CLARUSWAY

WAY TO REINVENT YOURSELF

Loops

Lesson:
JAVA Chapter 08

Loops (Döngüler)

Bir programımız için bazı olayların tekrarlanmasını istenebilir. Java'da döngüler kullanarak gerçekleştireceğimiz bu tekrarlanan olaylar ile işlemlerimizi daha kolay bir şekilde gerçekleştirebilir. Döngü oluştururken belirleyeceğimiz koşul doğru olduğu sürece döngü içerisinde bulunan komut satırları çalışmaya devam eder. Koşul **false** bir değer vermeye başladığında ise döngü kırılır ve sonraki komut satırları çalışmaya devam eder. Programlarımız içerisinde kullanabileceğimiz bazı döngü çeşitleri vardır.

Java'da Döngüler

1. Java For Loop
2. Java While Loop
3. Java Do-While Loop
4. Java for-each Loop
5. …

For Loop

Belirli bir şart altında tekrarlanan işlemler (aksiyonlar) için kullanılan kod bloklarına LOOP (döngü) denir. Tekrar sayısı sabit olan durumlarda for loop kullanılması önerilir.


```
for ( int i=4; i>1; i- - ) {
    System.out.println( i );
}
```

For Loop

FOR Döngüsünün Akış Şeması

For Loop

- TRICK :)

- **Ending Condition daima true olduğu durumda loop sonsuz döngüye girer.**
- **loop'ta Ending Condition hiç true olmazsa loop body asla execute(çalışmaz) olmaz.**
- **Artış veya azalma değeri 1 olmak zorunda değil, farklı da olabilir. (i+=2 i-=3 etc...)**
- **for döngüsünde her bölüme bir değer atama zorunluluğu yoktur. İlk değer atama, koşul ve artırma bölümlerinden birini veya tamamını hiç kullanmayabilirsiniz. Döngü içinde kullanılmayan bölüm boş kalır.**
  for (; ;) işlem-satırı
- **Döngüleri tek başına kullanabileceğiniz gibi, birbiri içinde de kullanabilir.**

Nested For Loop (İç içe Döngü)

Bir döngü yapısının içine başka bir döngü yapısının yerleştirilmesiyle elde edilen yapıya **iç içe döngü (nested loop)** adı verilir. Java dilinde, *if* deyimlerini herhangi bir derinliğe kadar iç içe kullanmak nasıl mümkünse, döngü deyimlerini de iç içe kullanmak olasıdır. Şu kural iç içe döngüler için daima geçerlidir:

İç içe döngülerde en içteki döngü en önce tamamlanır.

Nested For Loop (İç içe Döngü)

Aşağıdaki programda, dış döngü olan **a** parametreli döngü 5 kez çalışıyor **(a=1, 2, 3, 4, 5)**. **a**'nın her değeri içinse içteki döngü 3 kez çalışıyor **(i=1, 2, 3)**. Böylece aşağıdaki çıktı elde ediliyor:

```
public class IcIce
{
  public static void main(String args[])
   {
      int a,i; //5 kez tekrarla
      for(a=1;a<=5;a++)
       {
         System.out.println("a= "+a);
           for(i=1;i<=3;i++) //3 kez tekrarla
            {
              System.out.println("i= "+i);
            }
         System.out.println();
       }
   }
}
```

```
a= 1
i= 1
i= 2
i= 3

a= 2
i= 1
i= 2
i= 3

a= 3
i= 1
i= 2
i= 3

a= 4
i= 1
i= 2
i= 3

a= 5
i= 1
i= 2
i= 3
```

While Loop

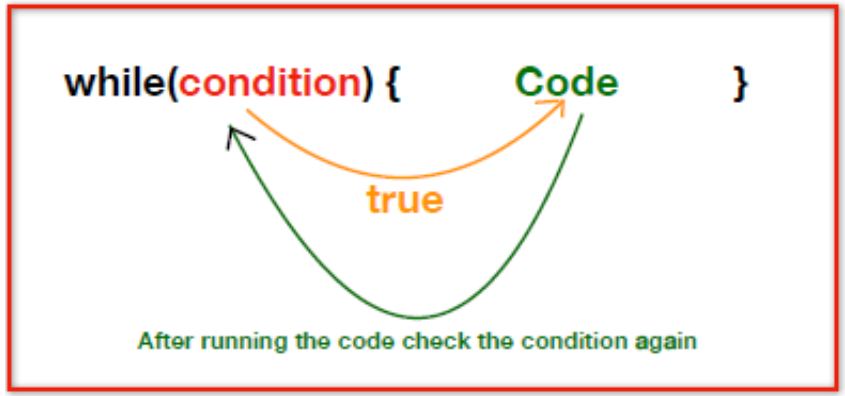
JAVA *while loop*, belirtilen Boolen şartı true olana kadar programın bir bölümünü tekrarlamak için kullanılır. Boolean şartı false olur olmaz döngü otomatik olarak kırılır ve sıradaki code satırından devam edilir.
While loop, For loop'tan farklı olarak tek bir parametre alır ve bu parametre doğru olduğu müddetçe loop sürer.

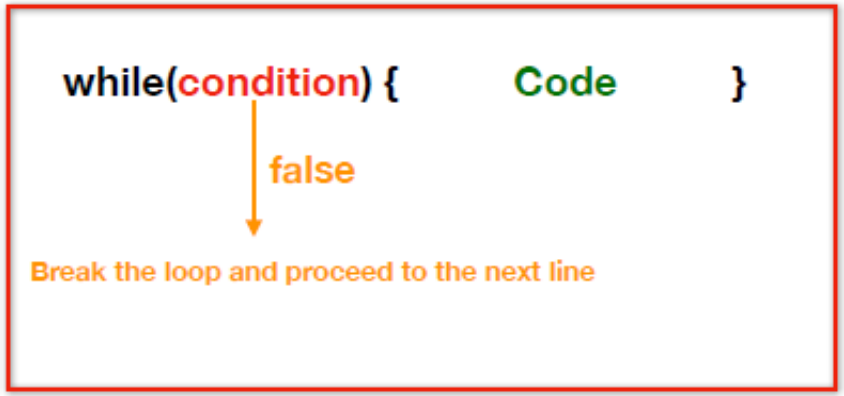
While loop, tekrarlanan bir if ifadesi olarak kabul edilir. Tekrar sayısı sabit değilse, while loop kullanılması önerilir.

Sözdizimi:

```
while  (koşul){
// yürütülecek kod
Artırma / eksiltme deyimi
}
```

While Loop






```
int i = 0;
while (i < 5) {
    System.out.println(i);
    i++;
}
```

While Loop

While Döngüsünün Akış Şeması

Do While Loop

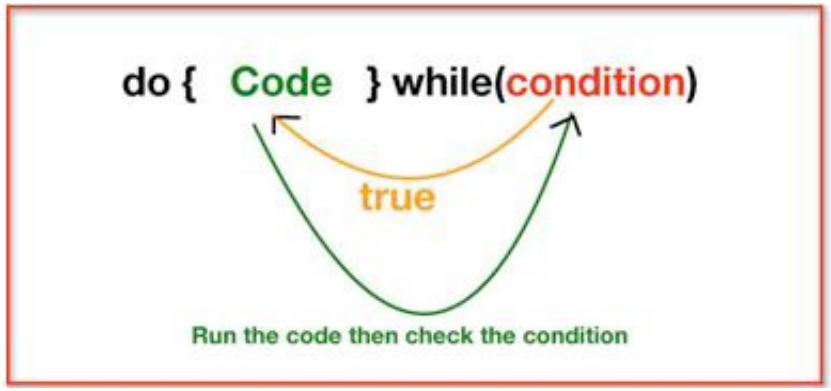
Java *do-while loop*, belirtilen şart **true** olana kadar programın bir bölümünü tekrarlamak için kullanılır. Tekrar sayısı sabit değilse ve loop'u en az bir kez çalıştırmanız gerekiyorsa, **do-while loop** kullanmanız önerilir.

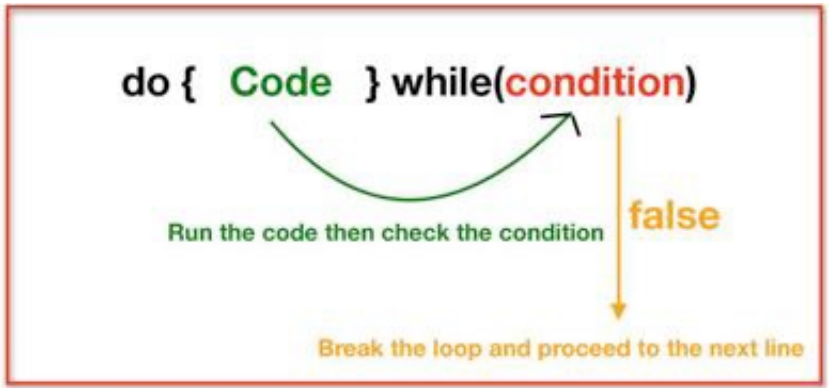
Java do-while döngüsüne çıkış kontrol loop'u da denir. Bu nedenle **while loop** ve **for loop'dan** farklı olarak **do-while**, loop body sonundaki şartı kontrol eder. ***do-while loop* , loop body'**den sonra şart kontrol edildiğinden en az bir kez çalışır.

Program **do** döngüsüne geldiğinde hiç bir şarta bağlı olmadan direk olarak döngüye giriş yapar. Döngünün içinde yer alan işlem satırlarını çalıştırır. Eğer döngünün son satırında yer alan **while** deyimi ile ilgili ifade **true** vermez ise döngü kırılır. **do** döngüsünde eğer sadece tek bir işlem satırı tekrarlanacaksa **({})** işaretlerine gerek yoktur. Döngü, **while** satırındaki ifade **true** olduğu sürece çalışmasına devam eder.

* **TRICK :)** **do** döngüsünde şart kontrolü döngü sonunda yapıldığından şart sağlanmasa bile döngü en az bir defa çalışır. Bu yönü ile **do** döngüsü **for** ve **while** döngülerinden farklıdır.

Do While Loop






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

    int i = 0;
    do {
        System.out.println(i);
        i++;
    } while (i < 7);
}
```

Do While Loop

Do While Loop vs. While Loop

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

    int i = 0;
    do {
        System.out.println(i);
        i++;
    } while (i < 7);
}
```

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

    int i = 17;
    while (i < 7) {
        System.out.println(i);
        i++;
    }
}
```

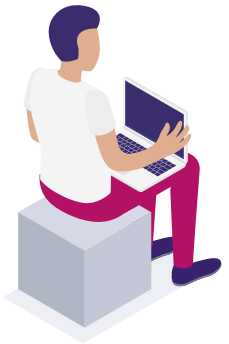
Fark: While Loop, döngünün başlangıcında koşulu kontrol eder ve koşul sağlanırsa body içindeki kodları çalıştırır. Do-While Loop'ta ise, koşul body içerisindeki kodlar 1 kere çalıştıktan sonra kontrol edilir.

Sonuç: Bir While Loop'daki şart False ise, loop hiç çalışmaz (zero execution). Do-While Loop'ta ise, şart False ise de 1 kere çalışır.

THANKS!

Any questions?

HalUk Bilgin I JAVA Backend Developer



BAŞARI GAYRETE AŞIKTIR ☺