید، از سمت راست علامت تساوی آغاز کنید:

```
addition + loopVal;
```

اولین اجرایloop ، متغیر addition مقدار 0 را در خود حفظ می کند. در این حین متغیر loopVal مقدار 1 را در خود دارد. جاوا 0 را به 1 اضافه خواهد کرد. سپس نتیجه را در متغیر سمت چپ علامت تساوی ذخیره خواهد کرد. مجددا این متغیر addition (مجموع) می باشد. هرآنچه از قبل در متغیر addition حفظ شد (0)، پاک خواهد شد و با یک مقدار جدید (1) جایگزین می شود.

در دومین اجرایloop ، مقادیر در دو متغیر عبارتند از:

```
addition (1) + loopVal (2);
```

واضح است که پاسخ 1+2 عدد 3 می شود. بنابراین 3 مقدار جدیدی است که در سمت چپ تساوی ذخیره خواهد شد.

در سومین اجرای loop مقادیر جدید عبارتند از:

```
addition (3) + loopVal (3);
```

جاوا 3 را با 3 جمع خواهد کرد و 6 را در سمت چپ تساوی ها ذخیره می کند .این چرخش ادامه می یابد تا اینکه loop به پایان برسد. نتیجه ی نهایی 55 می باشد.

)دقت داشته باشید که بعد از آخرین آکولاد ازloop ، خط چاپی ما خارج از for loopمی باشد(.

تمرین :کد خود را طوری تغییر دهید که جاوا اعداد را از 1 تا 100 اضافه کند. پاسخی که در یافت می کنید 5050 می باشد .تمرین :یک برنامه ی جدول زمانبندی بنویسید. برنامه باید از یوزر بخواهد تا یک عدد را وارد کند. سپس این عدد به عنوان دفعات مربوط به جدول استفاده خواهد شد. بنابراین اگر یوزر عدد 10 را وارد کند، جدول 10 مرتبه باید نمایش داده شود .کمک برای این تمرین for loop :شما فقط به دو خط کد بین آکولادها نیاز دارد و یکی از آنها خط چاپ (print line)می باشد. شما تنها به یک خط مجزا نیاز دارید تا پاسخ های جدول خود را محاسبه کنید .تقریبا باید چگونگی گرفتن عدد از یوزر را بدانید. این کار در آکولادهای loop استفاده می شود تا پاسخ به ضرب شما را انجام دهد .جول شما تنها نیاز به رفتن از 1 تا 10 را دارد .تمرین از for loopبرای چاپ اعداد فرد از 1 تا 10 استفاده کنید. (برای انجام آسان این تمرین در مورد افزایش مقدار loop فکر کنید که سومین آیتم بین پرانتزها می باشد (.یکی از راه های سخت انجام تمرین بالا استفاده از اپراتوری است که هنوز با آن آشنایی ندارید – اپراتور modulus. این اپراتور زمانی استفاده می شود که یک عدد را تقسم کرده و باقیمانده را نگاه می دارید. بنابراین 10 Mod 3عدد 1 می باشد، زیرا 10 تقسیم بر 3 می شود 3.

باقیمانده نیز 1 می باشد که آن را حفظ می کنید. اپراتور Modulus در جاوا نماد درصد می باشد که تا حدی گیج کننده است. مانند زیر:

```
int remainder;
int total = 10
remainder = total %3
```

بنابراین عددی را که می خواهید تقسیم کنید در ابتدا قرار می گیرد. سپس یک نماد درصد تایپ می کنید که پس از آن تقسیم کننده قرار می گیرد. پاسخ نیز باقیمانده می باشد.

در تمرین بالا می توانستید از 2 به عنوان عدد Mod استفاده کنید و سپس از عبارت IF در for loop برای امتحان باقیمانده استفاده کنید. (آیا متوجه می شوید که چرا 2 باید عدد Mod باشد؟)

در بخش بعد نگاهی به while loops خواهیم داشت.