Enhanced for Loop

Control Flow kategorisinin dördüncü topic'i: dizi/koleksiyon üzerinde for-each sözdizimi, üç gerçek sınırı (indekse erişilememesi, döngü değişkeninin bir kopya olması ve yapısal değişikliğin ConcurrentModificationException fırlatması, birden fazla koleksiyonu paralel gezememe), ve klasik `for` ile ne zaman hangisinin kullanılacağı.

Başlangıç 20 dk

Son güncelleme: 19 Ekim 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/enhanced-for-loop

---

Enhanced for Loop

Kontrol Akışı kategorisinin dördüncü konusu enhanced for (for-each) döngüsü -- bir dizinin ya da koleksiyonun her elemanını, bir sayaç değişkeni YAZMADAN gezmeyi sağlayan, klasik "for Loop" dersinin üzerine kurulu bir sözdizimi. Klasik `for`, KONUMA (indekse) ihtiyaç duyulan her durumda gereklidir; enhanced for ise yalnızca DEĞERLERİN kendisiyle ilgilenildiğinde -- konumun hiç önemi olmadığında -- aynı işi çok daha az kodla yapar. Bu ders, temel sözdizimini, üç gerçek SINIRINI (indekssizlik, döngü değişkeninin bir kopya olması, paralel gezinememe), ve klasik `for` ile aralarındaki seçimi ele alıyor.

Enhanced for Loop Nedir?

`for (Tip eleman : koleksiyon)` biçimindeki enhanced for, "koleksiyondaki her eleman için" diye okunur -- klasik `for`'un aksine BAŞLANGIÇ/KOŞUL/GÜNCELLEME bölümlerinin hiçbiri yazılmaz, JVM bunları arka planda kendisi yönetir. Herhangi bir dizi ya da `Iterable` arayüzünü uygulayan herhangi bir tip (örn. `List`, `Set`) üzerinde çalışır.

Neden Var?

Klasik `for` ile bir diziyi/koleksiyonu baştan sona gezmek -- `for (int i = 0; i < dizi.length; i++) { dizi[i] ... }` -- yaygın bir kalıptır ama iki gereksiz risk taşır: sınır koşulunu yanlış yazmak (`<=` yerine `<`, bkz. "for Loop" dersindeki "Yaygın Hatalar") ve indeksle DEĞER arasında kafa karışıklığı (`dizi[i]` yerine yanlışlıkla `i`'yi kullanmak). Konuma hiç ihtiyaç duyulmayan -- yalnızca her değerin kendisiyle ilgilenilen -- durumların ÇOĞUNLUĞUNDA bu risklerin ikisi de gereksizdir; enhanced for, bu riskleri sözdiziminden tamamen kaldırır.

Tarihçe

Enhanced for, Java 5'te (2004) generic'ler ve autoboxing'le (bkz. "Wrapper Classes & Autoboxing" dersi) AYNI sürümde, `Iterable` / `Iterator` arayüzleriyle birlikte tanıtıldı -- amaç, o zamana kadar her koleksiyon gezintisinde elle yazılan `Iterator` kullanım kalıbını (`while (it.hasNext()) { Tip eleman = it.next(); ... }`) gizleyip daha kısa bir sözdizimine indirgemekti. Klasik `for`'un YERİNİ almadı, yalnızca konuma ihtiyaç duyulmayan durumlar için bir ALTERNATİF sundu.

Temel Enhanced for Sözdizimi (Diziler)

Bir dizi üzerinde enhanced for, her elemanı sırayla `eleman` değişkenine atar -- indeks hiç yazılmaz, dizinin sınırları JVM tarafından otomatik kontrol edilir (bu yüzden `ArrayIndexOutOfBoundsException` riski YOKTUR).

```
public class EnhancedForArrayExample {
    public static void main(String[] args) {
        int[] scores = {70, 85, 92, 60, 78};

        // "for (Type element : array)" reads as "for each element in array" --
        // no counter, no condition, no update to write yourself.
        for (int score : scores) {
            System.out.println("Score: " + score);
        }

        int total = 0;
        for (int score : scores) {
            total += score;
        }
        System.out.println("Total: " + total);
    }
}
```

Beklenen Çıktı

```
Score: 70
Score: 85
Score: 92
Score: 60
Score: 78
Total: 385
```

Koleksiyonlar Üzerinde Enhanced for

Enhanced for yalnızca dizilerle sınırlı değildir -- `Iterable` arayüzünü uygulayan HER TİP (yani `List`, `Set` dahil hemen hemen tüm koleksiyon sınıfları) üzerinde aynı sözdizimiyle çalışır.

```
import java.util.ArrayList;
import java.util.List;

public class EnhancedForCollectionExample {
    public static void main(String[] args) {
        List<String> languages = new ArrayList<>();
        languages.add("Java");
        languages.add("Kotlin");
        languages.add("TypeScript");

        // Enhanced for works on anything Iterable, not just arrays -- List,
        // Set, and every other collection type included.
        for (String language : languages) {
            System.out.println("Language: " + language);
        }

        int totalLength = 0;
        for (String language : languages) {
            totalLength += language.length();
        }
        System.out.println("Total character count: " + totalLength);
    }
}
```

Beklenen Çıktı

```
Language: Java
Language: Kotlin
Language: TypeScript
Total character count: 20
```

Sınır: İndekse Erişilememesi

Enhanced for, her elemanın DEĞERİNİ verir ama KONUMUNU (indeksini) hiçbir zaman vermez -- "1. sırada X" gibi bir çıktı üretmek gerekiyorsa ya elle bir sayaç değişkeni tutulmalı ya da klasik `for` (bkz. "for Loop" dersindeki "for Döngüsüyle Dizi Üzerinde Gezinme" bölümü) kullanılmalıdır.

```
public class NoIndexAccessExample {
    public static void main(String[] args) {
        String[] winners = {"Alice", "Bob", "Carol"};

        // Enhanced for gives you each VALUE, but never its position -- there
        // is no built-in way to print "1st place: Alice" from inside the loop.
        System.out.println("Without an index (position is missing):");
        for (String winner : winners) {
            System.out.println("Winner: " + winner);
        }

        // A manual counter alongside enhanced for works, but at that point a
        // classic for loop (see the "for Loop" lesson) is usually clearer --
        // you're already tracking an index by hand.
        System.out.println("With a manual counter:");
        int place = 1;
        for (String winner : winners) {
            System.out.println("Place " + place + ": " + winner);
            place++;
        }
    }
}
```

Beklenen Çıktı

```
Without an index (position is missing):
Winner: Alice
Winner: Bob
Winner: Carol
With a manual counter:
Place 1: Alice
Place 2: Bob
Place 3: Carol
```

Sınır: Döngü Değişkeni ve Yapısal Değişiklik

Enhanced for'un döngü değişkeni her elemanın bir KOPYASIDIR -- ona yeni bir değer atamak, alttaki diziyi/koleksiyonu DEĞİŞTİRMEZ. Ayrıca bir `List` üzerinde enhanced for çalışırken listenin YAPISINI değiştirmek (bir eleman eklemek/çıkarmak) genellikle `ConcurrentModificationException` fırlatır -- döngü, koleksiyonun kendi ayaklarının altından değiştiğini böyle tespit eder. Elemanları güvenle çıkarmanın gerçek yolu (`Iterator.remove()`) "Lists" dersindeki "Iterator ve ListIterator" bölümünde ele alınıyor.

```
import java.util.ArrayList;
import java.util.ConcurrentModificationException;
import java.util.List;

public class ModifyingDuringIterationExample {
    public static void main(String[] args) {
        // The loop variable is a COPY of each value -- reassigning it does
        // NOT change the underlying array. A common misconception.
        int[] numbers = {1, 2, 3};
        for (int n : numbers) {
            n = n * 10; // only changes the LOCAL copy, not numbers[]
        }
        System.out.println("Array after the loop (unchanged): " + numbers[0] + ", " +
numbers[1] + ", " + numbers[2]);

        // Removing an element from a List WHILE iterating it with enhanced
        // for throws ConcurrentModificationException -- the loop detects
        // that the collection's structure changed underneath it.
        List<String> items = new ArrayList<>();
        items.add("remove-me");
        items.add("keep");
        items.add("keep");

        try {
            for (String item : items) {
                if (item.equals("remove-me")) {
                    items.remove(item);
                }
            }
        } catch (ConcurrentModificationException e) {
            System.out.println("Caught: " + e.getClass().getSimpleName());
        }
    }
}
```

> Bir listeden eleman çıkarırken `ConcurrentModificationException`'ın HER ZAMAN fırlayacağına güvenmeyin -- listenin SONUNA yakın bir eleman çıkarıldığında, iç sayaçların tesadüfen örtüşmesi yüzünden istisna fırlamayabilir (sessizce yanlış bir sonuç üretebilir). Güvenli olan tek yol, enhanced for içinde YAPISAL değişiklik yapmamaktır.

for-each vs Klasik for: Ne Zaman Hangisi

Konuma ihtiyaç YOKSA (yalnızca değerler işleniyorsa) enhanced for tercih edilmelidir -- daha kısa, daha az hataya açık. Konuma ihtiyaç VARSA (bkz. bir önceki bölüm), ya da diziyi /koleksiyonu YERİNDE değiştirmek gerekiyorsa (bkz. "for Loop" dersindeki "for Döngüsüyle Dizi Üzerinde Gezinme" bölümü), klasik `for` gereklidir.

```
public class ForEachVsClassicForExample {
    public static void main(String[] args) {
        int[] numbers = {2, 4, 6, 8, 10};

        // Same result, two ways.
        System.out.println("Classic for (has an index, more to write):");
        int classicTotal = 0;
        for (int i = 0; i < numbers.length; i++) {
            classicTotal += numbers[i];
        }
        System.out.println("Total: " + classicTotal);

        System.out.println("Enhanced for (no index, less to write):");
        int enhancedTotal = 0;
        for (int number : numbers) {
            enhancedTotal += number;
        }
        System.out.println("Total: " + enhancedTotal);
    }
}
```

Beklenen Çıktı

```
Classic for (has an index, more to write):
Total: 30
Enhanced for (no index, less to write):
Total: 30
```

Sınır: Birden Fazla Koleksiyonu Paralel Gezmek

Enhanced for, TEK bir `Iterable`'ı bir seferde gezer -- iki diziyi/koleksiyonu "AYNI ANDA, aynı konumdan" gezmenin doğrudan bir yolu yoktur (iki enhanced for'u iç içe yazmak, HER elemanı DİĞER koleksiyonun her elemanıyla eşleştirir, bu istenen davranış DEĞİLDİR). Klasik `for`, tek bir paylaşılan indeksle bu problemi doğrudan çözer (bkz. "for Loop" dersindeki "Birden Fazla Değişkenle for" bölümü).

```
public class ParallelIterationLimitationExample {
    public static void main(String[] args) {
        String[] names = {"Alice", "Bob", "Carol"};
        int[] ages = {30, 25, 35};

        // Enhanced for iterates ONE Iterable at a time -- there's no direct
        // way to walk two arrays together, because there's no shared index.
        // Nesting two enhanced-for loops would pair EVERY name with EVERY
        // age, not the matching ones.

        // A classic for loop, sharing a single index across both arrays,
        // solves this cleanly (see "Multiple Variables in for" in the
        // "for Loop" lesson for the two-index version of this pattern).
        for (int i = 0; i < names.length; i++) {
            System.out.println(names[i] + " is " + ages[i] + " years old.");
        }
    }
}
```

Beklenen Çıktı

```
Alice is 30 years old.
Bob is 25 years old.
Carol is 35 years old.
```

Best Practices

- **Konuma (indekse) ihtiyaç yoksa her zaman enhanced for'u tercih edin** -- daha kısa, sınır hatalarına kapalı.
- **Enhanced for içinde bir koleksiyonun YAPISINI (eleman ekleme/çıkarma) değiştirmeyin** -- gerekiyorsa `Iterator.remove()` ya da bir klasik `for` (geriye doğru gezinerek) kullanın.
- **Döngü değişkenine yeni bir değer atamanın diziyi/koleksiyonu DEĞİŞTİRMEDİĞİNİ unutmayın** -- bu yaygın bir yanlış beklentidir.
- **İki koleksiyonu paralel gezmeniz gerekiyorsa klasik `for`'u tek bir paylaşılan indeksle kullanın.**

Yaygın Hatalar

- **Enhanced for'un döngü değişkenini değiştirip diziyi/koleksiyonu güncellediğini SANMAK.** Döngü değişkeni yalnızca bir kopyadır, alttaki veri yapısı ETKİLENMEZ.
- **Enhanced for içinde bir `List`'ten doğrudan `list.remove(...)` çağırıp `ConcurrentModificationException` almak.** Güvenli kaldırma, `Iterator.remove()` gerektirir.
- **Konuma (indekse) ihtiyaç varken enhanced for kullanmaya çalışıp elle sayaç tutmaya zorlanmak.** Bu durumda klasik `for` zaten daha az kod gerektirir.
- **İki diziyi eşleştirmek için iki enhanced for'u İÇ İÇE yazmak.** Bu her elemanı diğerinin HER elemanıyla eşleştirir -- paralel gezinme değil, kartezyen çarpım üretir.

Özet, Cheat Sheet ve Terimler Sözlüğü

Enhanced for (`for (Tip eleman : koleksiyon)`), bir dizi/koleksiyonun her elemanını, indeks yazmadan gezmeyi sağlar -- `Iterable` uygulayan her tip üzerinde çalışır. Üç gerçek sınırı vardır: KONUMA erişilemez, döngü değişkenine atama alttaki veriyi DEĞİŞTİRMEZ (ve yapısal değişiklikler genellikle `ConcurrentModificationException` fırlatır), ve birden fazla koleksiyon PARALEL gezilemez. Bu üç durumda klasik `for` (bkz. "for Loop" dersi) hâlâ doğru araçtır.

Hızlı referans:

```
for (int eleman : dizi) {
    // yalnızca DEĞER, indeks yok
}

for (String eleman : liste) {
    // Iterable uygulayan her tip üzerinde çalışır
}

// YANLIŞ -- alttaki diziyi değiştirmez:
for (int eleman : dizi) {
    eleman = eleman * 2;
}

// Konuma ihtiyaç varsa klasik for:
for (int i = 0; i < dizi.length; i++) {
    System.out.println(i + ": " + dizi[i]);
}
```

Terimler Sözlüğü

Enhanced for (for-each) — Bir dizi/koleksiyonun her elemanını, indeks yazmadan gezen döngü sözdizimi.

Iterable — Enhanced for ile gezilebilen her tipin uyguladığı arayüz (`List` , `Set` dahil).

ConcurrentModificationException — Bir koleksiyon, üzerinde enhanced for çalışırken YAPISAL olarak değiştirildiğinde genellikle fırlatılan çalışma zamanı istisnası.