Module: ReadyQueue

DESCRIPTION

The ReadyQueue process holds the threads waiting to execute.

FSP TRANSLATION

The ReadyQueue class is the implementation of the QUEUE process found in the model.

The ReadyQueue class is a monitor. The QUEUE process allowed execution of certain actions based upon its state variable waiting, which represents the number of Processes currently in the queue its modeling. The ReadyQueue class also has a state variable waiting- an integer that represents how many non-null entries there are in the threadQueue (An array of Process Objects).

The ReadyQueue class methods are accessed by the Generator, Dispatcher, and GrimReaper classes- this reflects how in the FSP model the actions in the alphabet of the QUEUE process are shared with the DISPATCHER, GENERATOR, and GRIMREAPER processes.

The ReadyQueue will block the Dispatcher thread if the threadQueue is empty (Just like how the QUEUE process cannot engage in the select action if waiting==0)

The ReadyQueue will block the Generator if the threadQueue is full or if a process is loaded onto the CPU (Our model does not allow the Generator to engage in the add\_Q action if the select action has been executed but the backInQueue or remove\_Q actions have not been reached yet)

The ReadyQueue will allow the GrimReaper thread to remove a Process from the threadQueue or move a Process to the back of the queue. After doing either of these actions, the ReadyQueue will notify any threads waiting on it, allowing for another state transition (Just as our QUEUE process engages in another QUEUE process with updated state variables)

The GENERATOR class engages in the

COMPARISONS

|  |  |  |
| --- | --- | --- |
| ACTIONS USED IN LTSA MODEL | METHODS USED IN JAVA IMPLEMENTATION | COMPARISON |
| select | select() | Both actions/methods indicate the selection of a thread for execution. Both methods are shared between the ReadyQueue/QUEUE and the Dispatcher/DISPATCHER class/process. |
| add\_Q | enqueue(Process p) | Both actions/methods indicate the addition of a thread to the queue. Both actions/methods are dependant on whether the queue is full. |
| remove\_Q | dequeue() | Both actions/methods indicate the removal of a thread from the queue.  Both actions/methods are shared in the Generator/GENERATOR, and GrimReaper/GRIMREAPER class/process. In the QUEUE process this action is one of the possible actions as a result of WHATNOW. The java implementation does not require a WHATNOW process to decide to the methods being joint to the GrimReaper class. The java implementation through a checkTime() method in the GrimReaper class which will decide the result. |
| backInQueue | backInQueue(Process p) | Both actions/methods indicate the addition of a previously terminated thread back into the Queue. Both actions/methods are shared in the GrimReadper/GRIMREAPER class/process. The backInQueue action is chosen as a result of the checkTime action in the GRIMREAPER. Those actions are represented in the implementation through the method checkTime() in the GrimReaper class. |

Field Variables:

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Meaning |
| waiting | int | The number of threads waiting in the Queue |
| threadQueue | Thread[5] | The queue of processes waiting to be executed. |
| isLoaded | boolean | A boolean representing whether or not a process should be added to the queue. Corresponds to whether a process has been selected. |

INTERFACE:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
| ReadyQueue | void | ReadyQueue | Constructor | INITIAL STATE:  waiting=0  threadQueue=new Process[5] |  |
| enqueue(Process p) | A Process representing the new thread to be put into the threadQueue | void | Waits for the thread to have space and for the CPU to be clear(isLoaded==false), then adds the given process to the threadQueue | waiting++  threadQueue[lastEmptyIndex]=p | this.isFull()  this.findEmptyIndex()  notify()  wait() |
| dequeue() | void | void | Takes a thread out of the queue.  Notifies any thread waiting on this class's monitor | waiting--  for(i=1,i<lastNonEmptyIndex;i++){  threadQueue[i-1]=threadQueue[i]  threadQueue[lastNonEmptyIndex]=null  isLoaded=false | findNonEmptyIndex()  wait()  notifyAll() |
| backInQueue(Process p) | A Process p that represents the modified thread at the front of the queue  (This thread was taken by the dispatcher and is now being returned by the Grim Reaper) | void | Puts the first item in the threadQueue in the last available spot, then notifies any thread that's waiting. | placeHolder=threadQueue[0]  for(i=1<, i<lastNonEmptyIndex;i++){  threadQueue[i-1]=threadQueue[i]  threadQueue[lastNonEmptyIndex]=placeHolder  isLoaded=false; | findNonEmptyIndex  notifyAll() |
| select() | void | The process at the head of the threadQueue | Waits until the thread is not empty, then returns the thread at the head of the Queue | isLoaded=true; | isEmpty()  wait() |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
| findEmptyIndex | void | int | Finds the index of the first null element in the ThreadQueue and returns it | - | - |
| findNonEmptyIndex | void | int | Finds the index the the last element in the threadQueue | - | - |
| isFull() |  | boolean | Returns a boolean representing whether the queue is full or not | - | - |
| isEmpty() | void | boolean | Returns a boolean representing whether the thread is empty or not | - | - |