CS 3304 HW 8: Eiffel Spring 2002

Due: 23 April, 11:59:59 p.m.

This homework revolves around defining a class for a generic Queue using a linked structure. You are to submit a single text document containing your solution.

Here is the ADT that you are to implement. The ADT `QUEUE[G]` holds elements of type `G` and implements the following abstract data type.

Types:

1. $QUEUE[G]$

Functions:

1. $new : QUEUE[G]$
2. $empty : QUEUE[G] \rightarrow BOOLEAN$
3. $enqueue : G \times QUEUE[G] \rightarrow QUEUE[G]$
4. $head : QUEUE[G] \not\rightarrow G$ (partial function)
5. $dequeue : QUEUE[G] \not\rightarrow QUEUE[G]$ (partial function)
6. $length : QUEUE[G] \rightarrow INTEGER$

Axioms:
For any $x : G$, $q : QUEUE[G]$

1. $empty(new)$
2. $not\ empty(enqueue(x, q))$
3. $head(enqueue(x, new)) = x$
4. $q \neq new$ and $head(enqueue(x, q)) = head(q)$
5. $dequeue(enqueue(x, new)) = new$
6. $q \neq new$ and $dequeue(enqueue(x, q)) = enqueue(x, dequeue(q))$
7. $length(new) = 0$
8. $length(enqueue(x, s)) = length(s) + 1$

Preconditions

1. $dequeue(q)$ requires $not\ empty(q)$
2. $head(q)$ requires $not\ empty(q)$

The following exercises ask you to produce a class and then add details to it. In the end you should have one class, perhaps with notes in comments to indicate what you have done.

1. Define a generic class `QUEUE[G]` that implements the ADT given above using a linked structure. Since we don't have pointers in Eiffel you will have to use references to link the nodes of the queue together. It is possible to use only one class, but of course a node class can be used as well.

2. In your class, illustrate the the use of feature access control to publish different interfaces for different classes. In particular, hide restrict access to routines needed for treating an object as a node in the linked structure. These routines should only be available to the `QUEUE[G]` class or a node class if you have one. The functions of the ADT should all be public.

3. Add the preconditions defined above to the corresponding procedures or functions of the class.

4. Add postconditions to the appropriate routines that correspond to axioms 1 and 2, and axioms 7 and 8. It may be useful to use the notation `old` that can be used in postconditions to refer to the value of an expression prior to entering the routine.