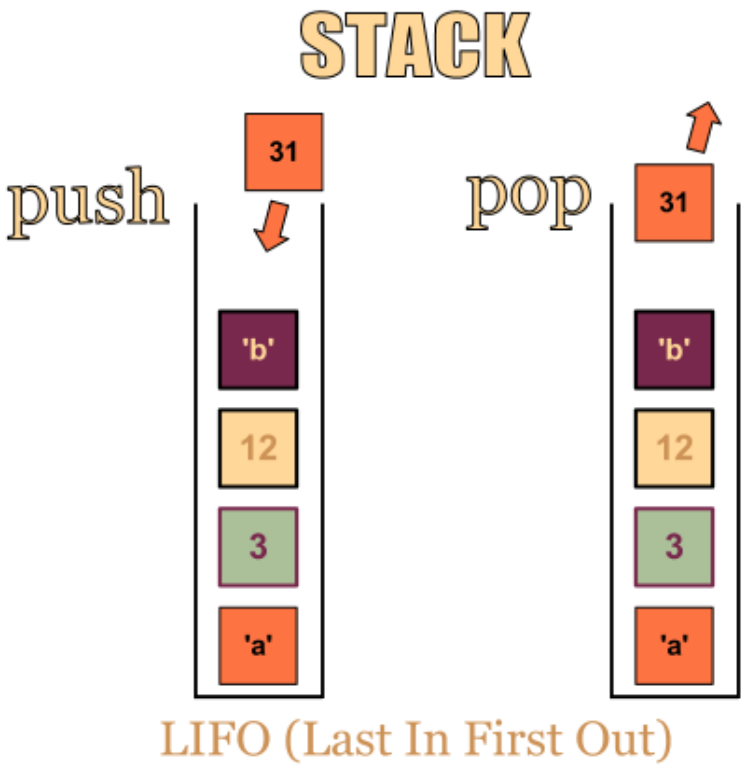
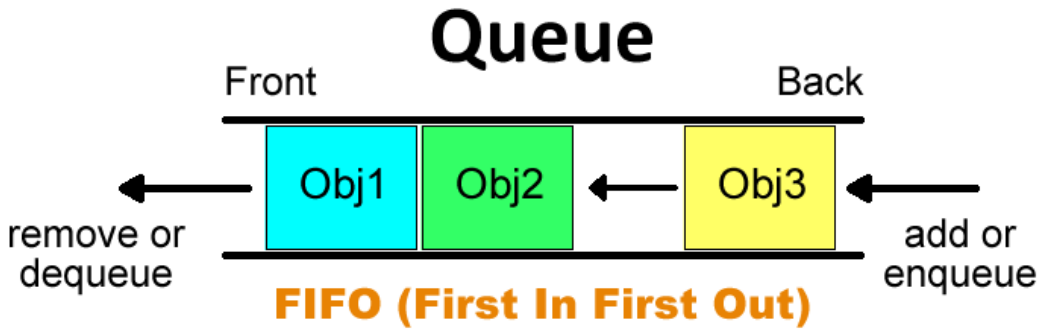
Stacks and Queues

David Greenstein
Monta Vista

Stack vs Queue

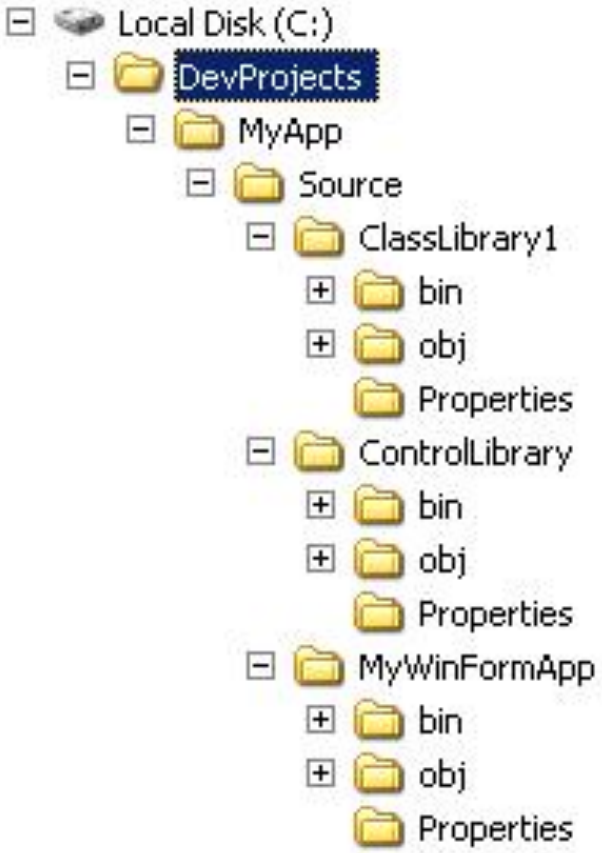




Stacks and queues are used for temporary storage, but in different situations

Stacks are Used for

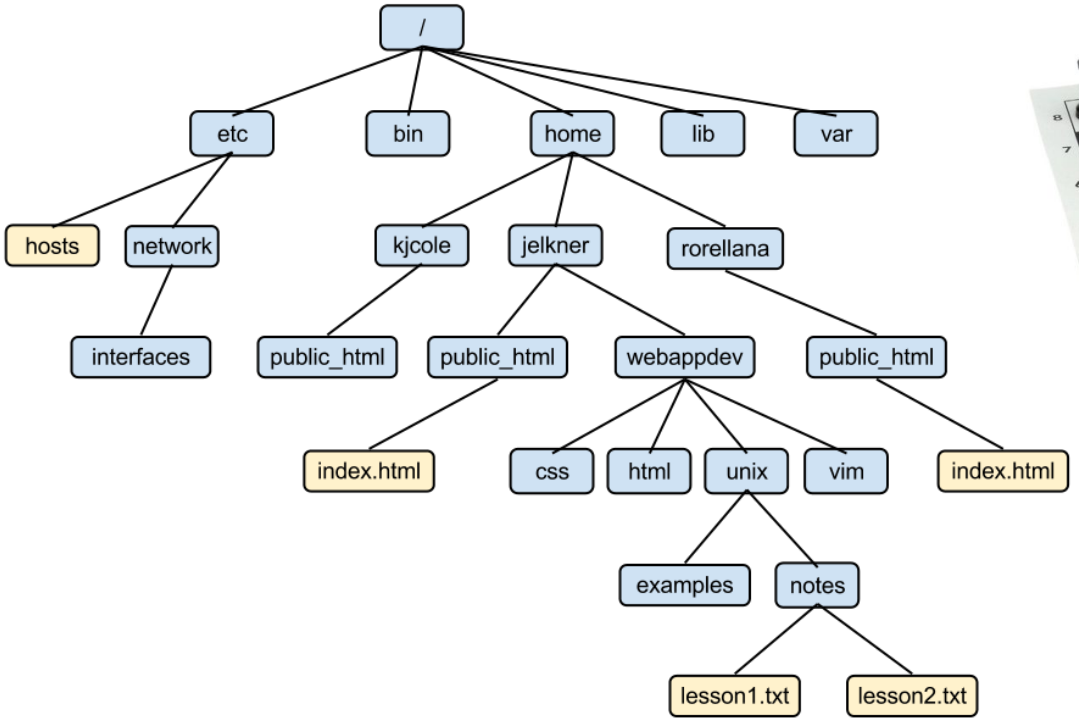
- handling nested structures:
  - processing directories within directories
  - evaluating expressions within expressions



$$c^2 = a^2 + b^2 - 2ab\cos\left(\angle C\right)$$

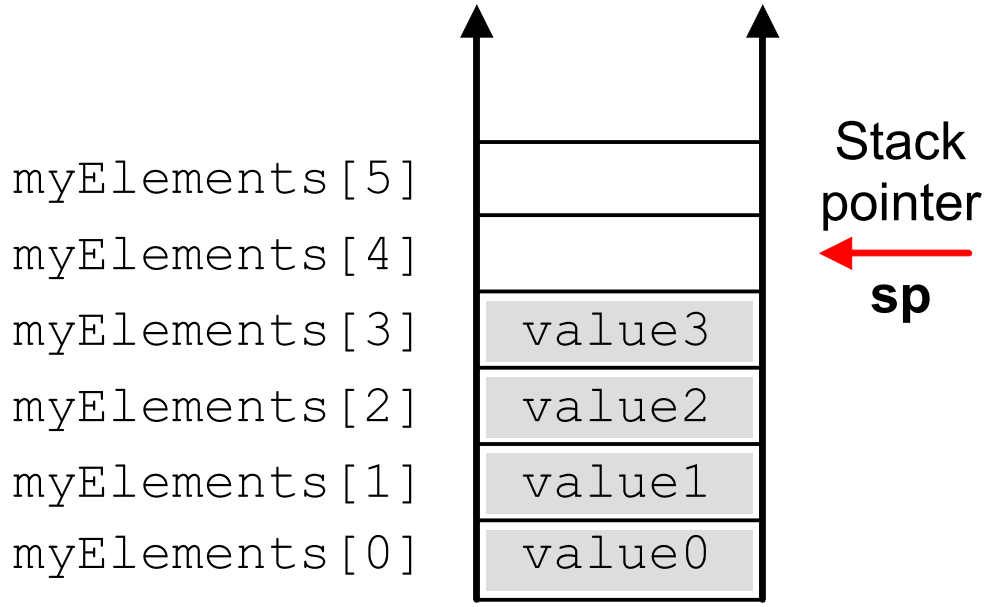
Stacks are Used for

- handling branching processes:
  - traversing a branching tree structure
  - planning a move in a chess game
  - tracking the sequence of method calls in a Java program (frame)



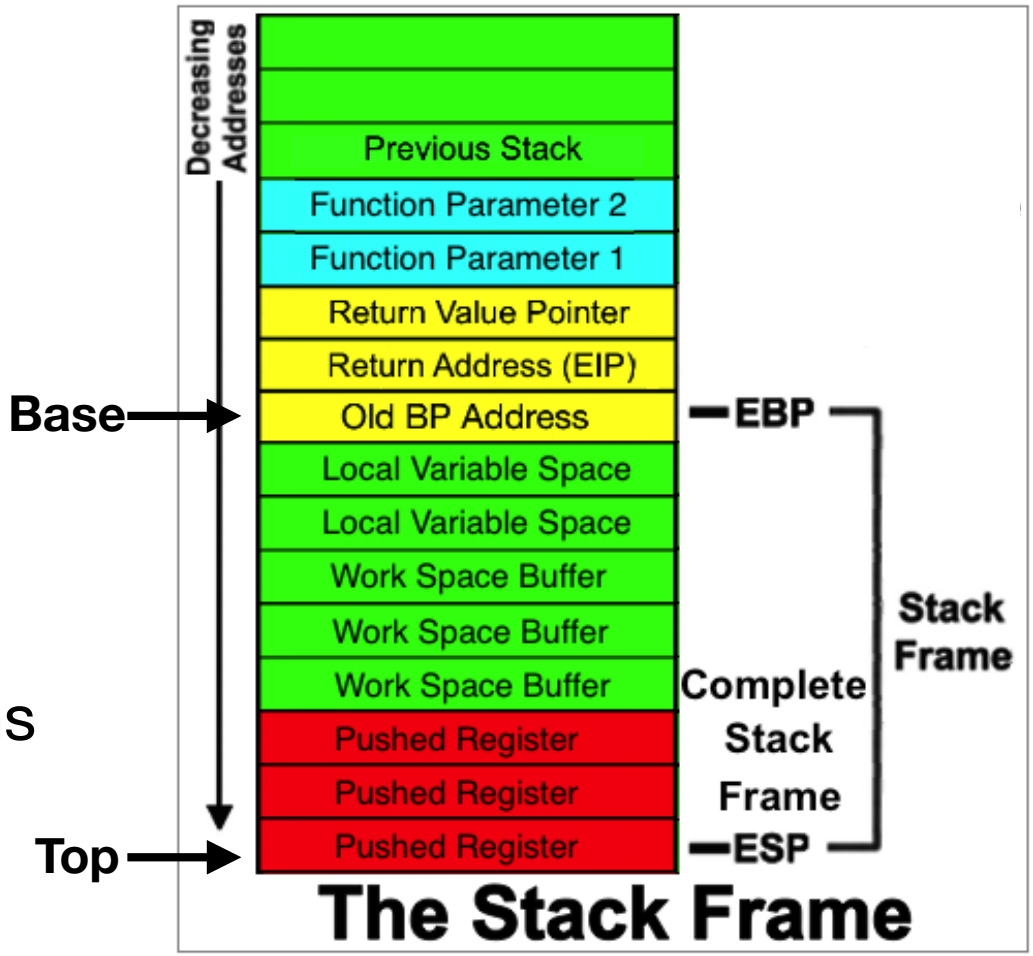
Stack Functions

- **push** - adding object to the “top” of the stack
- **pop** - removing object from the “top” of the stack
- **peek** - look at object on “top” of the stack
- **isEmpty** - true if the stack contains no objects; false otherwise



Hardware Stack Frame

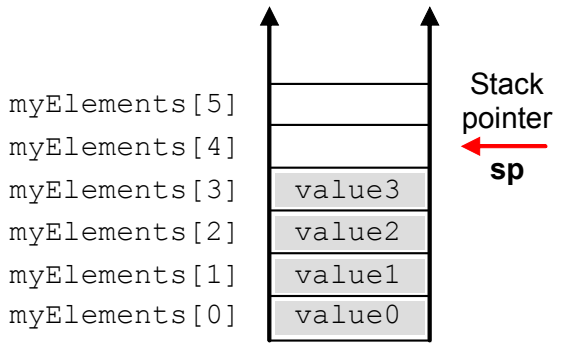
- A Java “method call” creates a “stack frame”.
- The frame contains all the necessary information to return (the “state”).
- Pointers
  - SP - Stack Pointer (in all stacks)
  - BP - Base Pointer (in stack frames)
- “Stack overflow” means large (infinite) number of method calls.



https://sites.google.com/site/illustratedcomputer/stackframe

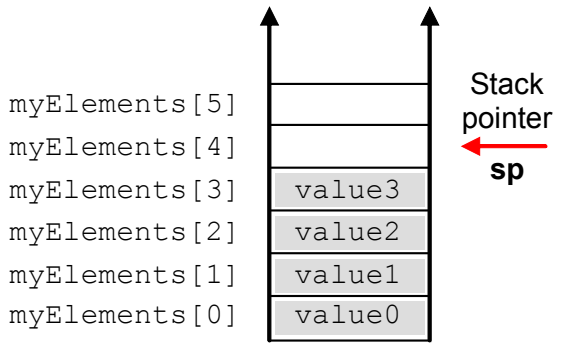
Properties of a Stack

- In an efficient implementation, **push**, **pop**, and **peek** methods run in O(1) time.
- **pop** and **peek** are expected to throw an **EmptyStackException** if the stack is empty.
- If we implement the stack using an **ArrayList**, we need to explicitly throw an **EmptyStackException**.



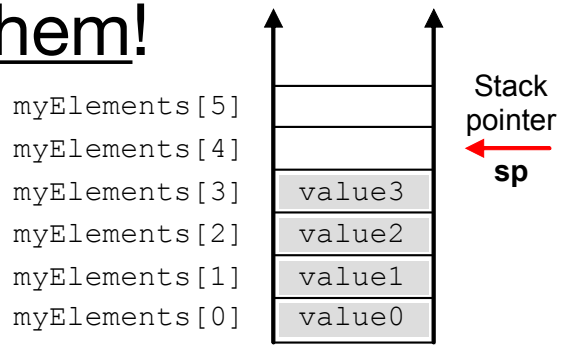
Properties of a Stack

- A stack of objects holds references to objects.
- If necessary, a stack can hold multiple references to the same object, but *be careful* !!! Changing a mutable object on a stack changes them all.
- It is best to **push** a copy of the object onto the stack, especially mutable objects, to insure all stack objects retain their original value.



java.util.Stack class

- Part of the Java Collections Framework (Chapter 19).
- **Stack** is a “generic” class. It works with objects of a specified type (e.g. **Stack<String>**).
- Based on the legacy **Vector** class, similar to **ArrayList**.
- **Stack** methods: **push**, **pop**, **peek**, **isEmpty**.
- Has other methods – but do not use them!



Simple Stack using ArrayList

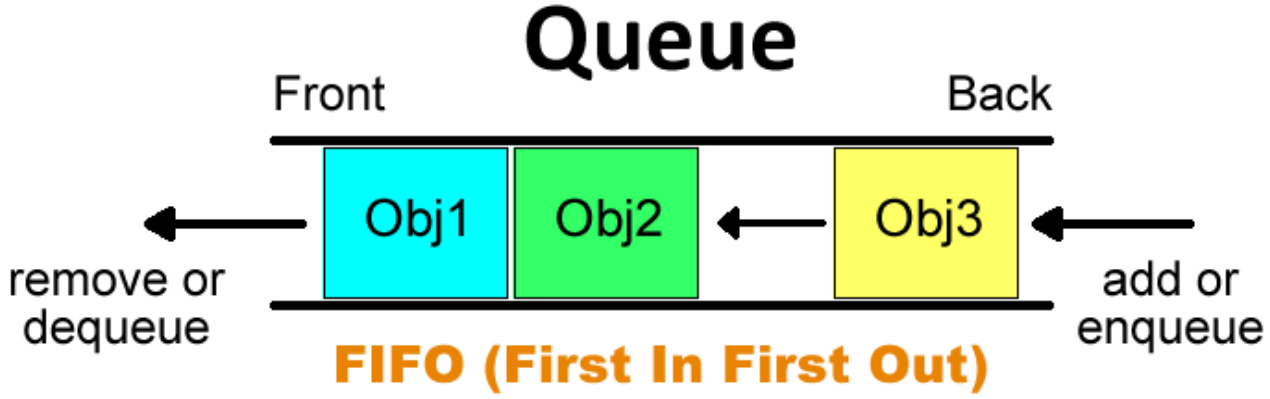
```
import java.util.ArrayList;
import java.util.List;

public class ArrayStack<E> implements Stack<E>
{
    private List<E> elements;

    public ArrayStack ( )
            {  elements = new ArrayList<E>( );  }

    public boolean isEmpty ( )   {  return elements.isEmpty ( );  }
    public void push (E obj)   {  elements.add (obj);  }
    public E pop ( )
            {  return elements.remove (elements.size ( ) – 1);  }
    public E peek ( )
            {  return elements.get (elements.size ( ) – 1);  }
}
```

Where is the “top” of an ArrayStack?

Queues are Used for

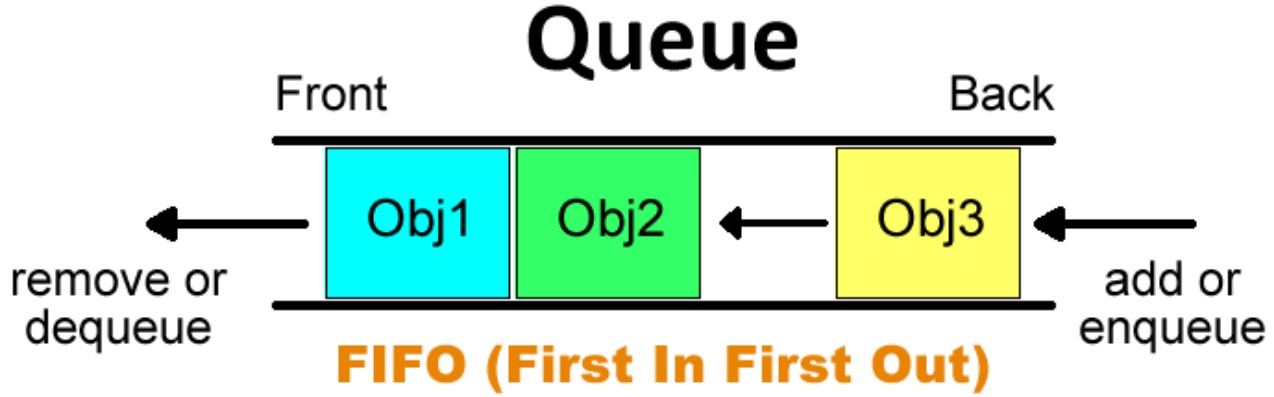
- Processing events or messages in order of their arrival

- System tasks
  - Queueing print jobs
  - Entering keystrokes
  - Processing mouse clicks



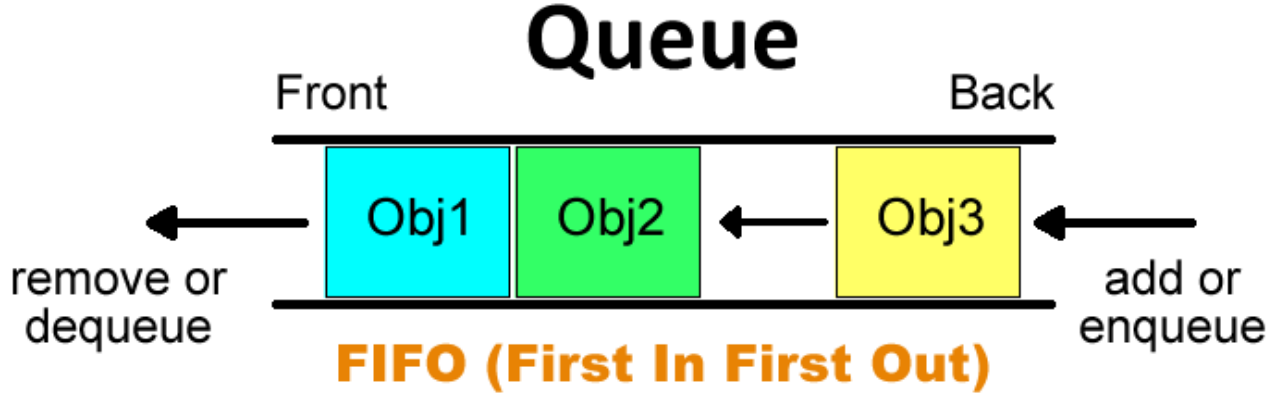
Queue Functions

- **add** - adding object to the “back” of the queue (enqueue)
- **remove** - removing object from the “front” of the queue (dequeue)
- **peek** - look at object at “front” of the queue
- **isEmpty** - true if the queue contains no objects; false otherwise



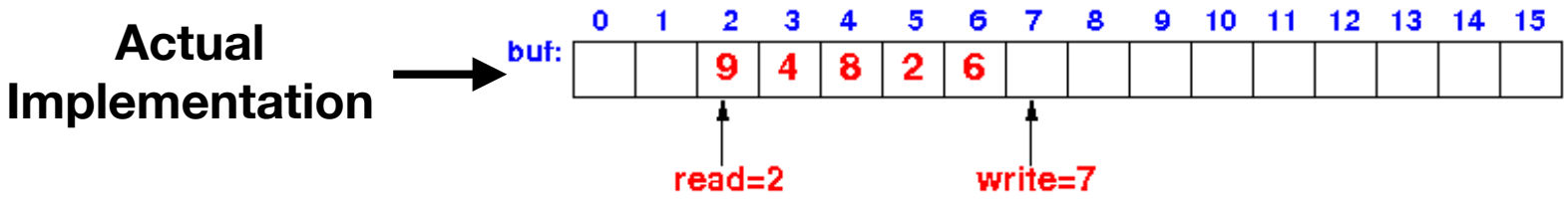
Properties of a Queue

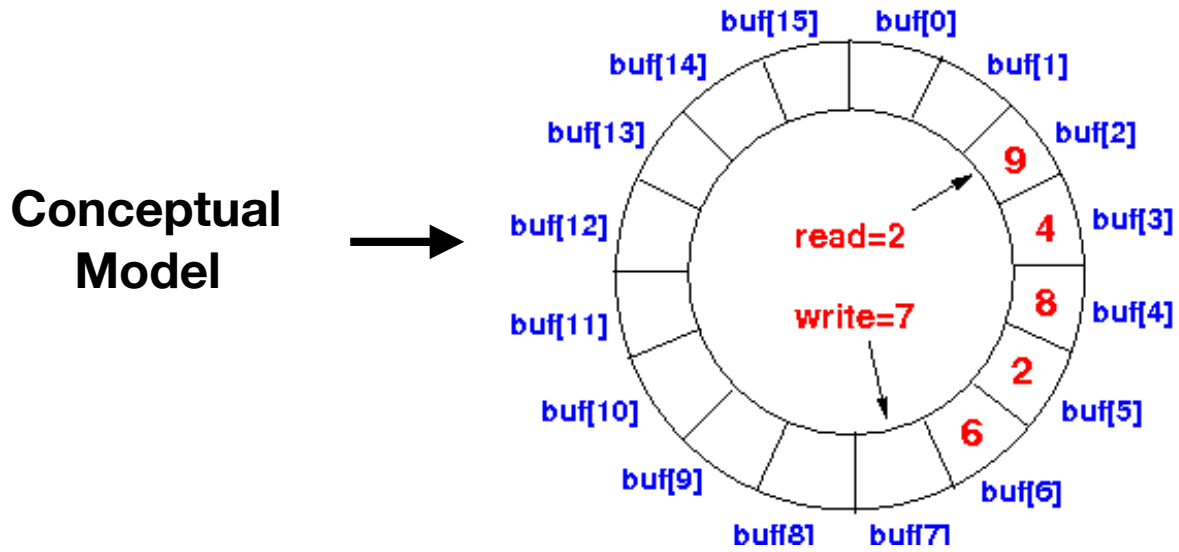
- In an efficient implementation, **add**, **remove**, and **peek** methods run in O(1) time.
- A queue of objects holds references to objects.
- Like a stack, a queue can hold multiple references to the same object.
- Like a stack, it is best to **add** a copy of the object onto the queue, especially mutable objects, to insure all queue objects retain their original value.



Ring Buffer

- An efficient implementation of a queue.
- Instead of shifting elements to the “front”, just keep track of front (“read”) and back (“write”) pointers.

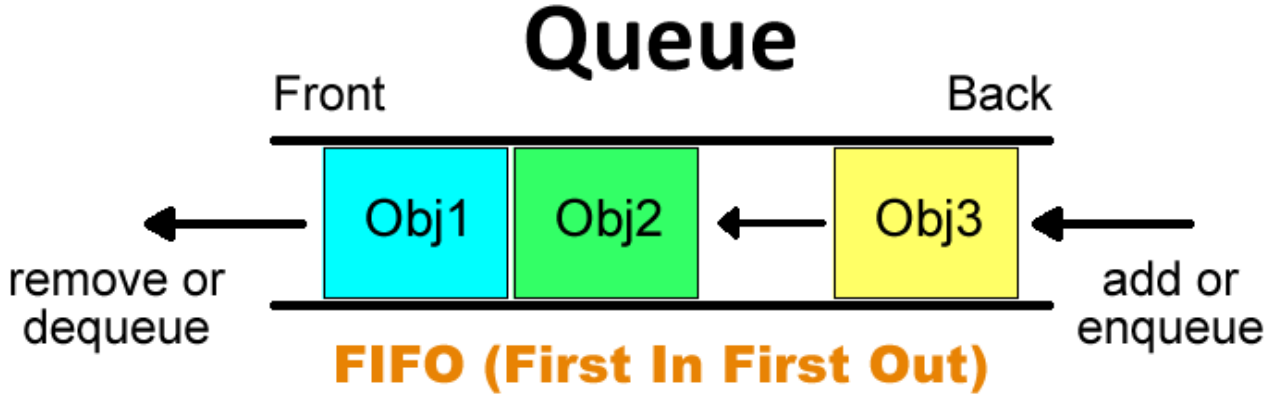




java.util.Queue interface

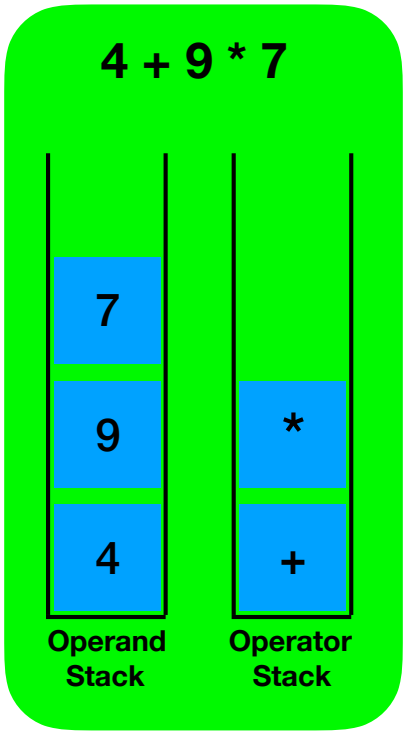
```
boolean isEmpty ()
boolean add (E obj)
E remove ()
E peek ()
```

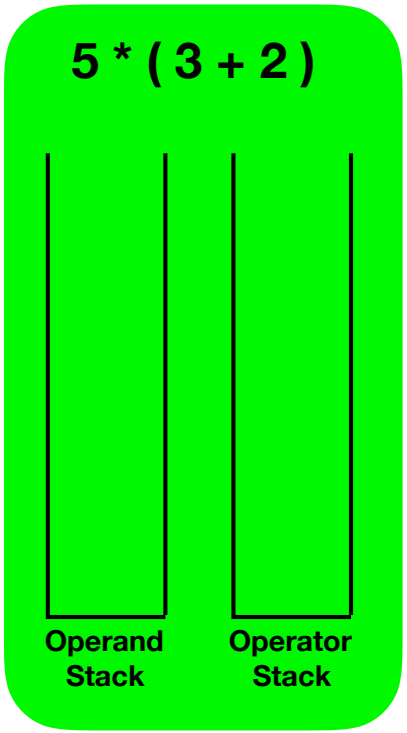
- A “generic” interface, part of the Java Collections Framework (Chapter 19)
- Implemented by other classes like **java.util.LinkedList** and **java.util.PriorityQueue**



SimpleCalc Project

- Evaluates arithmetic expressions using stacks.
- Two stacks: operand stack and operator stack.





Questions?