

Object Oriented Modeling and Simulation

Using Queues in OOSimL

José M. Garrido C.

Department of Computer Science
College of Computing and Software Engineering
Kennesaw State University

Spring, 2017

Types of Queues

OOsimL provides two general types of queues to store references to processes in simulation models; these are:

- simple queues
- priority queues

These queues are data structures that store references to processes. These references are inserted to the *tail* of the queue and removed from the *head* of the queue.

Illustration of a Queue

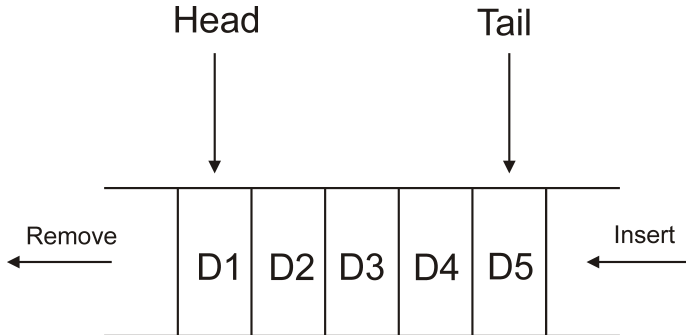


Figure: A queue

Simple Queues

To use simple queues in an OOSimL program, one or more objects of class *Squeue* must be created. These objects are simple first-in-first-out (FIFO) queues.

The **create** statement is used for creating a simple queue requires the name of the queue and the queue size (queue capacity). This last data item is optional; if not included, the assigned default size is 1,000.

Example Creating Queues

The following lines of code declare and create a simple queue, *custQueue* with an assigned name “Customer Queue” and size of 15 customers.

```
define cust_queue of class Squeue // declare queue
...
// now create simple queue
create custQueue of class Squeue using
    ``Customer Queue'', 15
```

Check Queue Full

To check for the general states of the queue, empty or full, conditions **full** and **empty** are provided. An **if** statement is used with these boolean values (true or false).

The following lines of code check if queue object *cust_queue* (defined above) is full.

```
if cust_queue is full then
    // if queue is full execute instructions here
    ...
else
    // queue is not full
    ...
endif
```

Check Queue Empty

In a similar manner, the **if** statement with the **empty** condition checks if a queue is empty.

The following lines of code check if queue object referenced by *cust_queue* (defined above) is empty. The process suspends itself if the queue is empty, otherwise, it proceeds to dequeue an object from the queue.

```
if cust_queue is empty then
    // if queue is empty, suspend here
    suspend self
else
    // dequeue object
    ...
endif
```

Length of Queue

The **assign length** statement gets the current number of objects in the specified queue (current queue size).

The general structure of the statement follows.

```
assign length of < queue_ref > to < variable >
```


Example Length of Queue

The following lines of code get the current length of the queue object referenced by *cust_queue* (defined above) and assigns it to variable *qlength*.

```
define qlength of type integer  
.  
.  
.  
assign length of cust_queue to qlength
```

Insert Object Into Queue

The **insert** statement inserts the object specified to the tail of the queue. The inserted object becomes the new object at the tail of the queue. The size of the queue is increased by one. The general structure of the statement follows.

```
insert < ref_variable > into < queue_ref >  
insert self into < queue_ref >  
enqueue < ref_variable > into < queue_ref >  
enqueue self into < queue_ref >
```

Example Queue Insertion

The following lines of code insert (enqueue) the object referenced by *cust_obj* into the simple queue referenced by *cust_queue*.

```
define cust_obj of class Customer
...
// enqueue customer process
insert cust_obj into cust_queue
```

Remove Object from a Queue

The **remove** statement dequeues (removes) the process at the head of from the specified simple queue. After removal, the size of the queue is reduced by one, as a result of this operation. The general forms of the statement are:

```
remove [object] < ref_var > of class < cl_name >  
      from [queue] < queue_ref >  
dequeue [object] < ref_var > of class < cl_name >  
      from [queue] < queue_ref >
```

Example Queue Removal

The following line of code removes (dequeues) a customer process from the head of the queue referenced by *cust_queue* and assigns it to reference *cust_obj*.

```
remove object cust_obj of class Customer from  
queue cust_queue
```

Get Object from a Queue

The **get** statement gets a copy of the process at the head of the specified simple queue, without removing it from the queue. The general form of the statement follows.

```
get [object] < ref_var > of class < cl_name >  
      from [queue] < queue_ref >
```

Example Queue Removal

The following line of code gets a copy of a customer process from the head of the queue referenced by *cust_queue* and assigns it to reference *cust_obj*.

```
get object cust_obj of class Customer from  
    queue cust_queue
```

Priority Queues

- Priority queues allow the simulation model to retrieve or remove objects from the queue, based on the priority of the objects or processes.
- As such, priority queues do not behave as FIFO queues.
- A priority queue is an object of class *Pqueue*.

Creating Priority Queues

The **create** statement is used to create a priority queue and requires:

- the name of the queue
- the optional number of different priorities; if this argument is not included, the number of priorities is assigned to 300 (default)
- the queue size for all priorities, which is also optional and the assigned default queue size is 1,000 for every priority.

Example Creating a Priority Queue

The following lines of code declare and create the priority queue with reference name *cust_queue*, with queue name “Customer Queue”, using 8 different priorities, and size (capacity) of 25 customers for every priority.

```
define cust_queue of class Pqueue
...
// create priority queue
create cust_queue of class Pqueue using
    ``Customer Queue'', 8, 25
```

Insert Object Into Priority Queue

The **insert** statement inserts the specified object into the priority queue. The size of the queue is increased by one. The priority of the process is automatically taken from the object to be inserted. The general forms of the statement are:

```
insert < ref_variable > into < queue_ref >  
insert self into < queue_ref >  
enqueue < ref_variable > into < queue_ref >  
enqueue self into < queue_ref >
```

Example Inserting to a Priority Queue

The following lines of code insert (enqueue) object *cust_obj* into priority queue *pcust_queue*.

```
define cust_prio = 3 of type integer
define cust_obj of class Customer // reference customer
...
// enqueue customer process
insert cust_obj into pcust_queue
```