**List Operations and Manipulations:**

**1.List of decimals in reverse order:**

***import java.util.Arrays; import java.util.Comparator; import java.util.List;***

**List<Double> decimalList = Arrays.asList(12.45, 23.58, 17.13, 42.89, 33.78, 71.85);**

**decimalList.stream().sorted(Comparator.reverseOrder()).forEach(System.out::println);**

**2.Remove duplicate elements from a list:**

***import java.util.Arrays; import java.util.List; import java.util.stream.Collectors;***

**List<String> listOfStrings = Arrays.asList("Java", "Python", "C#", "Java", "Kotlin", "Python");**

**List<String> uniqueStrngs = listOfStrings.stream().distinct().collect(Collectors.toList());**

**System.out.println(uniqueStrngs);**

**3.Find frequency of each element in a list:**

***import java.util.Arrays; import java.util.List; import java.util.Map; import java.util.function.Function; import java.util.stream.Collectors;***

**List<String> stationeryList = Arrays.asList("Pen", "Eraser", "Note Book", "Pen", "Pencil", "Stapler", "Note Book", "Pencil");**

**Map<String, Long> stationeryCountMap = stationeryList.stream().collect(Collectors.groupingBy(Function.identity(), Collectors.counting()));**

**System.out.println(stationeryCountMap);**

**4.Find the longest string in a List**

***import java.util.Arrays;import java.util.Comparator;import java.util.List;***

**List<String> list = Arrays.asList("Java", "Python", "C#", "JavaScript", "Ruby");**

**String longest = list.stream().max(Comparator.comparingInt(String::length)).orElse("");**

**System.out.println("Longest string: " + longest);**

**5.Sort a list of strings by increasing order of their length:**

***import java.util.Arrays; import java.util.Comparator; import java.util.List;***

**List<String> listOfStrings = Arrays.asList("Java", "Python", "C#", "HTML", "Kotlin", "C++", "COBOL", "C");**

**listOfStrings.stream().sorted(Comparator.comparing(String::length)).forEach(System.out::println);**

**6.Find three maximum and three minimum numbers in a list:**

***import java.util.Arrays; import java.util.Comparator; import java.util.List;***

**List<Integer> listOfIntegers = Arrays.asList(45, 12, 56, 15, 24, 75, 31, 89);**

**listOfIntegers.stream().sorted().limit(3).forEach(System.out::println);**

**listOfIntegers.stream().sorted(Comparator.reverseOrder()).limit(3).forEach(System.out::println);**

**7. Find the sum / Average of all elements in a List:**

**import java.util.Arrays;import java.util.List;**

**List<Integer> list = Arrays.asList(1, 2, 3, 4, 5);**

**int sum = list.stream().mapToInt(Integer::intValue).sum(); .average().orElse(0.0);**

**System.out.println("Sum of elements: " + sum);**

**8. Find the second largest element in a list**

***import java.util.Arrays; import java.util.List; import java.util.Comparator;***

**List<Integer> list = Arrays.asList(45, 12, 56, 75, 31, 89, 24);**

**Integer secondLargest = list.stream().sorted(Comparator.reverseOrder()).skip(1).findFirst().orElse(null);**

**System.out.println("Second Largest: " + secondLargest);**

**9. Remove all elements from a list that are greater than a given number**

**import java.util.Arrays; import java.util.List; import java.util.stream.Collectors;**

**List<Integer> list = Arrays.asList(45, 12, 56, 75, 31, 89, 24);**

**int threshold = 50;**

**List<Integer> filteredList = list.stream().filter(x -> x <= threshold).collect(Collectors.toList());**

**System.out.println(filteredList);**

**10. Check if a list contains a specific element**

***import java.util.Arrays; import java.util.List;***

**List<String> list = Arrays.asList("Java", "Python", "C#", "HTML", "Kotlin");**

**boolean contains = list.contains("Python");**

**System.out.println("List contains 'Python': " + contains);**

**11. Convert a list of strings to uppercase**

***import java.util.Arrays; import java.util.List; import java.util.stream.Collectors;***

**List<String> list = Arrays.asList("java", "python", "c#", "html", "kotlin");**

**List<String> uppercaseList = list.stream().map(String::toUpperCase).collect(Collectors.toList());**

**System.out.println(uppercaseList);**

**12. Find common elements between two lists**

***import java.util.Arrays; import java.util.List; import java.util.stream.Collectors;***

**List<Integer> list1 = Arrays.asList(1, 2, 3, 4, 5);**

**List<Integer> list2 = Arrays.asList(4, 5, 6, 7, 8);**

**List<Integer> commonElements = list1.stream().filter(list2::contains).collect(Collectors.toList());**

**System.out.println("Common Elements: " + commonElements);**

**13. Find the largest element in a list**

***import java.util.Arrays;import java.util.List;***

**List<Integer> list = Arrays.asList(45, 12, 56, 75, 31, 89, 24);**

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

**System.out.println("Largest Element: " + max);**

**14. Merge two lists into a single list**

***import java.util.Arrays; import java.util.List; import java.util.stream.Collectors;***

**List<Integer> list1 = Arrays.asList(1, 2, 3, 4);**

**List<Integer> list2 = Arrays.asList(5, 6, 7, 8);**

**List<Integer> mergedList = Stream.concat(list1.stream(), list2.stream()).collect(Collectors.toList());**

**System.out.println("Merged List: " + mergedList);**

**15. Convert a List to a Map**

***import java.util.\*; import java.util.stream.Collectors;***

**List<String> list = Arrays.asList("Java", "Python", "C#", "Java", "C++");**

**Map<String, Integer> map = list.stream().collect(Collectors.toMap(str -> str, str -> 1, Integer::sum));**

**System.out.println(map);**

**16. Find the intersection of two sets**

***import java.util.HashSet;import java.util.Set;***

**Set<Integer> set1 = new HashSet<>(Arrays.asList(1, 2, 3, 4, 5));**

**Set<Integer> set2 = new HashSet<>(Arrays.asList(4, 5, 6, 7, 8));**

**set1.retainAll(set2);**

**System.out.println("Intersection of sets: " + set1);**

**17. Remove elements from a Set based on a condition**

***import java.util.HashSet; import java.util.Set;***

**Set<Integer> set = new HashSet<>(Arrays.asList(1, 2, 3, 4, 5, 6));**

**set.removeIf(n -> n % 2 == 0); // Remove even numbers**

**System.out.println("Updated Set: " + set);**

**18. Convert a List to a Set (removing duplicates)**

***import java.util.Arrays;import java.util.List; import java.util.Set;import java.util.HashSet;***

**List<String> list = Arrays.asList("Java", "Python", "Java", "C#", "Python");**

**Set<String> set = new HashSet<>(list);**

**System.out.println("Set (duplicates removed): " + set);**

**19. Convert List to String (using joining)**

***import java.util.\*; import java.util.stream.Collectors;***

**List<String> list = Arrays.asList("Java", "Python", "C#", "Kotlin");**

**String result = list.stream().collect(Collectors.joining(", "));**

**System.out.println("List as String: " + result);**

**20. Find the smallest element in a List**

***import java.util.Arrays; import java.util.List;***

**List<Integer> list = Arrays.asList(45, 12, 56, 75, 31, 89, 24);**

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

**System.out.println("Smallest Element: " + min);**

**21. Shuffle a List : *import java.util.\*;***

**List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6);**

**Collections.shuffle(list);**

**System.out.println("Shuffled List: " + list);**

**22.Find the maximum /Min element in a List (Using Collections)**

***import java.util.Arrays; import java.util.Collections; import java.util.List;***

**List<Integer> list = Arrays.asList(1, 2, 3, 4, 5);**

**Integer max = Collections.max(list); 🡪 min(list)**

**System.out.println("Maximum Element: " + max);**

**String Operations:**

**1.Find frequency of each character in a string:**

***import java.util.Map; import java.util.function.Function; import java.util.stream.Collectors;***

**String inputString = "Java Concept Of The Day";**

**Map<Character, Long> charCountMap = inputString.chars().mapToObj(c -> (char) c).collect(Collectors.groupingBy(Function.identity(), Collectors.counting()));**

**System.out.println(charCountMap);**

**2.Check if two strings are anagrams:**

***import java.util.stream.Collectors; import java.util.stream.Stream;***

**String s1 = "RaceCar"; String s2 = "CarRace";**

**s1 = Stream.of(s1.split("")).map(String::toUpperCase).sorted().collect(Collectors.joining());**

**s2 = Stream.of(s2.split("")).map(String::toUpperCase).sorted().collect(Collectors.joining());**

**if (s1.equals(s2)) { System.out.println("Two strings are anagrams"); }**

**else { System.out.println("Two strings are not anagrams"); }**

**3.Join strings with ‘[‘ as prefix, ‘]’ as suffix and ‘,’ as delimiter:**

***import java.util.Arrays; import java.util.List; import java.util.stream.Collectors;***

**List<String> listOfStrings = Arrays.asList("Facebook", "Twitter", "YouTube", "WhatsApp", "LinkedIn");**

**String joinedString = listOfStrings.stream().collect(Collectors.joining(", ", "[", "]"));**

**System.out.println(joinedString);**

**4.Reverse each word of a string: *java.util.Arrays; import java.util.stream.Collectors;***

**String str = "Java Concept Of The Day";**

**String reversedStr = Arrays.stream(str.split(" ")).map(word -> new StringBuffer(word).reverse()).collect(Collectors.joining(" "));**

**System.out.println(reversedStr);**

**5.Find strings that start with a number: *import java.util.Arrays; import java.util.List;***

**List<String> listOfStrings = Arrays.asList("One", "2wo", "3hree", "Four", "5ive", "Six");**

**listOfStrings.stream().filter(str -> Character.isDigit(str.charAt(0))).forEach(System.out::println);**

**6.Print duplicate characters in a string:**

***import java.util.Arrays; import java.util.HashSet; import java.util.Set; import java.util.stream.Collectors;***

**String inputString = "Java Concept Of The Day".replaceAll("\\s+", "").toLowerCase();**

**Set<String> uniqueChars = new HashSet<>();**

**Set<String> duplicateChars = Arrays.stream(inputString.split("")).filter(ch -> ! uniqueChars.add(ch)).collect(Collectors.toSet()); System.out.println(duplicateChars);**

**7.Find the first repeated character in a string:**

***import java.util.Arrays; import java.util.LinkedHashMap; import java.util.Map; import java.util.function.Function; import java.util.stream.Collectors;***

**String inputString = "Java Concept Of The Day".replaceAll("\\s+", "").toLowerCase();**

**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();**

**System.out.println(firstRepeatedChar); 🡪 firstNonRepeatedChar : entry.getValue() == 1).**

**8. Check if a string is a palindrome: import java.util.StringJoiner; String str = "madam";**

**boolean isPalindrome = str.equals(new StringBuilder(str).reverse().toString());**

**System.out.println("Is palindrome: " + isPalindrome);**

**9. Count the number of vowels in a string:*import java.util.stream.\*;***

**String str = "Java Programming";**

**long countVowels = str.chars().filter(c -> "aeiouAEIOU".indexOf(c) != -1).count();**

**System.out.println("Number of vowels: " + countVowels);**

**10. Remove all white spaces from a string:*import java.util.regex.\*;***

**String str = "Java Concept Of The Day"; String result = str.replaceAll("\\s+", "");**

**System.out.println("String without spaces: " + result);**

**15. Find the length of a string without using length() method: *import java.util.stream.\*;***

***String str = "Interview";*int length = str.chars().map(c -> 1).sum();**

**System.out.println("Length of the string: " + length);**

**16. Check if a string contains only digits: *import java.util.stream.\*;***

**String str = "123456"; boolean isNumeric = str.chars().allMatch(Character::isDigit);**

**System.out.println("Is numeric: " + isNumeric);**

**17. Swap two strings without using a temporary string:**

**String str1 = "Hello", str2 = "World"; str1 = str1 + str2;**

**str2 = str1.substring(0, str1.length() - str2.length()); str1 = str1.substring(str2.length());**

**System.out.println("Swapped strings: " + str1 + ", " + str2);**

**18. Find the first non-repeated character in a string:**

***import java.util.Optional; import java.util.stream.\*;* String str = "swiss";**

**Optional<Character> firstNonRepeated = str.chars().mapToObj(c -> (char) c).filter(c -> str.indexOf(c) == str.lastIndexOf(c)).findFirst();**

**System.out.println("First non-repeated character: " + firstNonRepeated.orElse(' '));**

**19. Count the occurrences of a character in a string: *import java.util.stream.\*;***

**String str = "Java Concept Of The Day";char character = 'o';**

**long count = str.chars().filter(c -> c == character).count();**

**System.out.println("Occurrences of '" + character + "': " + count);**

**20. Reverse a string without using StringBuilder or StringBuffer: *import java.util.stream.\*;***

**String str = "Java";**

**String reversed = IntStream.rangeClosed(1, str.length()).map(i -> str.charAt(str.length() - i)).collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append).toString();**

**System.out.println("Reversed string: " + reversed);**

**21. Remove duplicate characters from a string:**

***import java.util.Arrays; import java.util.stream.Collectors;***

**String str = "Java Programming";**

**String result = Arrays.stream(str.split("")).distinct().collect(Collectors.joining());**

**System.out.println("String without duplicates: " + result);**

**22. Check if a string is a subsequence of another string: *import java.util.stream.\*;***

**String str1 = "ace", str2 = "abcde";**

**boolean isSubsequence = IntStream.range(0, str1.length()).allMatch(i -> str2.indexOf(str1.charAt(i)) >= i); System.out.println("Is subsequence: " + isSubsequence);**

**Mathematical and Numerical Operations:**

**1.Find sum of all digits of a number: *java.util.stream.Collectors; java.util.stream.Stream;***

**int i = 15623;**

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

**System.out.println(sumOfDigits);**

**2.Find sum and average of all elements in an integer array:*import java.util.Arrays;***

**int[] a = new int[] {45, 12, 56, 15, 24, 75, 31, 89};**

**int sum = Arrays.stream(a).sum(); System.out.println("Sum = "+sum);**

**double average = Arrays.stream(a).average().getAsDouble();**

**System.out.println("Average = "+average);**

**3.Find three maximum and three minimum numbers in a list:**

***import java.util.Arrays; import java.util.Comparator; import java.util.List;***

**List<Integer> listOfIntegers = Arrays.asList(45, 12, 56, 15, 24, 75, 31, 89);**

**listOfIntegers.stream().sorted().limit(3).forEach(System.out::println);**

**listOfIntegers.stream().sorted(Comparator.reverseOrder()).limit(3).forEach(System.out::println);**

**4.Find second largest number in an integer array:**

***import java.util.Arrays; import java.util.Comparator; import java.util.List;***

**List<Integer> listOfIntegers = Arrays.asList(45, 12, 56, 15, 24, 75, 31, 89);**

**Integer secondLargestNumber = listOfIntegers.stream().sorted(Comparator.reverseOrder()).skip(1).findFirst().get();**

**System.out.println(secondLargestNumber);**

**5.Print numbers which are multiples of 5: *import java.util.Arrays; import java.util.List;***

**List<Integer> listOfIntegers = Arrays.asList(45, 12, 56, 15, 24, 75, 31, 89);**

**listOfIntegers.stream().filter(i -> i % 5 == 0).forEach(System.out::println);**

**6.Generate Fibonacci series: *import java.util.stream.Stream;***

**Stream.iterate(new int[] {0, 1}, f -> new int[] {f[1], f[0]+f[1]}).limit(10).map(f -> f[0]).forEach(i -> System.out.print(i+" "));**

**7.Print the first 10 odd numbers: *import java.util.stream.Stream;***

**Stream.iterate(new int[] {1, 3}, f -> new int[] {f[1], f[1]+2}).limit(10).map(f -> f[0]).forEach(i -> System.out.print(i+" "));**

**8.Print the first 10 even numbers:*import java.util.stream.IntStream;***

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

**9.Find the sum of first 10 natural numbers:*import java.util.stream.IntStream;***

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

**10.Check if a number is prime\*\*:**

***import java.util.List; import java.util.Arrays; import java.util.stream.IntStream;***

**int number = 29;**

**boolean isPrime = IntStream.range(2, (int) Math.sqrt(number) + 1).noneMatch(i -> number % i == 0); System.out.println(number + " is prime: " + isPrime);**

**11. \*\*Generate the first N prime numbers\*\*:**

***import java.util.stream.IntStream; import java.util.List;import java.util.stream.Collectors;***

**int N = 10;**

**List<Integer> primes = IntStream.iterate(2, i -> i + 1).filter(i -> IntStream.range(2, (int) Math.sqrt(i) + 1).noneMatch(x -> i % x == 0)).limit(N).boxed().collect(Collectors.toList());**

**System.out.println("First " + N + " prime numbers: " + primes);**

**12. \*\*Find the factorial of a number\*\*:**

***import java.util.stream.IntStream;* int num = 5;**

**long factorial = IntStream.rangeClosed(1, num).reduce(1, (a, b) -> a \* b);**

**System.out.println("Factorial of " + num + " is: " + factorial);**

**13. \*\*Check if a number is an Armstrong number\*\*: *java.util.List; java.util.Arrays;***

**int number = 153; - int length = String.valueOf(number).length();**

**int sum = String.valueOf(number).chars().map(c -> (int) Math.pow(c - '0', length)).sum();**

**System.out.println(number + " is Armstrong: " + (sum == number));**

**14. \*\*Find the Greatest Common Divisor (GCD) /LCM of two numbers:**

***import java.util.List;***

**int a = 56, b = 98;**

**int gcd = (a \* b) / IntStream.rangeClosed(1, Math.min(a, b)).filter(i -> a % i == 0 && b % i == 0).max().getAsInt(); - System.out.println("GCD: " + gcd);**

**16. \*\*Reverse a number\*\*:**

**int number = 12345; int reversed = Integer.toString(number).chars().mapToObj(c -> (char) c).collect(StringBuilder::new, StringBuilder::append, StringBuilder::append).reverse().toString();**

**System.out.println("Reversed number: " + reversed);**

**17. \*\*Find the sum of digits of a number (recursive)\*\*:**

**int number = 15623; - int sum = sumOfDigits(number);**

**System.out.println("Sum of digits: " + sum);}**

**public static int sumOfDigits(int n) { return n == 0 ? 0 : n % 10 + sumOfDigits(n / 10);}}**

**18. \*\*Find the sum of the first N odd numbers\*\*: *import java.util.stream.IntStream;***

**int N = 10; - int sum = IntStream.iterate(1, i -> i + 2).limit(N).sum();**

**System.out.println("Sum of first " + N + " odd numbers: " + sum);**

**19. \*\*Find the sum of the first N even numbers\*\*: *import java.util.stream.IntStream;***

**int N = 10; - int sum = IntStream.iterate(2, i -> i + 2).limit(N).sum();**

**System.out.println("Sum of first " + N + " even numbers: " + sum);**

**20.Check if a number is a perfect number: *import java.util.stream.IntStream;* int number = 28; int sum = IntStream.range(1, number).filter(i -> number % i == 0).sum();**

**System.out.println(number + " is a perfect number: " + (sum == number));**

**22. \*\*Find the number of digits in a number\*\*:**

**int number = 15623; int count = Integer.toString(number).length();**

**System.out.println("Number of digits: " + count);**

**23. \*\*Find the sum of squares of numbers from 1 to N\*\*:*import java.util.stream.IntStream;***

**int N = 5; int sum = IntStream.rangeClosed(1, N).map(i -> i \* i).sum();**

**System.out.println("Sum of squares from 1 to " + N + ": " + sum);**

**Array and Collection Operations:**

**1.Find common elements between two arrays: *import java.util.Arrays; import java.util.List;***

**List<Integer> list1 = Arrays.asList(71, 21, 34, 89, 56, 28);**

**List<Integer> list2 = Arrays.asList(12, 56, 17, 21, 94, 34);**

**list1.stream().filter(list2::contains).forEach(System.out::println);**

**2.Extract duplicate elements from an array:**

***import java.util.Arrays; import java.util.HashSet; import java.util.List; import java.util.Set; import java.util.stream.Collectors;***

**List<Integer> listOfIntegers = Arrays.asList(111, 222, 333, 111, 555, 333, 777, 222);**

**Set<Integer> uniqueElements = new HashSet<>();**

**Set<Integer> duplicateElements = listOfIntegers.stream().filter(i -> ! uniqueElements.add(i)).collect(Collectors.toSet());**

**System.out.println(duplicateElements);**

**3.Reverse an integer array: *import java.util.Arrays; import java.util.stream.IntStream;***

**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));**

**4.Find the last element of an array: *import java.util.Arrays; import java.util.List;***

**List<String> listOfStrings = Arrays.asList("One", "Two", "Three", "Four", "Five", "Six");**

**String lastElement = listOfStrings.stream().skip(listOfStrings.size() - 1).findFirst().get();**

**System.out.println(lastElement);**

**5.Merge two unsorted arrays into a single sorted array:**

***import java.util.Arrays; import java.util.stream.IntStream;***

**int[] a = new int[] {4, 2, 7, 1}; int[] b = new int[] {8, 3, 9, 5};**

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

**System.out.println(Arrays.toString(c)); *🡪 Arrays.stream(b)).sorted().distinct().toArray() : without duplicates***

**6. Find the maximum element in an array: *import java.util.Arrays;***

**int[] array = {5, 1, 7, 3, 9, 6};**

**int maxElement = Arrays.stream(array).max().getAsInt();**

**System.out.println("Max element: " + maxElement);**

**7. Find the minimum element in an array:  *java.util.Arrays;*int[] array = {5, 1, 7, 3, 9, 6};**

**int minElement = Arrays.stream(array).min().getAsInt();**

**System.out.println("Min element: " + minElement);**

**8. Find the sum of all elements in an array: *java.util.Arrays;* int[] array = {5, 1, 7, 3, 9, 6};**

**int sum = Arrays.stream(array).sum(); System.out.println("Sum of elements: " + sum);**

**9. Find the average of elements in an array: *import java.util.Arrays;***

**int[] array = {5, 1, 7, 3, 9, 6};**

**double average = Arrays.stream(array).average().getAsDouble();**

**System.out.println("Average of elements: " + average);**

**10. Count the number of even numbers in an array:*import java.util.Arrays;***

**long evenCount = Arrays.stream(new int[]{5, 1, 7, 3, 9, 6}).filter(i -> i % 2 == 0).count();**

**System.out.println("Count of even numbers: " + evenCount);**

**11. Find the index of the first occurrence of an element in an array: *import java.util.Arrays;***

**int index = Arrays.stream(new int[]{5, 1, 7, 3, 9, 6}).boxed().toList().indexOf(7);**

**System.out.println("Index of element: " + index);**

**12. Find if an array contains a specific element: *import java.util.Arrays;***

**boolean contains = Arrays.stream(new int[]{5, 1, 7, 3, 9, 6}).anyMatch(i -> i == 7);**

**System.out.println("Contains 7: " + contains);**

**13. Check if two arrays are equal: *import java.util.Arrays;***

**boolean areEqual = Arrays.equals(new int[]{5, 1, 7, 3, 9, 6}, new int[]{5, 1, 7, 3, 9, 6});**

**System.out.println("Arrays are equal: " + areEqual);**

**14. Remove duplicates from an array: *java.util.Arrays; java.util.stream.IntStream;***

**int[] array = {5, 1, 7, 3, 9, 6, 7, 3};**

**int[] distinctArray = Arrays.stream(array).distinct().toArray();**

**System.out.println("Array without duplicates: " + Arrays.toString(distinctArray));**

**15. Rotate an array to the right by n positions: *import java.util.Arrays;***

**int[] array = {1, 2, 3, 4, 5}; - int n = 2; // Rotate by 2 positions**

**int[] rotatedArray = Arrays.copyOfRange(array, array.length - n, array.length);**

**rotatedArray = IntStream.concat(Arrays.stream(rotatedArray), Arrays.stream(array, 0, array.length - n)).toArray();**

**System.out.println("Array after rotation: " + Arrays.toString(rotatedArray));**

**16. Find the second largest element in an array: *java.util.Arrays; java.util.OptionalInt;***

**int[] array = {5, 1, 7, 3, 9, 6};**

**OptionalInt secondLargest = Arrays.stream(array).distinct().sorted().skip(array.length - 2).findFirst();**

**System.out.println("Second largest element: " + secondLargest.orElse(-1));**

**17. Find the intersection of two arrays:**

***import java.util.Arrays; import java.util.List; import java.util.stream.Collectors;***

**List<Integer> list1 = Arrays.asList(71, 21, 34, 89, 56, 28);**

**List<Integer> list2 = Arrays.asList(12, 56, 17, 21, 94, 34);**

**List<Integer> intersection = list1.stream().filter(list2::contains).collect(Collectors.toList());**

**System.out.println("Intersection: " + intersection);**

**18. Find the union of two arrays:**

***import java.util.Arrays; import java.util.List; import java.util.stream.Collectors;***

**List<Integer> list1 = Arrays.asList(71, 21, 34, 89, 56, 28);**

**List<Integer> list2 = Arrays.asList(12, 56, 17, 21, 94, 34);**

**List<Integer> union = Stream.concat(list1.stream(), list2.stream()).distinct().collect(Collectors.toList()); System.out.println("Union: " + union);**

**19. Find the difference between two arrays*: 18.imports***

**List<Integer> list1 = Arrays.asList(71, 21, 34, 89, 56, 28);**

**List<Integer> list2 = Arrays.asList(12, 56, 17, 21, 94, 34);**

**List<Integer> diff = list1.stream().filter(e -> !list2.contains(e)).collect(Collectors.toList()); System.out.println("Difference: " + diff);**

**20. Find the count of elements greater than a given value in an array: *java.util.Arrays;***

**long countGreaterThan5 = Arrays.stream(new int[]{5, 1, 7, 3, 9, 6}).filter(i -> i > 5).count();**

**System.out.println("Count of elements greater than 5: " + countGreaterThan5);**

**Date and Time Operations:**

**1.Find the age of a person in years from the birthday:**

***import java.time.LocalDate; import java.time.temporal.ChronoUnit;***

**LocalDate birthDay = LocalDate.of(1985, 01, 23); LocalDate today = LocalDate.now();**

**System.out.println(ChronoUnit.YEARS.between(birthDay, today));**

**2.Calculate the difference between two dates in days:**

***import java.time.LocalDate; import java.time.temporal.ChronoUnit;***

**LocalDate startDate = LocalDate.of(2023, 05, 15);**

**LocalDate endDate = LocalDate.of(2024, 01, 01);**

**long daysBetween = ChronoUnit.DAYS.between(startDate, endDate);**

**System.out.println("Days between: " + daysBetween);**

**3.Find the current date and time (LocalDateTime): *import java.time.LocalDateTime;***

**LocalDateTime now = LocalDateTime.now();**

**System.out.println("Current Date and Time: " + now);**

**4.Add days to the current date:**

***import java.time.LocalDate; import java.time.temporal.ChronoUnit;***

**LocalDate today = LocalDate.now();**

**LocalDate futureDate = today.plus(10, ChronoUnit.DAYS);**

**System.out.println("Date after 10 days: " + futureDate);**

**5.Subtract months from the current date: *import java.time.LocalDate;***

**LocalDate today = LocalDate.now();**

**LocalDate previousMonth = today.minusMonths(2);**

**System.out.println("Date two months ago: " + previousMonth);**

**6.Find the day of the week for a given date:**

***import java.time.LocalDate; import java.time.DayOfWeek;***

**LocalDate date = LocalDate.of(2024, 12, 31);**

**DayOfWeek dayOfWeek = date.getDayOfWeek();**

**System.out.println("Day of the week: " + dayOfWeek);**

**7.Check if a year is a leap year: *import java.time.Year;***

**int year = 2024; boolean isLeapYear = Year.isLeap(year);**

**System.out.println(year + " is a leap year: " + isLeapYear);**

**8.Get the current timestamp (DateTimeFormatter):**

***import java.time.LocalDateTime; import java.time.format.DateTimeFormatter;***

**LocalDateTime now = LocalDateTime.now();**

**DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");**

**System.out.println("Current Timestamp: " + now.format(formatter));**

**9.Find the number of days in a given month of a year:**

***import java.time.Month; import java.time.YearMonth;***

**YearMonth yearMonth = YearMonth.of(2024, Month.FEBRUARY);**

**int daysInMonth = yearMonth.lengthOfMonth();**

**System.out.println("Days in the month: " + daysInMonth);**

**10.Convert a string to a LocalDate:**

***import java.time.LocalDate; import java.time.format.DateTimeFormatter;***

**String dateString = "2024-12-29";**

**DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");**

**LocalDate date = LocalDate.parse(dateString, formatter);**

**System.out.println("Parsed Date: " + date);**

**11.Find the number of weeks /months between two dates:**

***import java.time.LocalDate;import java.time.temporal.ChronoUnit;***

**LocalDate startDate = LocalDate.of(2024, 01, 01);**

**LocalDate endDate = LocalDate.of(2024, 12, 31);**

**long weeksBetween = ChronoUnit.WEEKS.between(startDate, endDate);**

**long monthsBetween = ChronoUnit.MONTHS.between(startDate, endDate);**

**System.out.println("Weeks between: " + weeksBetween);**

**12.Check if a given date is in the future or past: *import java.time.LocalDate;***

**LocalDate today = LocalDate.now();**

**LocalDate futureDate = LocalDate.of(2025, 05, 01);**

**if (futureDate.isAfter(today)) { System.out.println("The date is in the future.");**

**} else {System.out.println("The date is in the past."); }**

**13.Get the first day of the current month: for *last day- replace***

***import java.time.LocalDate; import java.time.temporal.TemporalAdjusters;***

**LocalDate firstDayOfMonth = LocalDate.now().with(TemporalAdjusters.firstDayOfMonth());**

**System.out.println("First day of the current month: " + firstDayOfMonth);**

**14.Add years to the current date: *import java.time.LocalDate;***

**LocalDate today = LocalDate.now();**

**LocalDate futureDate = today.plusYears(5);**

**System.out.println("Date after 5 years: " + futureDate);**

**15.Subtract hours from the current time: *import java.time.LocalTime;***

**LocalTime now = LocalTime.now();long monthsBetween = ChronoUnit.MONTHS.between(startDate, endDate);**

**LocalTime newTime = now.minusHours(3);**

**System.out.println("Time after subtracting 3 hours: " + newTime);**

**16.Get the day of the year for a given date: *import java.time.LocalDate;***

**LocalDate date = LocalDate.of(2024, 12, 29);**

**int dayOfYear = date.getDayOfYear();**

**System.out.println("Day of the year: " + dayOfYear);**

**1.List Operations and Manipulations: 24 – pages: 3**

**2.String Operations: 22 - pages: 2**

**3. Mathematical and Numerical Operations: 23 - pages: 2**

**4. Array and Collection Operations: 20 - pages:2**

**5. Date and Time Operations: 16- pages: 2**

**Total - 105**