The Josephus Problem is a famous theoretical problem in mathematics and computer science that involves a group of people standing in a circle. Starting from a given point, every k-th person is eliminated from the group until only one person remains. The task is to determine the position of the last remaining person.

A **queue** is a great data structure to solve the Josephus Problem because it models the process of eliminating every k-th person by rotating the queue (i.e., moving people to the back until it's their turn to be eliminated).

**Problem Statement:**

* There are n people numbered 1, 2, ..., n.
* Every k-th person is eliminated in a circular fashion.
* The goal is to find the position of the last remaining person.

**Approach Using a Queue:**

1. **Enqueue all the people** into the queue (using their numbers).
2. **Rotate the queue** k-1 times, so the k-th person is at the front of the queue.
3. **Dequeue the person at the front**, and repeat the process with the remaining people.
4. Continue this process until only one person remains in the queue.

**Code Implementation:**

import java.util.LinkedList;

import java.util.Queue;

public class JosephusProblem {

// Solve Josephus problem and return the position of the last person

public static int josephus(int n, int k) {

Queue<Integer> queue = new LinkedList<>();

// Step 1: Enqueue all people (1 through n)

for (int i = 1; i <= n; i++) {

queue.add(i);

}

// Step 2: Eliminate people in the circle

while (queue.size() > 1) {

// Step 2a: Move the first k-1 people to the back of the queue

for (int i = 1; i < k; i++) {

queue.add(queue.poll()); // Remove the front and add it to the rear

}

// Step 2b: Remove the k-th person from the queue (this person is eliminated)

queue.poll(); // Eliminate the person at the front of the queue

}

// Step 3: The last remaining person is the winner

return queue.peek(); // The last remaining person

}

public static void main(String[] args) {

int n = 7; // Total number of people

int k = 3; // Eliminate every 3rd person

int result = josephus(n, k);

System.out.println("The last remaining person is at position: " + result);

}

}

**Explanation:**

1. **Queue Initialization**: We use a Queue<Integer> to simulate the circle of people. The people are represented by their numbers from 1 to n.
2. **Elimination Process**:
   * For each round, we rotate the queue k-1 times by moving people from the front to the back of the queue.
   * After rotating, the person at the front of the queue (the k-th person) is eliminated by calling poll().
3. **Repeat** until only one person is left in the queue.
4. **Result**: The last person remaining in the queue is the winner.

**Example:**

For n = 7 and k = 3, the sequence of eliminations will be:

* People: [1, 2, 3, 4, 5, 6, 7]
* Eliminate 3: [1, 2, 4, 5, 6, 7]
* Eliminate 6: [1, 2, 4, 5, 7]
* Eliminate 2: [1, 4, 5, 7]
* Eliminate 7: [1, 4, 5]
* Eliminate 5: [1, 4]
* Eliminate 4: [1]
* The last remaining person is 1.

**Output for the above example:**

The last remaining person is at position: 4

**Time Complexity:**

* Each elimination involves rotating the queue and removing an element, which takes O(k) time for each round.
* The total time complexity is O(n \* k), where n is the number of people and k is the step size (every k-th person is eliminated).

**Space Complexity:**

* The space complexity is O(n), as we are storing n people in the queue.