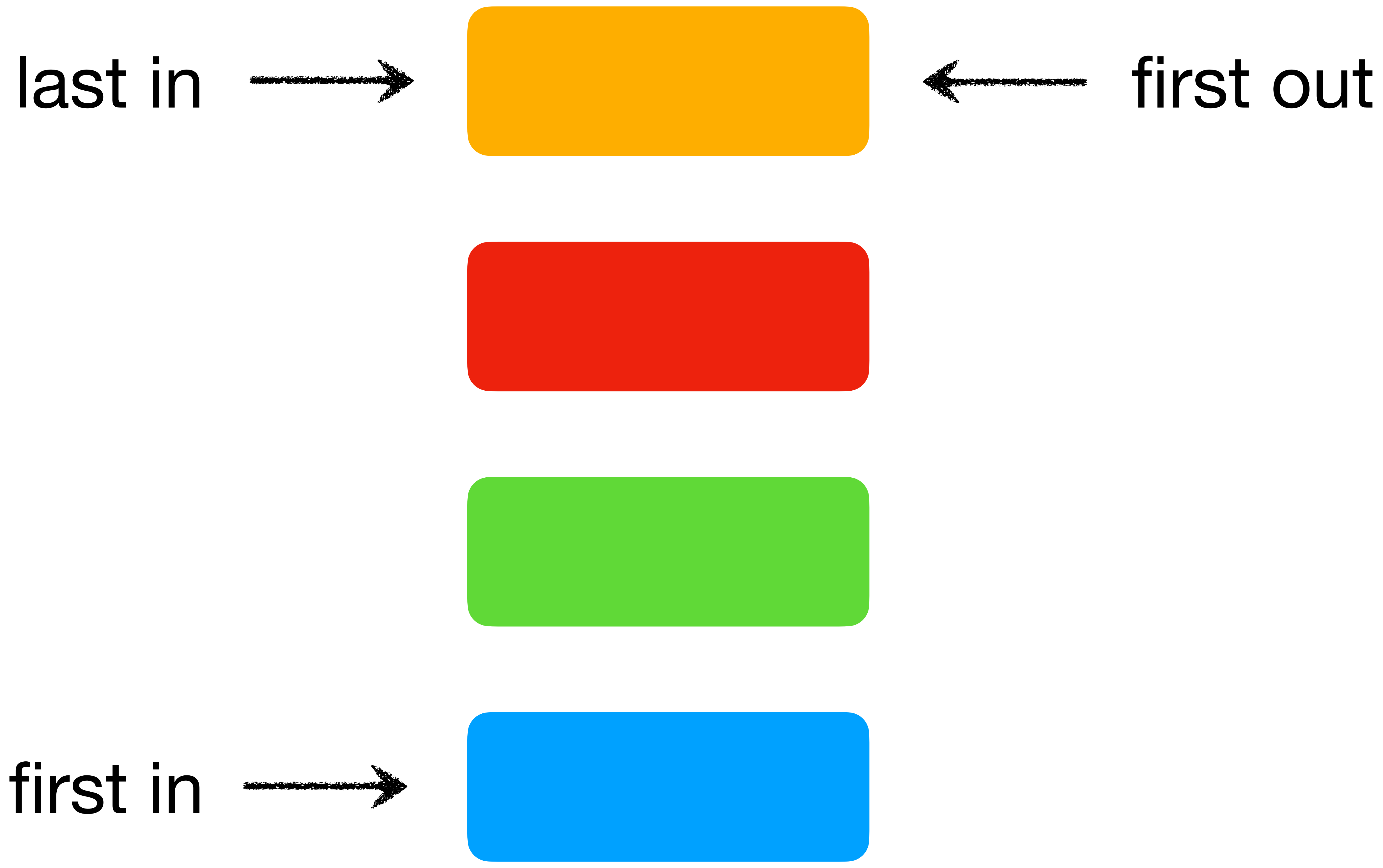
# Collections

**Deque Interface**

# Deque Interface Used as a Stack

- implemented by `LinkedList` and `ArrayQueue`
- adds element in the front, reads from the back
    - LIFO: **L**ast **I**n, **F**irst **O**ut
- proper methods
    - `peek()`, `push(E e)`, `poll()`
- methods inherited from Collection
    - `element()`, `add(E e)`, `remove()`

## What Is a Stack?



```
Deque<String> colors = new ArrayDeque<>();
colors.push("blue");
colors.push("green");
colors.push("red");
colors.push("yellow");
System.out.println(colors);
System.out.println(colors.peek());
colors.pop();
System.out.println(colors.peek());
colors.pop();
colors.pop();
colors.pop();
System.out.println(colors.peek());
```

```
[yellow, red, green, blue]
yellow
red
null
```

## Deque Interface as Double-Ended Queue

- proper methods
  - `peekFirst()`, `offerFirst(E e)`, `pollFirst()`
  - `peekLast()`, `offerLast(E e)`, `pollLast()`
- methods inherited from Collection
  - `getFirst()`, `addFirst(E e)`, `removeFirst()`
  - `getLast()`, `addLast(E e)`, `removeLast()`

```java
Deque<Integer> nums = new LinkedList<>();
nums.addFirst(9);
nums.offerFirst(-11);
nums.addLast(5);
System.out.println(nums);
System.out.println(nums.getFirst());
System.out.println(nums.peekLast());
nums.pollFirst();
System.out.println(nums);
System.out.println(nums.getFirst());
System.out.println(nums.peekLast());
```

```
[-11, 9, 5]
-11
5
[9, 5]
9
5
```