Practical 6

Aim : Write a Program for finding the Product of the three largest Distinct Elements. Use a Priority Queue to efficiently find and remove the largest elements

Algorithm :

1. Initialize Priority Queue:

- Create a Priority Queue pq with a custom comparator to keep the elements in descending order.

2. Insert Elements:

- Iterate over the input array nums.
- Add each element to the priority queue using the offer method.

3. Find Three Largest Elements:

- Retrieve the three largest elements from the priority queue by using the poll method three times.
- Store them in variables largest1, largest2, and largest3.

4. Calculate Product:

- Compute the product of the three largest elements obtained.
- Multiply largest1, largest2, and largest3.
- Return the result as the product of the three largest distinct elements.

5. Main Method Execution:

- Create an array nums with integers {5, 10, 2, 8, 15, 3}.
- Call the findProductOfThreeLargest method with nums array as input.

・ Display the product of the three largest distinct elements to the console.

Example

Example Walkthrough

Given the input array **{5, 10, 2, 8, 15, 3}**:

- **Distinct Elements**: The distinct elements are **{2, 3, 5, 8, 10, 15}**.
- **Priority Queue**: After adding these elements to the priority queue, the order will be: **15, 10, 8, 5, 3, 2**.
- **Poll Operations**:
  - First **poll()** returns **15** (largest).
  - Second **poll()** returns **10** (second largest).
  - Third **poll()** returns **8** (third largest).
- **Product Calculation**: The product is **15 * 10 * 8 = 1200**.

Poll() :-

In Java, the **poll()** method is a part of the **Queue** interface, which is implemented by various classes, including **PriorityQueue**. The **poll()** method is used to retrieve and remove the head (the first element) of the queue. If the queue is empty, it returns **null**

Priority Queue()

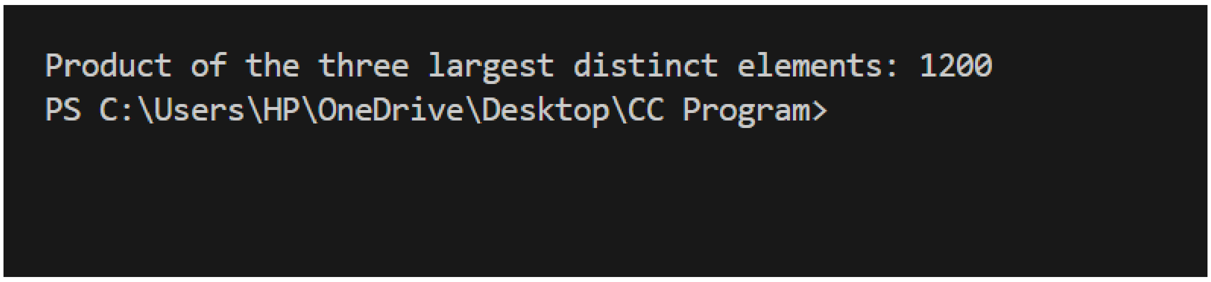
In the context of a PriorityQueue, the poll() method retrieves and removes the element that is considered the highest priority. The priority is determined by the natural ordering of the elements or by a specified comparator.

Code:-

```
import java.util.*;
public class ThreeLargestProduct {
   public static int findProductOfThreeLargest(int[] nums) {
      // Create a PriorityQueue to store elements in
descending order
      PriorityQueue<Integer> pq = new
PriorityQueue<>(Collections.reverseOrder());

      // Add elements to the PriorityQueue
      for (int num : nums) {
         pq.offer(num);
      }

      // Retrieve the three largest elements
      int largest1 = pq.poll();
      int largest2 = pq.poll();
      int largest3 = pq.poll();

      // Return the product of the three largest elements
      return largest1 * largest2 * largest3;
   }
```

```
    public static void main(String[] args) {
        // Input array
        int[] nums = {5, 10, 2, 8, 15, 3};

        // Calculate and display the product of the three largest
distinct elements
        System.out.println("Product of the three largest distinct
elements: " +
                findProductOfThreeLargest(nums));
    }
}
```

Output:-



```
Product of the three largest distinct elements: 1200
PS C:\Users\HP\OneDrive\Desktop\CC Program>
```