

# COMP 410 - Fall 2014

## Programming Assignment 1



### PROJECT DESCRIPTION

A **VipQueue** is a regular queue enhanced with an additional enqueue operation called an *vipEnqueue*, which allows an item to be added to the front of the queue. The following “main” class illustrates the use of a VipQueue of integers:

```
import comp410.vipqueue.VipQueue;
public class Tester {
    public static void main(String args[]){
        VipQueue<Integer> vq = new VipQueue<Integer>(10);
        for (int i=0; i<5; i++){
            if (!vq.isFull()) vq.enqueue((Integer)i);           //a "regular" enqueue
            if (!vq.isFull()) vq.vipEnqueue((Integer)(i*i));    //a vip enqueue
        }
        while (!vq.isEmpty()) System.out.printf("->%d", vq.dequeue());
    }
}
```

Executing the code above should yield output that looks like this:

->16->9->4->1->0->0->1->2->3->4

For this assignment you are to implement a VipQueue class that supports the operations illustrated in the Tester class above. This is how you should proceed.

- 1) Implement a Stack class that supports the operations in Appendix A. This Stack should be implemented as a linked list.
- 2) Implement a Queue class that supports the operations in Appendix A. This Queue should also be implemented as a linked list.
- 3) Implement your VipQueue class such that each VipQueue object is represented as a single Stack object and a single Queue object. That is, the *only* data members of your VipQueue class should be one Stack, one Queue, and perhaps some constant number of additional primitive variables (integers, booleans, etc.).
- 4) You may assume that the enqueue and vipEnqueue methods have the precondition that the vipQueue not be full, and that your dequeue operation has the precondition that the vipQueue not be empty. In other words, we will never test your program by adding to a full collection or removing from an empty one.

### NOTE

Do *not* use any in-built Java implementation (e.g., Stack, Queue, Lists, ArrayList, etc) from the java.util.\* API. You *must* implement the above from scratch.

### GRADING RUBRIC

- (15 points) Stack correctly implemented using a linked list
- (15 points) Queue correctly implemented using a linked list
- (10 points) Use of generics
- (15 points) VipQueue enqueue method implementation
- (15 points) VipQueue vipEnqueue method implementation
- (20 points) Other VipQueue methods
- (10 points) Clean code

## SUBMISSION INSTRUCTIONS

Upload all your source code in a .zip file to Sakai. You are responsible for ensuring that your program compiles and functions properly. Any non-functioning program will receive a zero.

**On Mac:** right-click on your `src` folder and select the *Compress "src"* item.

**On Windows:** right-click on your `src` folder and select *Send to* then select *Compressed (zipped) folder*

## HONOR CODE

Please review the honor code description from the course syllabus. No collaboration (with anyone) is permitted in assignments. Collaboration in assignments, or the use of code not the students own, constitutes an honor code violation. Any violation will be reported to the Student Attorney General.

## APPENDIX A: REQUIRED METHODS

Your `Stack` class should support the following constructor and operations (T is a generic type):

```
public Stack(int capacity)
public boolean isEmpty()
public boolean isFull()
public T peek()
public T pop()
public void push(T element)
```

Your `Queue` class should support the following constructor and operations (T is a generic type):

```
public Queue(int capacity)
public boolean isEmpty()
public boolean isFull()
public T peek()
public T dequeue()
public void enqueue(T element)
```

Your `VipQueue` class should support the following constructor and operations (T is a generic type):

```
public VipQueue(int capacity)
public boolean isEmpty()
public boolean isFull()
public T peek()
public T dequeue()
public void enqueue(T element)
public void vipEnqueue(T element)
```