**What is streams API?**

|  |
| --- |
| 1. It is an API introduced in java 1.8 version 2. It provides a power full and concise way to process elements which are in a sequence(array/collection) |

**What is a stream?**

|  |
| --- |
| 1. Stream are not data structures themselves, but rather a way to process data from existing collections likes lists, sets and arrays etc.. 2. They represents a sequence of elements that can be processed one by one 3. Streams are lazy, meaning they don't actually process the data until a terminal operation is called. This allows efficient operations on large datasets. |

**What are the key concepts of streams API?**

|  |
| --- |
| * **Stream Creation:** You can create streams from various sources like collections (using stream() method), arrays (using Arrays.stream()), or generate them on-demand. * **Intermediate Operations:** These operations transform the elements in the stream without actually changing the original data structure. Examples include filter (to keep elements based on a condition), map (to transform each element), sorted (to sort elements). You can chain multiple intermediate operations to create complex pipelines. * **Terminal Operations:** These operations consume the stream and produce a result. Examples include forEach (to iterate and perform actions on each element), count (to count the number of elements), collect (to collect elements into a new data structure). Terminal operations mark the end of the stream processing. |

**What are the advantages of streams?**

|  |
| --- |
| * **Readability:** Stream operations are often more concise and readable than traditional for loops. * **Efficiency:** Streams are lazy-evaluated and can be parallelized for faster processing on large datasets. * **Immutability:** Streams don't modify the original data source, promoting functional programming style. |

**What is the use of Arrays class and in which package it is existed?**

|  |
| --- |
| The **Arrays** class in **java.util package** is a part of the **Java Collection Framework**. This class provides static methods to dynamically create and access **Java arrays**. It consists of only static methods and the methods of Object class. |

**Explain about asList() method of Arrays class?**

|  |
| --- |
| 1. It is having only one var-args parameter and it is generic type so we can pass any type and any number of elements to this method. 2. It returns the passed elements as a List object   **Syntax**: public static List asList(T... a)  **Special Note:** The type of array must be a Wrapper Class(Integer,Float, etc) in case of primitive data types(int, float,etc) , i.e you can’t pass int a[] but you can pass Integer a[]. If you pass int a[], this function will return a List <int a[]> and not List <Integer> , as “autoboxing” doesn’t happen in this case and int a[] is itself identified as an object and a List of int array is returned, instead of list of integers , which will give error in various Collection functions .  **AsListUage.java**  import java.util.Arrays;  import java.util.List;  class AsListUsage  {  public static void main(String[] args) {  List l1= Arrays.asList(10,20,30,40,50,60); //list with 6 elements will be created  List l2= Arrays.asList(new int[]{10,20,30,40,50,60}); //list with single element will be created  List l3= Arrays.asList(new Integer[]{10,20,30,40,50,60}); //list with 6 elements will be created  System.out.println("l1:\t"+l1+" size is "+l1.size());  System.out.println("l2:\t"+l2+" size is "+l2.size());  System.out.println("l3:\t"+l3+" size is "+l3.size());  }  }  **Output:**  l1: [10, 20, 30, 40, 50, 60] size is 6  l2: [[I@53d8d10a] size is 1  l3: [10, 20, 30, 40, 50, 60] size is 6 |

**Write a program to finding and counting long names?**

|  |
| --- |
| import java.util.Arrays;  import java.util.List;  import java.util.stream.Collectors;  class FindLongNames  {  public static void main(String[] args) {  List<String> names= Arrays.asList("madhu","shekar","giri babu","shobha rani","priya","sam","ram","sai","vinnu","prannu");  List<String> filterdNames=names.stream().filter(name->name.length()>=5).collect(Collectors.toList());  System.out.println("Actual List:\t"+names);  System.out.println("FilterdNames:\t"+filterdNames);  }  }  **Output:**  Actual List: [madhu, shekar, giri babu, shobha rani, priya, sam, ram, sai, vinnu, prannu]  FilterdNames: [madhu, shekar, giri babu, shobha rani, priya, vinnu, prannu] |

**In Summary**

the Streams API provides a powerful and efficient way to process collections of data in Java. By understanding its concepts and using its operations, you can write cleaner, more concise, and performant code for data manipulation tasks.

**What are terminal Operations in streams?**

|  |
| --- |
| Terminal Operations are the type of Operations that return the result. These Operations are not processed further just return a final result value  Some terminal operations used in streams are   1. Collect 2. forEach 3. sorted() 4. count() 5. reduce() 6. toList() 7. toSet()   etc.. |

What is Streams API?

What is a stream?

What are the key concepts of streams API?

What are the advantages of streams?

What is the use of Arrays class and in which package it is existed?

Explain about asList() method of Arrays class?

Write a program to finding and counting long names?

What are terminal Operations in streams?