Semih Kırdinli

Java Developer at Solvia

Linkedin : https://www.linkedin.com/in/semihkirdinli/
Github : https://github.com/semih

Method References

- Metot Referans Çeşitleri
- Örnek Çalışmalar

Method References

Anonymous metotlar oluşturmak için lambda expression kullanmıştık. Bazen lambda expression hiçbir şey yapmaz, sadece mevcut metodu çağırır. Bu durumlarda mevcut metodun adıyla ifade etmek genellikle daha açık bir yazım şekli olacaktır. Metot referansları da tam olarak bunu sağlar. Bir metodun referans olarak verilebilmesi için ise **::** ikilisi kullanılır.

Aşağıdaki gibi bir kişi (Person) sınıfı düşünelim.

```
public class Person {
    LocalDate birthday;
    public int getAge() {
        return 0;
    }
    public LocalDate getBirthday() {
        return birthday;
    }
    public static int compareByAge(Person a, Person b) {
        return a.birthday.compareTo(b.birthday);
    }
}
```

Person sınıfının birer nesnesi olan kişilerden bir liste olduğunu ve bu listeyi yaşa göre sıralamak istediğimizi düşünelim.

Sıralama işlemi için **Arrays** sınıfına ait sort metodu kullanılabilir.

```
static <T> void sort(T[] a, Comparator<? Super T> c)
```

main.exercise1.MethodReferencesTest.java

```
List<Person> personList = Person.createPersonList();
Person[] personListAsArray = personList.toArray(new Person[personList.size()]);

// 1. Comparator ayrıca oluşturulup, iki tane Person parametresi gönderilerek
// compareTo metoduyla sıralanması sağlanabilir.
class PersonAgeComparator implements Comparator<Person> {
    public int compare(Person a, Person b) {
        return a.getBirthday().compareTo(b.getBirthday());
    }
}
Arrays.sort(personListAsArray, new PersonAgeComparator());

// 2. Comparator bir @FunctionalInterface olduğu için new ile yeni bir instance oluşturmak yerine
// lambda expression olarak yazılabilir.
Arrays.sort(personListAsArray, (Person a, Person b) -> {
    return a.getBirthday().compareTo(b.getBirthday());
});

// 3. İki Person nesnesinin doğum tarihlerini karşılaştıran yöntem, Person.compareByAge olarak zaten mevcuttur.
Arrays.sort(personListAsArray,
        (a, b) -> Person.compareByAge(a, b)
);

// 4. Lambda expression'ın mevcut bir metodu çağırmasından dolayı,
// lambda expression yerine method reference kullanılabilir.
Arrays.sort(personListAsArray, Person::compareByAge);
```

Metot Referans Çeşitleri

Kind	Syntax	Examples

Statik metotlara referans vermek	*ContainingClass*::*staticMethodName*	`Person::compareByAge`
		`MethodReferencesExamples::appendStrings`
		`String::concat`

Statik olmayan metotlara referans vermek	*containingObject*::*instanceMethodName*	`myComparisonProvider::compareByName` `myApp::appendStrings2`
Constructor'a referans vermek	*ClassName*::`new`	`HashSet::new`

`main.exercise1.MethodReferencesExamples.java` sınıfındaki metotları, metot referansı olarak kullanalım.

BiFunction, java.util.function paketindeki çok sayıda functional interface'ten biridir. BiFunction interface'i iki parametre kabul eden ve bir sonuç üreten bir lambda ifadesini veya metot referansını temsil edebilir.

Bir diğer functional interface olan **Supplier** parametre almaz ve geriye bir nesne döner.

`main.exercise1.SupplierExample.java` sınıfında bulunan List<Person> tipini Set<Person>'a dönüştüren metodun Supplier parametresini lambda expression ile ve metot referans ile yazalım.

Örnek Çalışmalar

`main.exercise2` paketinde yer alan örnekleri sırayla çözelim.

1. Dünyadaki en kalabalık başkent şehrini bulunuz.
2. Şehir sayılarına göre ülkeleri tersten sıralayınız.
3. Türü **sadece** "Drama" ve "Comedy" olan filmleri listeleyiniz.
4. Yıllara göre filmleri gruplayın ve sıralayınız.
5. Ülkeleri nüfus yoğunluklarına göre tersten sıralayarak nüfusu sıfır olan ülkeleri listeden çıkarınız.
6. En çok film içeren yılı bulunuz.
7. Her bir kıtaya ait en kalabalık şehri bulunuz.