Java 8 Streams PART-I - by Mr. RAGHU

1. CREATING A STREAM IN JAVA 8

```
//empty stream
Stream<String> st1=Stream.empty();
st1.forEach(System.out::println);

//Array to Stream
String[] arr1= {"W","E","L","C","O","M","E"};
Stream<String> st2=Arrays.stream(arr1);
st2.forEach(System.out::println);

//var-args to stream
Stream<String> st3=Stream.of("T","O","A","L","L");
st3.forEach(System.out::println);

//List/Arrays.asList to Stream
List<String> al1=Arrays.asList("N","I","T");
Stream<String> st4=al1.stream();
st4.forEach(System.out::println);
List<String> al2=List.of("R","A","G","H","U");
Stream<String> st5=al2.stream();
st5.forEach(System.out::println);

//Map to Stream
Map<String,Integer> map1=
Map.of("ENG",85,"MAT",66,"JAVA",99,"HIN",36);
Stream<Entry<String,Integer>> st6=
map1.entrySet().stream();
st6.forEach(System.out::println);

Stream<String> st7=map1.keySet().stream();
st7.forEach(System.out::println);
Stream<Integer> st8=map1.values().stream();
st8.forEach(System.out::println);

//Set to Stream
Set<String> set1=Set.of("H","E","L","O");
```

```
Stream<String> st9=set1.stream();
st9.forEach(System.out::println);


//using Stream Builder
Stream<String> st10=Stream.<String>builder()
.add("R").add("A")
.add("G").add("H")
.add("U")
.build();
st10.forEach(System.out::println);


//using generator **
Stream.generate(()->"RAGHU").limit(4)
.forEach(System.out::println);

//using iterator
Stream.iterate(5, i->i+5).limit(10)
.forEach(System.out::println);


//Stream from file
Path path1=Path.of("message.txt");
Stream<String> st11=Files.lines(path1);
st11.forEach(System.out::println);

String s1="Hi-All-How-Are-You!";
Stream<String> st12= Pattern.compile("-
").splitAsStream(s1);
st12.forEach(System.out::println);
```


2: STREAMS BASIC OPERATONS IN JAVA 8

```
List<Employee> emps=Arrays.asList(
new Employee(54,"K",90.5), new Employee(10,"R",95.5),
new Employee(35,"L",89.0), new Employee(35,"S",5.0),
new Employee(35,"Z",36.0), new Employee(95,null,9.6),
new Employee(31,"D",45.4), new Employee(28,null,5.8)
            );

emps.stream()
.filter(e->e.getEname()!=null)
```

```
.sorted((e1,e2)->
Double.valueOf(e2.getEsal()).compareTo(e1.getEsal()))
.map(e->e.getEid()+")"+e.getEname()+"--"+e.getEsal())
.skip(3).limit(2)
      .peek(System.out::println)
      .collect(Collectors.toList());
```

3. COLLECTORS USING STREAM IN JAVA 8

```
//----Stream to List---
List<String> al1=
Stream.of("A","B","C","D","A","K").collect(Collectors.toList());
al1.forEach(System.out::println);

//----Stream to Set---
Set<String> set1=
Stream.of("A","B","C","D","A","K").collect(Collectors.toSet());
set1.forEach(System.out::println);

//----Stream to Map---
Map<String,String> map1= Stream.of("A","B","C","D").collect(
Collectors.toMap(Function.identity(), e->e+"Data"));
           map1.forEach((k,v)->System.out.println(k+"-"+v));

//--Stream to Specific collection
Stream.of("A","B","C","D","A","K").collect(Collectors.toCollection(A
rrayList::new));

Stream.of("A","B","C","D","A","K").collect(Collectors.toCollection(H
ashSet::new));


//--------List-to--Map----------
     List<Employee> emps=Arrays.asList(
                       new Employee(54,"K",10.5),
                       new Employee(10,"R",15.5),
                       new Employee(35,"S",5.0)
                       );
      Map<Integer,Employee> map2=emps.stream()
      .collect(
            Collectors.toMap(Employee::getEid, Employee::toThis));
map2.forEach((k,v)->System.out.println(k+"-"+v));

//----List - to - Set-------
Set<Employee> set2=emps.stream().collect(toSet());
set2.forEach(System.out::println);
```

```
//-- Data operations sum,min,max,avg,total rows-----------
DoubleSummaryStatistics totalSal=emps.stream()
                        .collect(
            Collectors.summarizingDouble(Employee::getEsal));
            System.out.println(totalSal);
            System.out.println(totalSal.getSum());
            System.out.println(totalSal.getMax());
            System.out.println(totalSal.getMin());
            System.out.println(totalSal.getAverage());
            System.out.println(totalSal.getCount());


//---Read Specific data as List---------
            emps.stream()
            .map(Employee::getEname)
            //.map(Employee::getEid)
            //.map(Employee::getEsal)
            .collect(Collectors.toList())
            .forEach(System.out::println);

//--## counting ## - ---
            String arr[]=
{"RAGHU","NIT","JAVA","RAGHU","RAGHU","NIT"};
            System.out.println(
            Stream.of(arr).collect(Collectors.counting())
            );

//--## grouping ## Find no.of occurrences of every element ---
      Stream.of(arr).collect(
                        Collectors.groupingBy(Function.identity(),
Collectors.counting())
                        ).forEach((k,v)->System.out.println(k+"="+v));


//--##Joining ## -----------
            String st1=
Stream.of(arr).collect(Collectors.joining(","));
            System.out.println(st1);

//------## Partition ##--------------
Map<Boolean,List<String>> map3=
            Stream.of(arr).distinct()
            .collect(Collectors.partitioningBy(e->e.length()%2==0));
            System.out.println(map3);
```