Java8

@04/07/2022

```
list.stream().forEach(System.out::print)
list.forEach(System.out::print)
```

@05/07/2022

```
Map<String, Object> mapList = new LinkedHashMap<String, Object>();
mapList.put("1", "İstanbul");
mapList.put("2", "İzmir");
mapList.put("3", "Bursa");
mapList.put("4", "Ankara");
//******* 4- Iterator *******
// Key benzersiz olmalı: ve key için(Set) kullanacağım
Set<String> set = mapList.keySet();
Iterator iterator = set.iterator();
while (iterator.hasNext()) {
System.out.println(mapList.get(iterator.next()));
}

//****** 5- M-E-l-es *******
for (Map.Entry<String, Object> temp : mapList.entrySet()) {
System.out.println(temp);
}

//****** 6- Stream ForEach ******
mapList.entrySet().forEach(System.out::println);
```

@07/07/2022

Lambda Expression (Lambda Function)

```
(x)->{x*5};
```

Lambda Expression yazarken dikkat edeceğimiz hususlar
1-) @FunctionInterface ==> 1 tane gövdesiz metot yaz
2-) Parametre sayısı ve türü çağıranda aynı olmalı (temp)=>{}
3-) return varsa return türü yazılır

@FunctionalInterface ==> interface 1 tane gövdesiz metot yazabilirsin başka

```
@FunctionalInterface
interface ILambdaExpressionData {
public void deneme(String adi);
}
```

```
ILambdaExpressionData expressionData2 = (temp) -> {
System.out.println(temp + " function interface yapısı");
};
```

Stream API

Stream: 2 grup çalışma sitili vardır.
1-) Sequential (Senkron: sadece 1 işlem yapar)
2-) Paralel (Asenkron:Aynı anda birden fazla işlem yapabilir.)

Çeşitleri: intStream,LongStream,DoubleStream,Stream

```
int toplam = listem2.stream()
            .filter(temp -> temp % 2 == 0)
            .reduce(0, (x, y) -> x + y);
Predicate<Integer> ciftSayilar = temp -> temp % 2 == 0;
int toplam2 = listem2.stream()
              .filter(ciftSayilar)
              .reduce(0, (x, y) -> x + y);
```

@08/07/2022

```
List<String> listem = Arrays.asList("malatya", "istanbul", "ankara", "malatya", "bursa", "bolu");

// collect(Collectors.toList()) ==> stream objesini Listeye Çevir
List<String> listem2 = listem.stream().collect(Collectors.toList());

// Sorted() ==> Küçükten büyüğe sıralama
// List<String> listem2 = listem.stream().sorted().collect(Collectors.toList());

// reverse() ==> Büyükten küçüğe sıralama
List<String> listem2 = listem.stream().sorted(Comparator.reverseOrder()).collect(Collectors.toList());

// limit() ==> belirlenen miktar kadar veriyi bana list olarak döner
List<String> listem2 = listem.stream().limit(3).collect(Collectors.toList());

// filter() ==> Var olan listeden sadece benim istediklerimi getir.
// filter veriler üzerinde bir değişklik yapmıyor.
// map ==> lambda expression (lambda function kullanıyoruz)

// filter((temp) -> "malatya".equals(temp)) --> listede malatya olanları getirir.
List<String> listem2 = listem.stream().filter((temp) -> "malatya".equals(temp)).collect(Collectors.toList());

// filter((temp) -> !"malatya".equals(temp)) --> listedeki malatya geçmeyenleriiiiii bana listele
List<String> listem2 = listem.stream().filter((temp) -> !"malatya".equals(temp)).collect(Collectors.toList());

// sorted() --> listeyi küçükten büyüğe sıralama listele
List<String> listem2 = listem.stream().sorted().collect(Collectors.toList());

// sorted(Comparator.reverseOrder()) --> listeyi büyükten küçüğe sıralama
List<String> listem2 = listem.stream().sorted(Comparator.reverseOrder()).collect(Collectors.toList());

// distinct() ==> Var olan listedeki tekrar eden verileri yazdırmayan liste
List<String> listem2 = listem.stream().distinct().collect(Collectors.toList());
```

```
// map((temp) -> temp.toUpperCase()) --> listedeki verileri büyük harf'e çevirmek için kullanıyoruz.
List<String> listem2 = listem.stream().map((temp) -> temp.toUpperCase()).collect(Collectors.toList());
```

@22/07/2022

```
public interface IDaoConnection<T> {
// Gövdeli metot interface
  default Connection getInterfaceConnection() {
    return DatabaseConnection.getInstance().getConnection();
  }
}
```

Kadir - Tolga