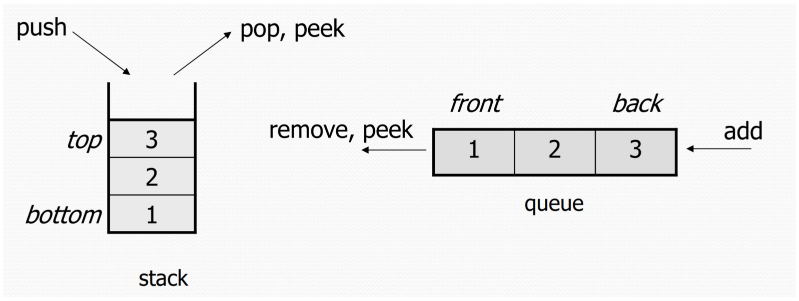
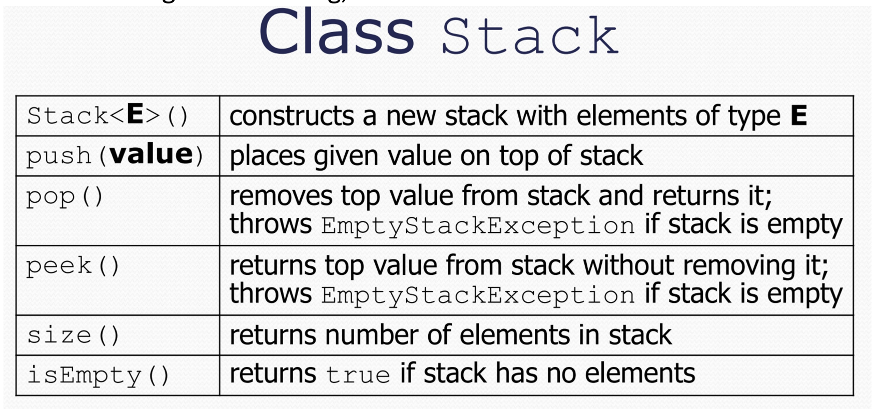
Stack & Queue

Wednesday, October 3, 2018 12:26 PM

- **Abstract data type(ADT)**: a specification of a collection of data an the operation that can be performed on it.

- Stacks and queues:
  Some collections are constrained so clients can only use optimized operations
  - **Stack**: retrieves elements in reverse order as added.
  - **Queue**: retrieves elements in same order as added.



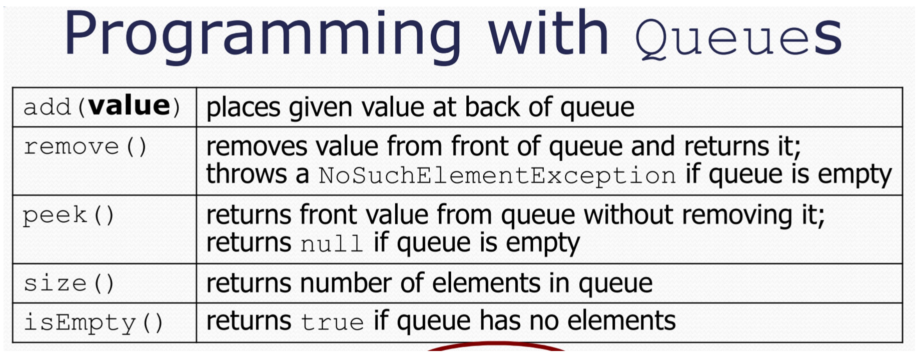
- Stack:
  - Last-In, First-Out("LIFO")
  - Client can only add/remove/examine the last element added(the "top")
  - Stack in CS:
    - Programming languages & compliers
    - Matching up related pairs of things
    - Sophistacted algorithms
      Eg. Backtracking, "undo stack"
  - 

Class `Stack`

`Stack<`**`E`**`>()`	constructs a new stack with elements of type **E**
`push(`**`value`**`)`	places given value on top of stack
`pop()`	removes top value from stack and returns it; throws `EmptyStackException` if stack is empty
`peek()`	returns top value from stack without removing it; throws `EmptyStackException` if stack is empty
`size()`	returns number of elements in stack
`isEmpty()`	returns `true` if stack has no elements

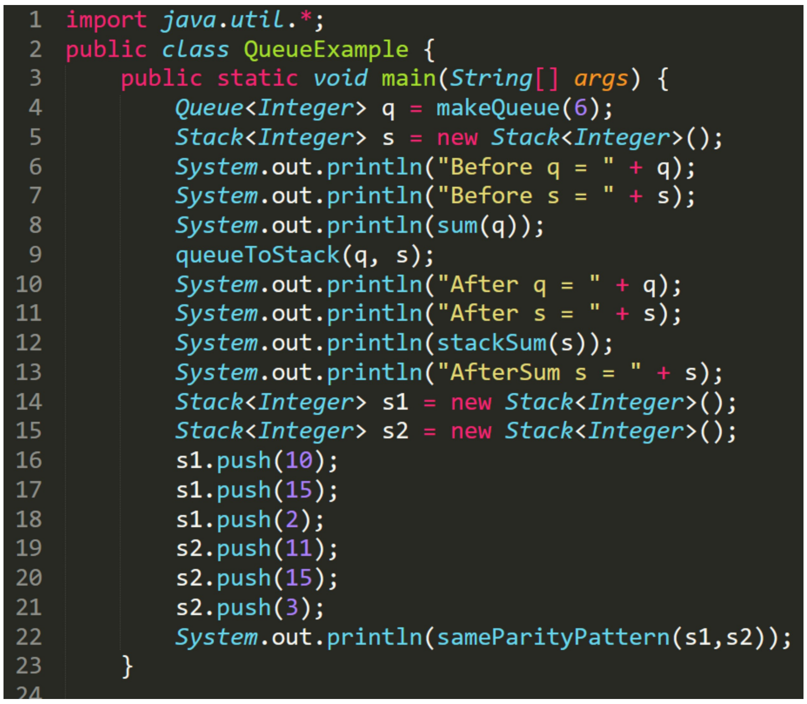
  - Stack has other methods that are off-limits(not efficient)
  - There is no stack interface.

- Queue
  - Retrieves elements in the order they were added
  - Fist-In, First-Out("FIFO")
  - Client can only add to the end of the queue, and can only examine/remove the front of the queue
  - Elements are sorted in order of insertion, but don't have indexes.
  - Queue is a interface.
    - Queue<Integer> q = new LinkedList<Integer>();
  - Queues in CS
    - Operating systems
    - Programming
    - Real world examples

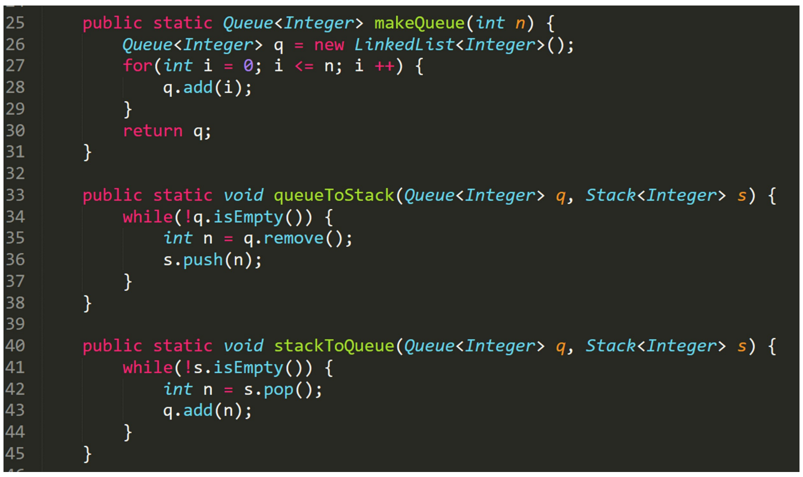


Programming with Queues

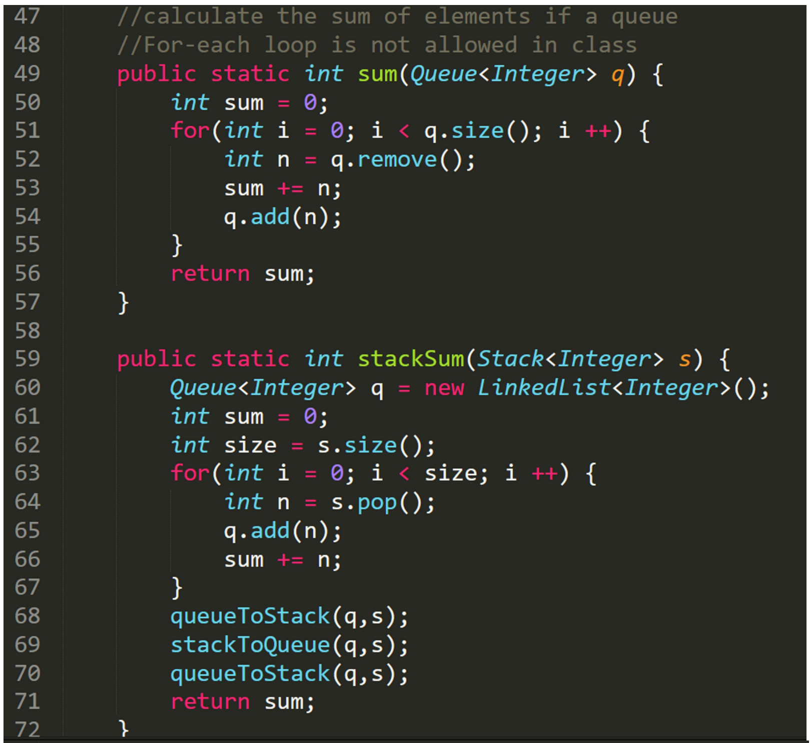
add(**value**)	places given value at back of queue
remove()	removes value from front of queue and returns it; throws a NoSuchElementException if queue is empty
peek()	returns front value from queue without removing it; returns null if queue is empty
size()	returns number of elements in queue
isEmpty()	returns true if queue has no elements



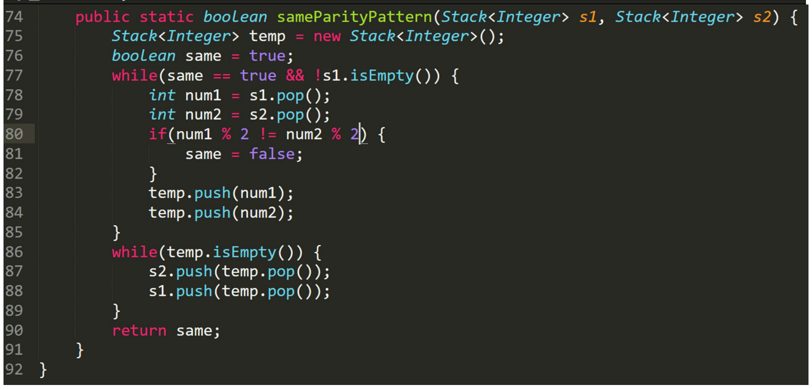
```java
import java.util.*;
public class QueueExample {
    public static void main(String[] args) {
        Queue<Integer> q = makeQueue(6);
        Stack<Integer> s = new Stack<Integer>();
        System.out.println("Before q = " + q);
        System.out.println("Before s = " + s);
        System.out.println(sum(q));
        queueToStack(q, s);
        System.out.println("After q = " + q);
        System.out.println("After s = " + s);
        System.out.println(stackSum(s));
        System.out.println("AfterSum s = " + s);
        Stack<Integer> s1 = new Stack<Integer>();
        Stack<Integer> s2 = new Stack<Integer>();
        s1.push(10);
        s1.push(15);
        s1.push(2);
        s2.push(11);
        s2.push(15);
        s2.push(3);
        System.out.println(sameParityPattern(s1,s2));
    }
```



```java
    public static Queue<Integer> makeQueue(int n) {
        Queue<Integer> q = new LinkedList<Integer>();
        for(int i = 0; i <= n; i ++) {
            q.add(i);
        }
        return q;
    }

    public static void queueToStack(Queue<Integer> q, Stack<Integer> s) {
        while(!q.isEmpty()) {
            int n = q.remove();
            s.push(n);
        }
    }

    public static void stackToQueue(Queue<Integer> q, Stack<Integer> s) {
        while(!s.isEmpty()) {
            int n = s.pop();
            q.add(n);
        }
    }
```



```java
    //calculate the sum of elements if a queue
    //For-each loop is not allowed in class
    public static int sum(Queue<Integer> q) {
        int sum = 0;
        for(int i = 0; i < q.size(); i ++) {
            int n = q.remove();
            sum += n;
            q.add(n);
        }
        return sum;
    }

    public static int stackSum(Stack<Integer> s) {
        Queue<Integer> q = new LinkedList<Integer>();
        int sum = 0;
        int size = s.size();
        for(int i = 0; i < size; i ++) {
            int n = s.pop();
            q.add(n);
            sum += n;
        }
        queueToStack(q,s);
        stackToQueue(q,s);
        queueToStack(q,s);
        return sum;
    }
```



```java
    public static boolean sameParityPattern(Stack<Integer> s1, Stack<Integer> s2) {
        Stack<Integer> temp = new Stack<Integer>();
        boolean same = true;
        while(same == true && !s1.isEmpty()) {
            int num1 = s1.pop();
            int num2 = s2.pop();
            if(num1 % 2 != num2 % 2) {
                same = false;
            }
            temp.push(num1);
            temp.push(num2);
        }
        while(temp.isEmpty()) {
            s2.push(temp.pop());
            s1.push(temp.pop());
        }
        return same;
    }
}
```