

Java 8 Features

www.youtube.com/@ProgrammingKT

```
* 1. filter
* Example: Finding All Factors of 3 from given input
List<Integer> list = IntStream.of(2,3,4,5,6,7,9,12,15)
                                .filter(n->n\%3==0)
                                .boxed()
                                .toList();
```

System.out.println(list);

```
* 2. dropWhile
 * Example: Returning all numbers after condition fails
List<Integer> dropWhile = IntStream.of(2,4,6,7,8,10,12,14)
                                    .dropWhile(n->n\%2==0)
                                    .boxed()
                                    .toList();
System.out.println(dropWhile);
```

```
* 3. takeWhile
 * Example: Returning all numbers Before condition fails
 */
List<Integer> takeWhile = IntStream.of(2,4,6,7,8,10,12,14)
                                     .takeWhile(n->n\%2==0)
                                     .boxed()
                                     .toList();
System.out.println(takeWhile);
```

```
* 4. map
* Example: Convert All Integers from String to Integer
List<String> listOfNumStr = List.of("1","2","3","5","8");
List<Integer> listOfNum = listOfNumStr.stream()
                                      .map(s->Integer.parseInt(s))
                                      .filter(n->n\%2==0)
                                      .toList();
```

System.out.println(listOfNum);

```
* 5. mapToObj -> to convert one type to another type
* Example: Convert All Integers from String to Integer
String str = "ProgrammingKT";
Map<Character, Long> charCount = str.chars()
                         .mapToObj( n -> (char) n )
                         .collect(Collectors.groupingBy(
                                           Function.identity(),
                                           Collectors.counting())
```

```
6. flatMap -> is used to process inner collection
 Example: Convert List Of Lists to List
List<List<Integer>> listOfLists = List.of(List.of(1,2,3),List.of(5,3,6),List.of(9,7,8));
List<Integer> listOfInt = listOfLists.stream()
                                         .flatMap(l->l.stream())
                                         .toList();
System.out.println(listOfInt);
```

```
* 7. flatMapToInt
* Example: Finding Sum from given List of Lists
List<List<Integer>> listOfLists = List.of(List.of(1,2,3),List.of(5,3,6),List.of(9,7,8));
int sum = listOfLists.stream()
                   .flatMapToInt(l->l.stream().mapToInt(Integer::valueOf))
                   .sum();
System.out.println(sum);
```

```
* 8. Parallel -> To execute logic in multithreaded env
List<List<Integer>> listOfLists = List.of(List.of(1,2,3),List.of(5,3,6),List.of(9,7,8));
listOfLists.stream()
          .parallel()
          .flatMap(l -> {
             System.out.println(Thread.currentThread().getName());
            return l.stream();
           }).toList();
```

```
* 9. distinct -> used to eliminate duplicates
*/
List<Integer> listOfNumbers = List.of(1,4,2,6,7,8,5,4,6,8,9,10,13,12,15);
System.out.println(listOfNumbers);
List<Integer> uniqueList = listOfNumbers.stream()
                                          .distinct()
                                          .toList();
System.out.println(uniqueList);
```

```
* 10. sorted -> used to sort given input
*/
List<Integer> listOfNumbers = List.of(1,4,2,6,7,8,5,4,6,8,9,10,13,12,15);
List<Integer> uniqueSortedList = listOfNumbers.stream()
                                                 .distinct()
                                                 .sorted()
                                                 .toList();
System.out.println(uniqueSortedList);
```

www.youtube.com/@ProgrammingKT

```
* 11. iterate
 Example: Printing numbers from 1 to 10
*/
IntStream.iterate(0, i->i+1)
           .limit(10)
           .forEach(n->System.out.println(n));
```

```
* 12. range
 Example: Print numbers from 10 to 19
*/
IntStream.range(10, 20)
   .forEach(n->System.out.println("Range OutPut "+n));
```

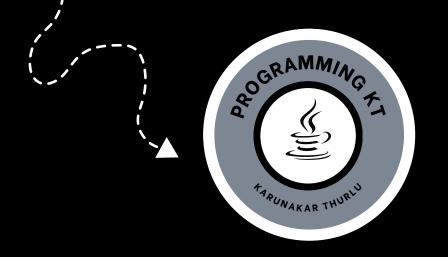
```
/*
 * 13. rangeClosed
 * Example : Print numbers from 10 to 20
 */
```

```
IntStream.rangeClosed(10, 20)
   .forEach(n->System.out.println("RangeClosed OutPut "+n));
```

```
* 14. limit -> Frequently used with iterate
* Example: Print numbers from 1 to 10
*/
IntStream.iterate(1, i -> i+1)
          .limit(10)
          .forEach(n -> System.out.println(n));
```

```
* 15. skip -> ignore first n elements from given source
*/
List<Integer> listOfNumbers = List.of(1,4,2,6,7,8,5,4,6,8,9,10,13,12,15);
System.out.println(listOfNumbers);
List<Integer> skipedList = listOfNumbers.stream()
                                          .skip(3)
                                          .toList();
System.out.println(skipedList);
```

```
* 16. boxed -> used to convert from primitive datatype to
     wrapper datatype
*/
int[] intArray = \{2,4,7,9,5\};
List<Integer> convertedArray = Arrays.stream(intArray)
                                        .boxed()
                                        .toList();
System.out.println(convertedArray);
```



Thanks for Reading

Fallow Me and Subscribe @ProgrammingKT to get more content like this

www.youtube.com/@ProgrammingKT