Week 4 - Lab 2

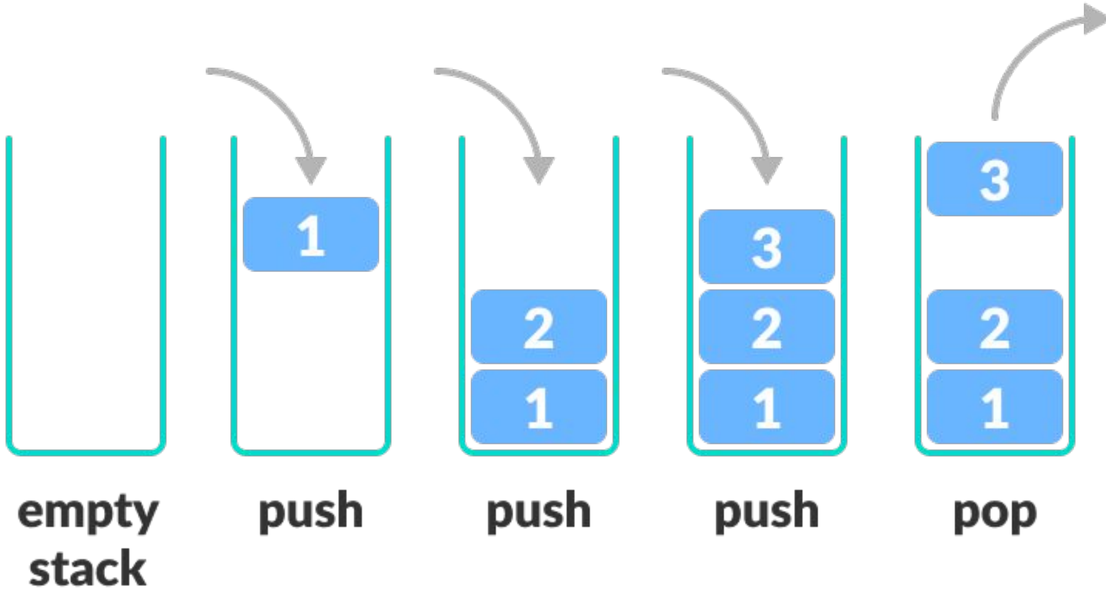
Section:103
Simge Tekin

Stack - Review

- **LIFO behaviour**

Common Operations:

- **push(E item):** Adds an element to the top.
- **pop():** Removes and returns the top element.
- **peek():** Retrieves the top element without removing it.
- **isEmpty():** Checks if the stack



Stack IS A List

1. **Stack extends Vector —Stack IS A Vector**

```java
public class Stack<E> extends Vector<E>
```

2. **Vector implements List —Vector IS A List**

```java
public class Vector<E>

extends AbstractList<E>

implements List<E>, RandomAccess, Cloneable, java.io.Serializable
```

Stack - List

1. **Stack extends Vector —Stack IS A Vector**

```
public class Stack<E> extends Vector<E>
```

2. **Vector implements List**

```
public class Vector<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, java.io.Serializable
```

This implies that Stack also implements List

Stack implements List Interface

and a variable can be declared as:

```
List<Integer> myListStack= new Stack<>();
```

What happens if you instantiate a stack object as a List?

Stack implements List Interface

and can be declared as:

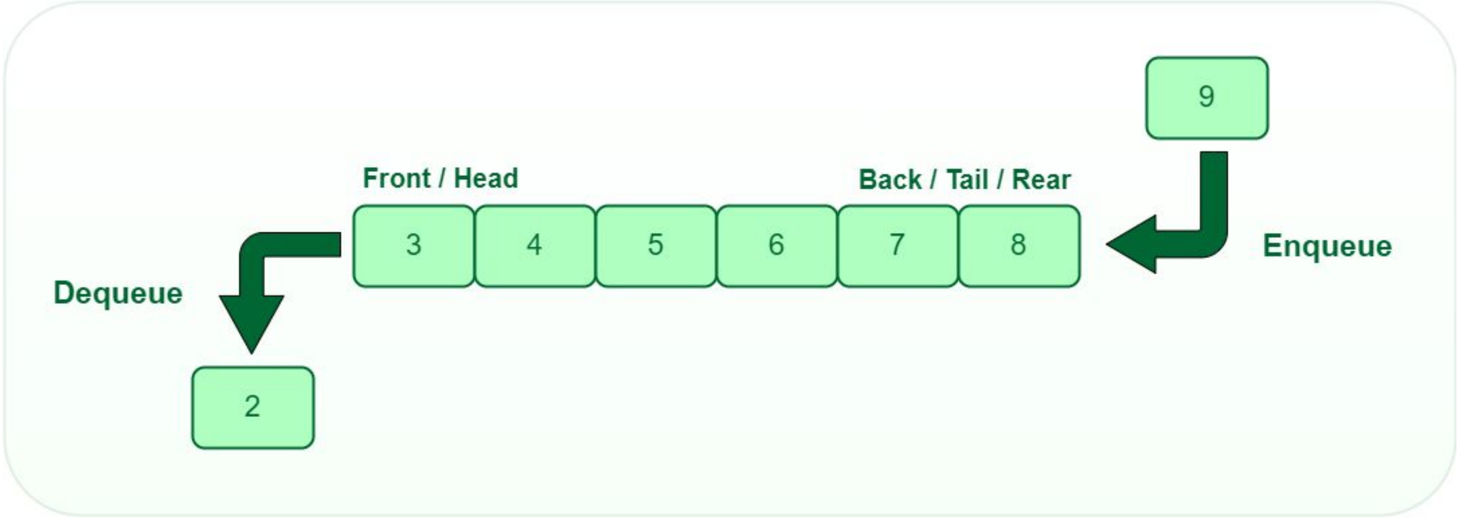
```
List<Integer> myListStack= new Stack<>();
```

What happens if you instantiate a stack object as a List?

Hint: When you declare a class using an interface type, you can only use the methods defined in the interface, not the methods specific to the class.

Queue - Review

- Queue is an interface in java.util
  - `public interface Queue<E> extends Collection<E>`
- FIFO behaviour



Queue - Review

- Queue is an interface in java.util
  - `public interface Queue<E> extends Collection<E>`
- FIFO behaviour
- Key Methods:

add(e) / offer(e): Add element to the queue.

remove() / poll(): Remove and return the head of the queue.

element() / peek(): Retrieve the head without removing it.

Queue - Review

Since Queue is an interface, you cannot instantiate a Queue object as :

```
new Queue<>();
```

You need to use a **concrete class** that **implements** Queue.

Common implementations:

- LinkedList (implements both List and Queue).
- PriorityQueue (elements ordered by natural order or a comparator).

Queue Example

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

queue.add(10);

queue.add(20);

System.out.println(queue.poll());
```

Problem: Reverse a Queue Using Recursion

Description:

Given a queue, write a recursive function to reverse it. You must achieve this using recursion.

Example:

Input:
```
Queue: [1, 2, 3, 4, 5]
```
Output:
```
Queue: [5, 4, 3, 2, 1]
```

Solution Approach:

1. **Base Case:** If the queue is empty, return.
2. **Recursive Case:**
    - Dequeue the front element.
    - Recursively reverse the remaining queue.
    - Enqueue the removed element back.