NESNEYE YÖNELİK PROGRAMLAMA

Lambda ifadeleri

Emir Öztürk

LAMBDA İFADESİ

- Java 8
- Kod bloğu
- Parametre alıp değer döndürür.
- Metot gövdesi

Lambda ifadeleri desteği Java 8'den itibaren gelmiştir. Belirli bir fonksiyonun belirli bir akış üzerinde gerçekleştirilmesi için kullanılabilir. Parametreli olarak tanımlanabilir ve sonuç değeri üretebilir.

LAMBDA İFADESİ

- @FunctionalInterface
  - Zorunlu değil
- Tek metotlu arayüz
- Bu arayüzü kullanan ifadeler

Bir lambda ifadesi ile kullanılacak metodun kullanacağı bir arayüz olması gerekmektedir. Arayüzün tanımlanmasında başına @FuncitonalInterface yazılmaktadır. Bu anotasyonun kullanılması zorunlu değildir.
Arayüzün içerisinde bir metot olmalıdır. Bu arayüzü kullanan metotlar lambda ifadeleri ile kullanılabilmektedir.

LAMBDA İFADESİ

```
@FunctionalInterface
interface MatematikIslem{
  int islem(int a,int b);
}
```

- MatematikIslem toplama = (a,b) -> a+b;
- MatematikIslem toplama =(int a,int b)->{return a+b;};

LIST.FOREACH

- Bir liste üzerinde döngü kurma
- List<int> liste = {1,2,3,4};

```
liste.foreach(
  (eleman) -> System.out.println(eleman)
  );
```

Bir listenin elemanları üzerinde döngü kurmak için liste.foreach kullanılabilmektedir. Her elemanı ifade edecek bir değişken adı parantez içerisinde verilir. -> “arrow notation” kullanılarak yapılmak istenen işlem gerçekleştirilir.

SIRALAMA

```
List<Kisi> liste;
for(int i=0;i<liste.count();i++)
  for(int j=0;j<liste.count();j++)
    if(liste[i].yas>liste[j].yas){
      int temp = liste[i];
      liste[i] = liste[j];
      liste[j] = temp;
    }
Collections.sort(liste,  (eleman1,eleman2) -> { return eleman1 > eleman2;});
Collections.sort(liste,(e1,e2)->e1.yas>e2.yas);
```

Sıralamanın uzun ve kısa hali

JAVA.UTIL.COLLECTIONS

- Collections.addAll(liste,elemanlar)
- Collections.binarySearch(liste, "Aranan")
- Collections.indexOfSubList(liste,altListe)
- Collections.max(liste)
- Collections.min(liste)
- Collections.replace(liste,eski,yeni)
- Collections.reverse(liste)
- Collections.shuffle(liste)

addAll metodu ile listenin içerisine birden fazla eleman bir kerede eklenebilir.
binarySearch ile liste içerisinde ikili arama gerçekleştirilebilir.
indexOfSubList metodu liste içerisinde başka bir listenin geçtiği indeksi verir.
max ve min metotları en büyük ve en küçük elemanları döndürür.
replace bir liste içerisinde verilen elemanları istenilen başka bir eleman ile değiştirir.
reverse ile bir liste tersine çevrilebilir.
shuffle ile bir liste karıştırılabilir.

STREAM

- Toplu işlemlerin gerçekleştirilmesi
- Map-Reduce
- Fonksiyonel kullanım
- Liste.stream()

Herhangi bir listenin stream'e çevrilmesi ile fonksiyonel programlama kullanılabilmektedir. Bu stream üzerinde map-reduce temelli işlemler de yapılabilmektedir.

LİSTE ÜZERİNDE KOŞULLU SEÇİM

- List<int> liste;
- var sonuc = liste.stream().foreach(eleman -> eleman * 10);
- Sonucun dönüştürülmesi

liste stream'e çevrildikten sonra foreach metodu ile her elemanın 10 katı yeniden bir değişkene atılmaktadır.
Bu değişkenlerin tekrar listeye çevrilmesi için bir işlem daha yapılması gerekmektedir.

COLLECT

- Stream'e dönüştürülmüş listelerin üzerine uygulanan fonksiyon sonuçları
- Yeni koleksiyon oluşturma
- .collect
- sonuc.collect(Collectors.toList());
- sonuc.collect(Collectors.toArray());

Lambda ifadesinin sonuçlarını yeniden bir listeye ya da diziye çevirmek için collect kullanılır. Sonuçlar Collectors.toList() liste, Collectors.toArray() ile de dizi olarak elde edilir.

FILTER

- Belirli koşulların sağlanması
- Yeni bir liste eldesi
- steam ve collect ile birlikte kullanılabilir

```
var ondanBuyuklerListesi = liste.stream().filter(eleman -> eleman > 10).toList()
```

Liste elemanları içerisinden seçim yapmak için filter kullanılabilir. Filter içerisine istenilen bir koşul yazılmaktadır. Filter işleminden sonra bu koşula uyan elemanlardan yeni bir koleksiyon döndürülmüş olur. Daha sonra collect metodu ile dizi veya liste olarak bu sonuçlar elde edilebileceği gibi üzerlerinde de döngü ile gezilebilir.

FINDFIRST

- Liste üzerinde filtrelenmiş bir sonucun ilk değerinin eldesi
- Collect'ten farklı olarak eleman döndürür.
- liste.stream().filter(eleman -> eleman > 10)
    - .findFirst();

Stream metotları art arda kullanılabilir. filter ile elde edilen sonuçlardan sonra findFirst metodu kullanılabilir. Bu metot ile elde edilen sonuçların ilki alınabilir.

ORELSE

- FindFirst kullanıldığında sonucun boş olması durumu
- Varsayılan değer döndürülebilir

```
liste.stream()
      .filter(eleman -> eleman > 10)
      .findFirst()
      .orElse(null);
```

Filter'dan sonra elde edilen sonuç eleman içermeyebilir. Bu sebeple findFirst metodu değer döndürmeyecektir. Bunun için bir değer bulunamadığı takdirde varsayılan olarak ne döndürüleceği orElse ile tanımlanabilir.

SORTED

- Elde edilen sonuçların düzenlenmesi
- `liste.stream().filter(eleman -> eleman > 10).sorted();`
- `liste.stream().filter(eleman -> eleman > 10).sorted().findFirst().orElse(null);`

METOTLAR

- `liste.stream().average()`
- `liste.stream().max()`
- `liste.stream().min()`
- `liste.stream().foreach()`
- *`liste.stream().max()`*`.ifpresent()`
- `liste.stream().mapToInt(Integer::parseInt)`
- `liste.stream().mapToDouble(Double::parseDouble)`

average max ve min, ortalama maksimum ve minimum bulmak için kullanılır.
foreach ile liste elemanları üzerinde sırayla gezilerek işlem yapılabilir.
ifpresent ile bulunan bir sonuç varsa bir işlem gerçekleştirilebilir.
mapToInt ile liste elemanlarını integera dönüştürme işlemi gerçekleştirilir.
maptToDouble ile liste elemanlarını double'a dönştürme işlemi gerçekleştirilir.

FLATMAP

- Listelerin listesi
- Tek boyuta indirgeme
- List<List<string>> isimlerListesi
- Örn: [[“emir”,”ahmet”],[“esat”,”tufan”],[“eren”]]
- isimlerListesi.stream().flatMap(Collection::stream).collect(Collectors.toList());
- Sonuç: [“emir”,”ahmet”,”esat”,”tufan”,”eren”]

List’lerden oluşan bir listeyi tek boyuta indirgemek için flatmap kullanılır.

SAYMA, ELEME, SINIRLAMA

- List<int> liste;
- liste.stream().filter(…).skip(n).limit(n).count();
- Sayı
- Eleman geçme
- Sınırlı sayıda eleman alma
- .collect(Collectors.toList());

Sonucun eleman sayısı için count, belirli sayıda elemanı geçmek için skip, belirli sayıda eleman almak için ise limit kullanılır.

ÖRNEK

- Liste içerisindeki 50den büyük sayıların 5. elemandan 10. elemana kadar olanlarını alan ve bu sayıların karelerini küçükten büyüğe sıralayıp bir string dizisi olarak döndüren metot