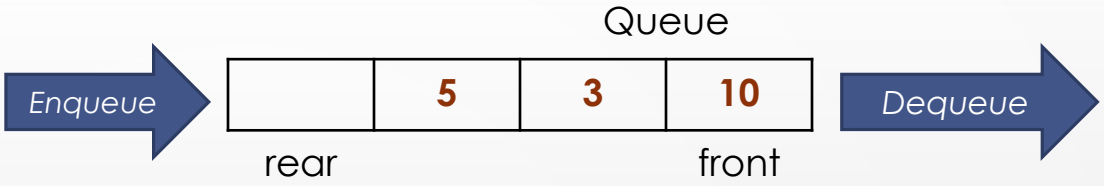
PART 2
About Queue

QUEUE



What is queue?	Queue is **a linear data structure** where the first element is inserted from one end called REAR and deleted from the other end called as FRONT Front points to the beginning of the queue and Rear points to the end of the queue.
Queue Concept	Queue follows the **FIFO: First-In First-Out** structure Element inserted first will also be removed first. Queue is open at both ends: for *enqueue* and *dequeue* operations

Queue Implementation

- 2 ways:
  1. Use *Java Collection framework* that provides a ***Queue class*** (import **java.util.Queue**).
  2. Include **Queue class** from Queue **API** (**A**pplication **P**rogramming **I**nterface)

Both classes contain methods that apply the queue concept, but the method name might be different

In this topic we will implement Queue using **Queue API**

Queue API : List of Methods

public class Queue<Item>	Description	Example code
Queue<Item>()	Create queue of Items	Queue <Integer> q1 = **new Queue<Integer>();**
void enqueue(Item item)	to add item into queue.	**for (int i=5; i>1; i--)** q1.enqueue(i);
Item dequeue()	to remove the front Item from queue.	System.***out.println("Remove an Item in q1-"+ q1.dequeue());***
boolean isEmpty()	to check if the queue is empty	**if(!q1.isEmpty())** System.***out.println("Front Item of q1:"+q1.peek());***
int size()	to get the number of item in queue	System.***out.println("Number of Item in q1-"+ q1.size());***
Item peek()	retrieve the front item without removing it	Total=10+***q1.peek();***

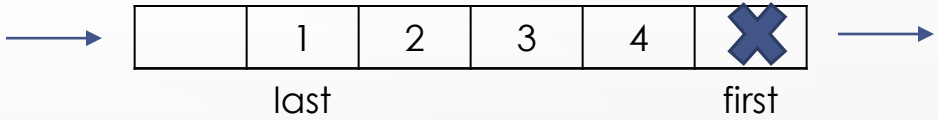
Queue API (Queue.java)

```
public class Queue<Item> {
   private Node first, last;
   int N = 0;

   private class Node {
      Item item;
      Node next;
   }
   public boolean isEmpty() {
      return first == null;
   }
   public void enqueue(Item item)
{
      Node oldlast = last;
      last = new Node();
      last.item = item;
      last.next = null;
      if (isEmpty()) first = last;
      else oldlast.next = last;
      N++;
   }
```

```
public Item dequeue() {
      Item item = first.item;
      first     = first.next;
      N--;
      if (isEmpty())
         last = null;
      return item;
   }
   public int size() {
      return N;
   }

   public Item peek() {
           if (!isEmpty())
                  return first.item;
   }
}
```

This is a simplified version. Use the generic Queue.java attached

Example: Queue program




```
import java.util.*;

public class MyQueue {

     public static void main(String args[]) {
          Queue <Integer> q1 = new Queue<Integer>();

          for (int i=5; i>0; i--)
               q1.enqueue(i);
          System.out.println("Elements of q1-"+q1);
          System.out.println("Remove Item q1-"+q1.dequeue());
          System.out.println("Number of Item in q1-"+ q1.size());

          if(!q1.isEmpty())
               System.out.println("Front Item of q1: "+q1.peek());
     }
}
```

Demonstration

Read integer numbers end with 0, push it onto queue. At the end, output all the queue's elements.