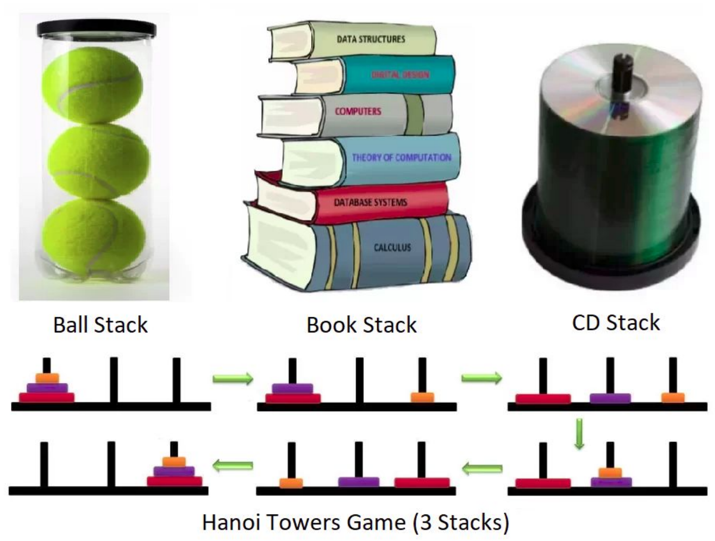
Stacks and Queues

Stacks

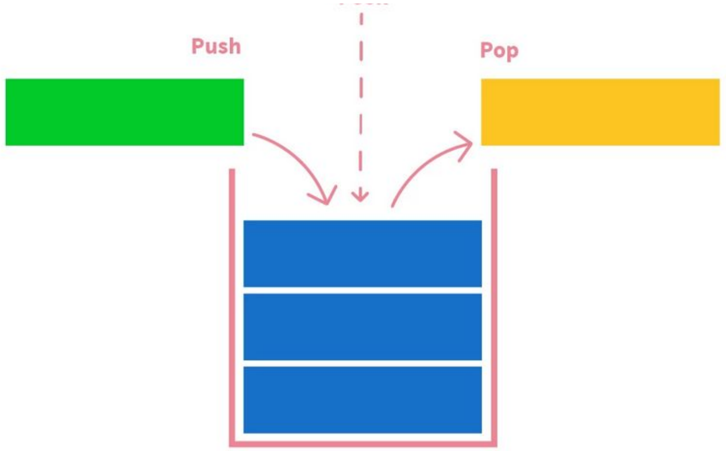
A stack is a container of objects based on the last-in-first-out (LIFO) rule.

Elements are stored in order of insertion.

We can only add/remove/examine the last element added (the "top").



Stacks: Supported Operations



push(element): Add an element to the top of stack

element pop(): Remove the top element and returns it

int size(): how many items are in the stack?

isEmpty(): false if there are 1 or more items in stack, true otherwise

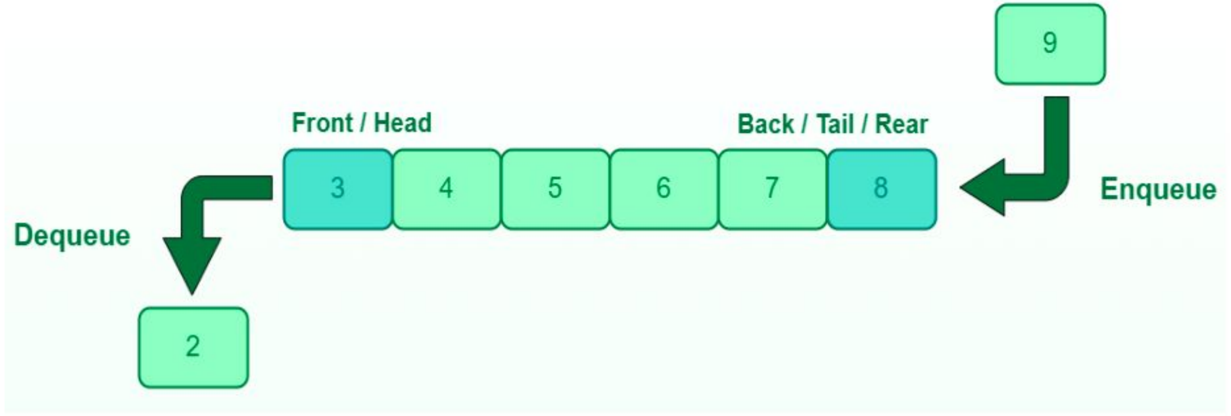
element peek(): Examine the top element without removing it

Queue

A Queue is defined as a linear data structure that is open at both ends and the operations are performed in First-In-First-Out (FIFO) order.

Rear of Queue
Front of Queue
next...
PAY HERE

Queue

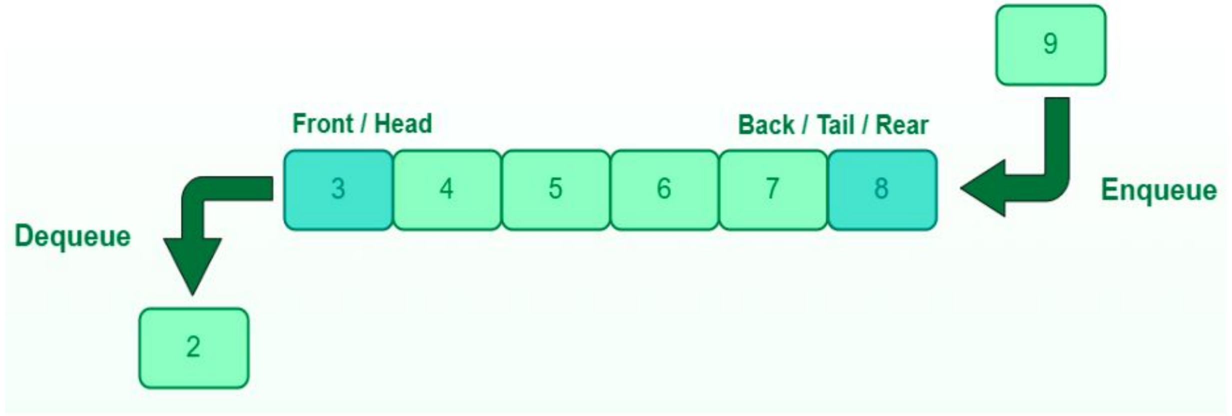


Supported Operations

- add(item): “enqueue” add an element to the back.
- remove(): “dequeue” Remove the front element and return.
- peek(): Examine the front element without removing it.
- size(): how many items are stored in the queue?
- isEmpty(): if 1 or more items in the queue returns false, true otherwise

Simple Queue

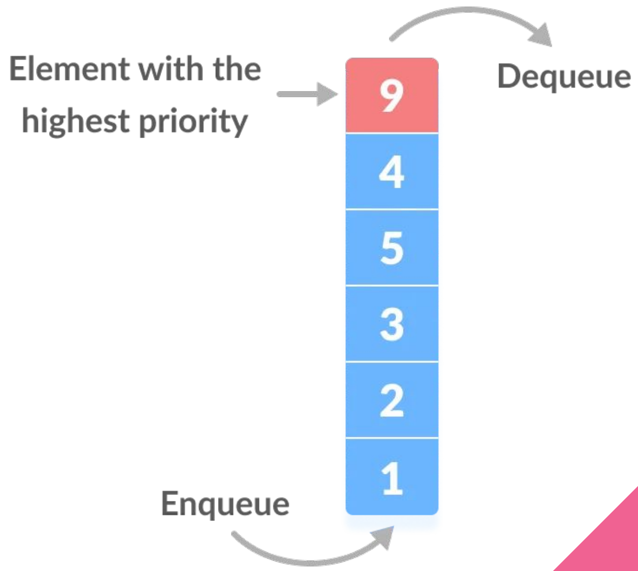
Insertion takes place at the rear and removal occurs at the front. It strictly follows the FIFO rule



Priority Queue

- Special queue where each element is associated with a priority value.
- Elements are served based on their priority (higher priority served first)
- If elements with the same priority occur, they are served according to their order in the queue.

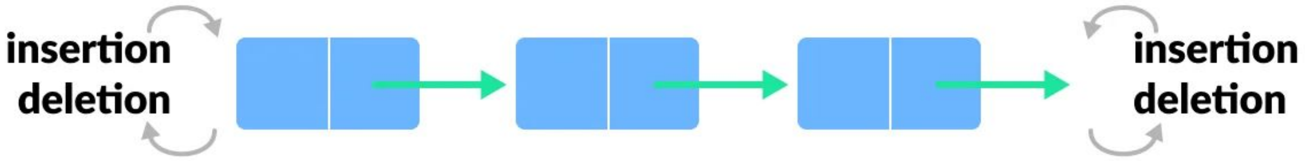
In a queue, the **first-in-first-out rule** is implemented whereas, in a priority queue, the values are removed **on the basis of priority**. The element with the highest priority is removed first.



Deque (Double Ended Queue)

Insertion and removal of elements can be performed from either from the front or rear.

It does not follow the FIFO (First In First Out) rule.



Deque Operations

- void addLast(element)
- void addFirst(element)
- element removeLast()
- element removeFirst()
- element getLast()
- element getFirst()

Implementing Stack, Queue and Deque

Which data structures could we use to implement Stack, Queue and Deque?

Let’s implement our Stack and Simple Queue

- **Write** your code here: classwork/xx_stack_queue/YOUR_FILES_HERE
- **Copy** the the files from the materials repo **classwork/stack_queue/*** to your classwork/xx_stack_queue/ folder.
- Let’s have fun **implementing** your **Stack.java and Queue.java**