1. ****Given a list of integers, find out all the numbers starting with 1 using Stream functions?****

list.stream().map(String::*valueOf*).filter(x->x.startsWith("1")).forEach(System.*out*::println);

1. ****How to find duplicate elements in a given integers list in java using Stream functions?****

HashSet<Integer> hs= new HashSet<>();

list.stream().filter(x->!hs.add(x)).forEach(System.out::println);

1. ****Given the list of integers, find the first element of the list using Stream functions?****

list.stream().findFirst().ifPresentOrElse((value)->{  
 System.*out*.println(value);  
},()->{System.*out*.println("No value present");});

list.stream().findFirst().ifPresent(System.*out*::println);

1. ****Given a list of integers, find the total number of elements present in the list using Stream functions?****

list.stream().count();

1. ****Given a list of integers, find the maximum value element present in it using Stream functions?****

Integer max = list.stream().max(Integer::*compare*).get();

Integer min = list.stream().min(Integer::*compare*).get();

1. ****Given a String, find the first non-repeated character in it using Stream functions?****

String input = "sachitanand";  
  
Character result = input.chars() // Stream of String  
 .mapToObj(s -> Character.*toLowerCase*((char) s)) // First convert to Character object and then to lowercase  
 .collect(Collectors.*groupingBy*(Function.*identity*(), LinkedHashMap::new, Collectors.*counting*())) //Store the chars in map with count  
 .entrySet()  
 .stream()  
 .filter(entry -> entry.getValue() == 1L)  
 .map(entry -> entry.getKey())  
 .findFirst()  
 .get();

1. ****Given a String, find the first repeated character in it using Stream functions?****

String input = "sachitanand";  
  
Character result = input.chars() // Stream of String  
 .mapToObj(s -> Character.*toLowerCase*((char) s)) // First convert to Character object and then to lowercase  
 .collect(Collectors.*groupingBy*(Function.*identity*(), LinkedHashMap::new, Collectors.*counting*())) //Store the chars in map with count  
 .entrySet()  
 .stream()  
 .filter(entry -> entry.getValue() > 1L)  
 .map(entry -> entry.getKey())  
 .findFirst()  
 .get();

1. ****Sort a list using stream()****

list.stream().sorted(Comparator.*naturalOrder*()).toList().forEach(System.*out*::println);

list.stream().sorted().forEach(System.out::println);

list.stream().sorted(Comparator.reverseOrder()).toList().forEach(System.out::println);

1. ****Convert an integer array to a List****

Integer[] arr={1,3,4,6,7};  
Arrays.*stream*(arr).toList().forEach(System.*out*::println);

1. ****Get the local date and time using java8****

System.*out*.println(java.time.LocalDate.*now*());  
System.*out*.println(java.time.LocalDateTime.*now*());

1. ****How to convert a List of objects into a Map by considering duplicated keys and store them in sorted order?****

Map<String, Integer> notesRecords = noteLst.stream()  
 .sorted(Comparator  
 .*comparingInt*(Notes::getTagId)  
 .reversed()) // sorting is based on TagId 55,44,33,22,11  
 .collect(Collectors.*toMap* (Notes::getTagName, Notes::getTagId,  
 (oldValue, newValue) -> oldValue,LinkedHashMap::new));

1. ****How to count each element/word from the String ArrayList in Java8?****

Map<String,Long> map=list.stream().collect(Collectors.*groupingBy*(Function.*identity*(),Collectors.*counting*()));  
System.*out*.println(map);

1. ****How to find only duplicate elements with its count from the String ArrayList in Java8?****

Map<String, Long> collect = list.stream().filter(x->Collections.*frequency*(list,x)>1)  
 .collect(Collectors.*groupingBy*(Function.*identity*(), Collectors.*counting*()));

1. ****How to check if list is empty in Java 8 using Optional, if not null iterate through the list and print the object?****

Optional.*ofNullable*(list).orElseGet(ArrayList::new).stream().filter(Objects::*nonNull*).map(String::toUpperCase).forEach(System.*out*::println);

1. Write a program to print the count of each character in a String?

String str="sachitanand";  
LinkedHashMap<String, Long> collect = Arrays.*stream*(str.split("")).collect(Collectors.*groupingBy*(Function.*identity*(), LinkedHashMap::new, Collectors.*counting*()));  
System.*out*.println(collect);

1. Frequency of each element in a list

List<String> list= Arrays.*asList*("sachit","bunty","sachit","raja");  
  
Map<String, Long> collect = list.stream().collect(Collectors.*groupingBy*(Function.*identity*(), Collectors.*counting*()));  
System.*out*.println(collect);

1. Sum of digits of a number

Integer sumOfDigits = Stream.of(String.valueOf(i).split("")).collect(Collectors.summingInt(Integer::parseInt));

Integer num=236454;  
Integer sumOfNum=String.*valueOf*(num)  
 .chars()  
 .map(Character::*getNumericValue*)  
 .sum();

1. Find the second largest number in an array

Integer integer = list.stream().sorted(Comparator.*reverseOrder*()).skip(1).findFirst().get();  
System.*out*.println(integer);

1. Common elements between two arrays

list.stream().filter(list2::contains).forEach(System.*out*::println);

1. Sum of all the integers in a list

int sum = list.stream().mapToInt(Integer::intValue).sum();  
System.*out*.println(sum);

1. Reverse an integer array

int[] array = new int[] {5, 1, 7, 3, 9, 6};  
  
int[] reversedArray = IntStream.*rangeClosed*(1, array.length).map(i -> array[array.length - i]).toArray();  
  
System.*out*.println(Arrays.*toString*(reversedArray));

1. Reverse all words in a string

String str= "hello! I am sachit. How are you?";  
  
String rev = Arrays.*stream*(str.split(" ")).map(word -> new StringBuilder(word).reverse()).collect(Collectors.*joining*(" "));  
System.*out*.println(rev);

1. Sort employee by name and salary

List<Employee> sortedEmployees = employees.stream()  
 .sorted(Comparator.*comparing*(Employee::getName)

.thenComparing(Employee::getSalary))  
 .collect(Collectors.*toList*());

1. Print duplicate characters in a string

String str="Hello Sachit how are you?".replaceAll("\\s+","").toLowerCase();  
Set<String> set= new HashSet<>();  
  
Set<String> collect = Arrays.*stream*(str.split("")).filter(x -> !set.add(x)).collect(Collectors.*toSet*());  
System.*out*.println(collect);

1. Find the first repeated character in a string

Map<String, Long> charCountMap =  
 Arrays.*stream*(inputString.split(""))  
 .collect(Collectors.*groupingBy*(Function.*identity*(), LinkedHashMap::new, Collectors.*counting*()));  
  
String firstRepeatedChar = charCountMap.entrySet()  
 .stream()  
 .filter(entry -> entry.getValue() > 1)  
 .map(entry -> entry.getKey())  
 .findFirst()  
 .get();

1. Print the last element of a list

Integer lastElement = list.stream().skip(list.size() - 1).findFirst().get();  
System.*out*.println(lastElement);

1. Print 10 random numbers

Random random= new Random();  
random.ints().limit(10).forEach(System.*out*::println);

1. Print first 10 even numbers

IntStream.rangeClosed(1, 10).map(i -> i \* 2).forEach(System.out::println);

1. Sum of first 10 natural number

**int** sum = IntStream.range(1, 11).sum();

1. Merge two unsorted array into a single sorted array

**int**[] c = IntStream.concat(Arrays.stream(a), Arrays.stream(b)).sorted().toArray();

1. Join the names with a delimiter , and assigns it to a string

List<String> names = Arrays.*asList*("Alice", "Bob", "Charlie", "David");   
String joinedNames = names.stream().collect(Collectors.*joining*(", "));

1. ****PremGen:**** MetaData information of classes was stored in PremGen (Permanent-Generation) memory type before Java 8. PremGen is fixed in size and cannot be dynamically resized. It was a contiguous Java Heap Memory

.

1. ****MetaSpace:**** Java 8 stores the MetaData of classes in native memory called 'MetaSpace'. It is not a contiguous Heap Memory and hence can be grown dynamically which helps to overcome the size constraints. This improves the garbage collection, auto-tuning, and de-allocation of metadata.
2. Functional Interfaces are an interface with only one abstract method. Due to which it is also known as the Single Abstract Method (**SAM**) interface.
3. A functional interface cannot extend another interface with abstract methods but It can extend other interfaces which do not have any abstract method and only have the default, static, another class is overridden, and normal methods.
4. Java Pre-defined functional interfaces ****Runnable, Comparator, Comparable.****
5. **Pre-defined functional interfaces: Predicate, Consumer, Supplier, Operator.**
6. With the older versions of ****Java****, java.util.The date was mutable, no thread safety.

|  |  |
| --- | --- |
| ****Intermediate Operation**** | ****Terminal Operation**** |
| Used for the transition to a new state | Used to end the process under execution |
| Lazy execution of code, i.e., code is not executed as soon as it is encountered | Not lazy; code is immediately executed upon encounter |