import java.util.LinkedList;  
 import java.util.Queue;  
  
public class RoundRobinScheduling {  
 public static void simulateRoundRobinScheduling(int[] processes, int timeQuantum) {  
 // Create a queue to store the processes  
 Queue<Integer> queue = new LinkedList<>();  
  
 // Add processes to the queue  
 for (int process : processes) {  
 queue.add(process);  
 }  
  
 // Simulate round-robin scheduling  
 while (!queue.isEmpty()) {  
 // Retrieve the current process from the front of the queue  
 int currentProcess = queue.remove();  
  
 // Process the current process for the given time quantum  
 System.out.println("Processing process " + currentProcess + " for time quantum " + timeQuantum);  
  
 // Check if the current process still requires processing  
 if (currentProcess > timeQuantum) {  
 // Reduce the remaining time for the current process  
 currentProcess -= timeQuantum;  
  
 // Add the current process back to the end of the queue  
 queue.add(currentProcess);  
 } else {  
 // The current process has completed execution  
 System.out.println("Process " + currentProcess + " has completed execution.");  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 int[] processes = { 5, 10, 15, 20 }; // Example processes (replace with your own)  
  
 int timeQuantum = 8; // Example time quantum (replace with your own)  
  
 simulateRoundRobinScheduling(processes, timeQuantum);  
 }}