CS 240 Homework 4

Alex Vondrak

Due: February 10, 2012

In general, data structures can be implemented in many different ways. These ways will impact the performance of the structure's operators—some for the better, others for the worse. We must therefore strike a balance between ease of implementation and efficiciency. While this homework won't give you a good idea of this balance, it will have you implement a data structure in an outlandish way.

Using the generic `ArrayStack<E>` class developed during the lectures, you will implement a queue. You must do this by having a class, `TwoStackQueue<E>`, that implements the following generic interface:

```java
interface Queue<E> {
   public void enqueue(E item);
   public E dequeue() throws QueueUnderflowException;
   public E peek() throws QueueUnderflowException;
}
```

`TwoStackQueue<E>` must have two fields: **`private`** `ArrayStack<E> main` and **`private`** `ArrayStack<E> aux`. `main` must hold the contents of the queue, and `aux` should be used for temporary storage of elements (as needed) in your queue methods.

The I/O for this homework will just perform operations on an instance of type `TwoStackQueue<Integer>` to ensure that you've implemented the methods correctly.

Input Format

Input consists of any number of whitespace-separated tokens. *Hint:* the `next` method of the `Scanner` class will find and return the next complete token.

Output Format

Starting with an empty queue, you must evaluate each token and print a corresponding output. A token will either be:

- The string `"dequeue"`, in which case you should print `dequeue = [result]`, where `[result]` is the returned value of the `dequeue` method. If a `QueueUnderflowException` is thrown, instead print `Queue underflow.`.
- The string `"peek"`, in which case you should print `peek = [result]`, where `[result]` is the returned value of the `peek` method. If a `QueueUnderflowException` is thrown, instead print `Queue underflow.`.

- A string representing an integer, in which case you should enqueue it, and print `enqueue [the integer]`.

  Hint: use methods of the `Integer` class (`http://docs.oracle.com/javase/6/docs/api/java/lang/Integer.html`).

- None of the above, in which case you should print out `Unrecognized operator: [the token]`.

After each such token is evaluated, print out a line with the contents of the queue from rear to front. If the queue is empty, print `[empty queue]`. See the output sample.

Input Sample

```
1 2 3
peek
dequeue peek
dequeue
push 4
dequeue dequeue dequeue
```

Output Sample

```
enqueue 1
1
enqueue 2
2 1
enqueue 3
3 2 1
peek = 1
3 2 1
dequeue = 1
3 2
peek = 2
3 2
dequeue = 2
3
Unrecognized operator: push
3
enqueue 4
4 3
dequeue = 3
4
dequeue = 4
[empty queue]
Queue underflow.
[empty queue]
```