***Terminal operations***

* *Terminal operations used perform last operation on streams*
* *Used to get End result*

***Collect***

* *It is used to have collectors related methods inside of it*

List<Integer> list = Arrays.asList(1, 2, 3);  
list.stream().skip(1).collect(Collectors.toList());  
list.stream().skip(1).toList();

***forEach***

* *It is used to have consumer related methods inside of it*

List<Integer> list = Arrays.asList(1, 2, 3);

list.stream().forEach(x -> System.out.println(x));

***reduce***

* *It is used to computate and provide single result*
* *It is used have binaryOperation interface related method inside*

List<Integer> list = Arrays.asList(1, 2, 3);

Optional<Integer> optionalInteger = list.stream().reduce((x,y)->x+y);  
System.*out*.println(optionalInteger.get());

***Count***

* *It is used to give count of the stream*

List<Integer> list = Arrays.asList(1, 2, 3);

System.*out*.println(list.stream().filter(x -> x <= 2).distinct().peek(System.*out*::println).count());

***findFirst,findAny***

List<Integer> list = Arrays.asList(1, 2, 3);

**System.*out*.println(list.stream().findFirst().get());  
System.*out*.println(list.stream().findAny().get());**

***anyMatch, noneMatch, allMatch***

* *It is used to check for condition in terminal operation*

// 5. anyMatch, allMatch, noneMatch  
List<Integer> list = Arrays.asList(1, 2, 3);

boolean b = list.stream().anyMatch(x -> x % 2 == 0);  
System.*out*.println(b);  
boolean b1 = list.stream().allMatch(x -> x > 0);  
System.*out*.println(b1);  
boolean b2 = list.stream().noneMatch(x -> x < 0);  
System.*out*.println(b2);

***toArray***

Object[] array = Stream.of(1, 2, 3).toArray();

***Min and max***

* *It is used have comparator related method inside it*

System.out.println("max: " + Stream.of(2, 44, 69).max((o1, o2) -> o2 - o1));  
System.out.println("min: " + Stream.of(2, 44, 69).min(Comparator.naturalOrder()));

***forEachOrdered***

* *For parallel stream forEach might give random values for strem but forEachOrder gives ordered interators*

List<Integer> numbers0 = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);  
System.out.println("Using forEach with parallel stream:");  
numbers0.parallelStream().forEach(System.out::println);  
System.out.println("Using forEachOrdered with parallel stream:");  
numbers0.parallelStream().forEachOrdered(System.out::println);