Lambda

Javada Lambda Kullanımı

Mehmet Bektaş
mehmet.bektas@assistt.com.tr

Eğitim İçeriği

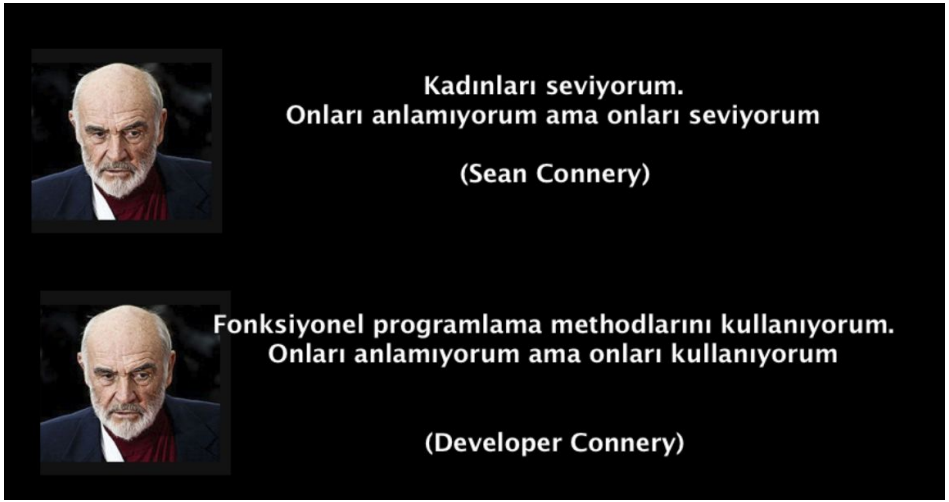
1. Lambda Nedir? Neden kullanmalıyız?
2. Functional Interface nedir?
3. Method reference nedir?
4. Collection işlemlerinde lambda kullanımı

1. Lambda Nedir? Neden kullanmalıyız?

- Functional programming yapmaya olanak sağlar.
- Nesne bağımlılığından kurtarır.
- Okunabilir kod yazmaya olanak sağlar.
- Paralel işlem yapmaya olanak sağlar.

Functional Programming

- En basit açıklama ile matematik fonksiyonlarıdır.
- Global değişkenler, nesneler (objects) yoktur.
- Sadece fonksiyonun çalışabilmesi için gerekli değerler yani girdiler vardır.



Örnek 1

```
function toplam (a, b) {
 return a + b;
}
```

- Global değişkenlerin olmayışı akışın durumsuz (stateless) olmasını sağlar.
- Geriye sadece sonuç döndürülür.
- Bu durum aynı zamanda yan etkileri (side-effects) ortadan kaldırır.
- Fonksiyonel programlamaya bir örnektir.
- Buna aynı zamanda saf fonksiyon (pure-function) denir.

Örnek 2

```
var sonuc = 0;
sonuc = toplam(1, 2);


function toplam (a, b) {
 sonuc = a + b;
}
```

- Sonuç değişkenin durumu değişiyor.
- Fonksiyon hem stateless değildir hem de saf fonksiyon değildir.
- Fonksiyonel programlamaya bir örnek değildir.

OOP ile FP arasındaki farklar

- OOP sonuç odaklıdır, değişkenler, nesneler ve atamalar vardır, FP'de sadece fonksiyonun çalışabilmesi için gerekli değerler yani girdiler vardır.
- Geriye sadece sonuç dönerler ve herhangi bir değişkenin veya objenin durumu değişmez.
- Fonksiyonel programlamada metodlar ayrı scopelarda çalışır. FP programın farklı farklı biçimlerde çalışması sorununu ortadan kaldırır. Çünkü ortam bağımsız sadece aldığı girdilere göre sonuç döner. Bu her ortamda aynı olur.

Functional Interface

- Tek bir soyut metodu bulunan arayüzlere fonksiyonel arayüz denir.
- İki veya üç değil, yalnızca bir tane soyut metodu olmalı.
- Interfacede **@FunctionalInterface** annotasyonu kullanılabilir. **@Override** gibi opsiyoneldir.

```
@FunctionalInterface // Opsiyonel
interface Foo{
    int apply(int x, int y);
}
```

Örnek

```
public void updateDurum(List<ModelMebKurum> modelMebKurumList) {
	String sql = "UPDATE MHRS.DK_MEB_KURUM_LISTESI SET OGRENCI_SAYISI = ?, DURUM = ?, GUNCELLENME_TARIHI = SYSDATE WHERE KURUM_KODU = ?";
	ParameterizedPreparedStatementSetter<ModelMebKurum> parameterSetter = new ParameterizedPreparedStatementSetter<ModelMebKurum>() {
		@Override
		public void setValues(PreparedStatement ps, ModelMebKurum argument) throws SQLException {
			ps.setInt(1, argument.getOgrenciSayisi());
			ps.setBoolean(2, argument.isDurum());
			ps.setInt(3, argument.getKurumKodu());
		}
	};
	getJdbcTemplate().batchUpdate(sql, modelMebKurumList, modelMebKurumList.size(), parameterSetter);

}
```

Method Reference

- Bir static metodu, Functional Interfacedeki metod yerine kullanabilmektir.
- Lambda ifadeleri yazılırken, tek metoda sahip arayüzün (fonksiyonel arayüz) metod girdi ve çıktısı baz alınmaktadır. Eğer daha önce yazdığınız bir metodun girdi ve çıktısı, bir fonksiyonel arayüz metodunun girdi ve çıktısına birebir uyuyorsa, o metod bir lambda deyimi yerine kullanılabilmektedir.

Java Dilinde Fonksiyonel Programlama

- Fonksiyonel interface, lambda operatörü veya method reference kullanılarak yapılabilir.
- Her üç durumda da aslında aynı işlem yapılmış olur.
- Lambda oparatörü ve method reference, fonksiyonel interfacelerin farklı biçimde kullanılmasıdır.

Java - Javascript Bir Fark

```
var kisiList = [
        "Ahmet",
        "Ali",
        "Yiğit",
        "Mine",
        "Merve"
];
int erisilemeyenDegisken = 0;
kisiList.forEach(function(kisi,i){
        console.log(i + ": " + kisi);
        erisilemeyenDegisken++; //Bu kısım
çalışmaz
});
```

```
0: Ahmet
1: Ali
2: Yiğit
3: Mine
4: Merve
```

- Döngü içerisinde, dışardaki bir metoda veya değişkene ulaşılamaz.
- Fonksiyon dışarıya kapalıdır.
- Sadece kendisinden istenilen işlemi yapar.

Kaynakça

- https://javabrains.io/courses/java_lambdabasics/
- https://kodedu.com/2014/03/java-8-lambda-expressions-and-functional-interfaces/
- https://kodedu.com/2014/09/java-8-method-reference/

Teşekkürler