Module: ReadyQueue

Uses: None

INTERFACE:

|  |  |  |
| --- | --- | --- |
| 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. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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, then adds the given process to the threadQueue | waiting++  threadQueue[lastEmptyIndex]=p | isFull()  findEmptyIndex()  notifyAll()  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 | 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 | 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 | N/A | 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 |  |  |  |

Module: Process

Extends: Thread

INTERFACE:

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Meaning |
| executionTime | long | represents how many clock cycles of work it has left |
| id | int | Represents the id number of the Process |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
|  |  |  |  |  |  |
| @Override  run() | void | void | Waits until the execution time has passed or until notified/ |  | wait(integer long) |
| Process | int ID | Process | Constructor  Generates a random number for executionTime and returns the ProcessObject | id=ID  executionTime=Random.nextLong | java.util.Random |
| updateTime(int Long l) | A long integer representing how long the Process ran for (CPU maxRunTime) | integer Long  how long the thread ran |  | int newTime=executionTime-l  int r = executionTime;  if (newTime>0){  executionTime=newTime;  return newTime  }  else{  executionTime=0;  return r;  } |  |
| checkTime() | void | integer Long  returns the executionTime |  |  |  |

IMPLEMENTATION

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Meaning |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method Name | Input | Output | State Transitions | Dependencies |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Module: Generator

Uses: ALL SCREENS

Extends:

INTERFACE:

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Meaning |
| readyQ | ReadyQueue | represents the ReadyQueue Process |
| scheduler | RRScheduler |  |
| waiting | int | represents how many queues are in the readyQueue |
| processCount | int | represents how many processes have been generated |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
| Generator | ReadyQueue readyQ |  | Constructor | this.readyQ=RRscheduler.readyQ  waiting=readyQ.threadQueue.length |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

IMPLEMENTATION

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Meaning |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method Name | Input | Output | State Transitions | Dependencies |
| enqueue() | void | void | Creates a new Process and adds it to the readyQueue | t=new Process(processCount)  readQ.enqueue(t)  processCount++ | Process constructor |
| checkQueue | void | void | checks the size of the readyQ | waiting=readyQ.readyQueue.length |  |
| generateMessage() |  | none | Generates a message of the form  "Process # loaded into ready Queue) |  |  |

Module: Dispatcher

Uses: ALL SCREENS

Extends:

INTERFACE:

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Meaning |
| readyQ | ReadyQueue | represents the ReadyQueue Process |
| cpu | CPU | represents the CPU process |
| processSelected | Process | the selectedProcess |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
| Dispatcher | ReadyQueue readyQ, CPU cpu | Dispatcher | Constructor | this.readyQ=readyQ  this.cpu=cpu |  |
| run | void | void | The dispatcher class tries to select things from the readyQ process and then attempts to pass them into the CPU.  The dispatcher process waits until the cpu thread finishes executing before trying to select again.  The dispatcher generates a message each time processSelected changes. | processSelected=readyQ.select() | this.select()  join() |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
| select | void | void | take a Process from the readyQueue  Generate a message indicating a Process was selected | processSelected=readyQ.select() |  |
| generateMessage() |  | none | Generates a message of the form  "Process # loaded into CPU) |  |  |
|  |  |  |  |  |  |

Module: CPU

Extends: Thread

INTERFACE:

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Meaning |
| maxExecutionTime | int long | represents the max executionTime a Process may run |
| reaper | GrimReaper | represents GrimReaper process |
| loadedThread | Process | The current thread on the CPU |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
| CPU | (int long maxExecution, GrimReaper reaper) |  |  | maxExecutionTime=maxExectution  this.reaper=reaper  loadedThread=null; |  |
| run | void | void | Runs the loadedThread  Updates the amount of time left on the thread  Sends the loadedThread to the GrimReaper, then finishes executing (And thereby "freeing" the Dispatcher) |  | execute() |
| load | Process p | void |  | loadedThread=p; |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
| unload() | Represents a Process being unloaded from the CPU |  | Passes the Process object to the GrimReaper. Starts the GrimReaper. | reaper.give(loadedThread)  reaper.run()  loadedThread=null; |  |
| execute() |  |  | Runs the Process and then updates the remaining execution time | loadedThread.run()  wait(maxExecutionTime)  loadedThread.notify()  loadedThread.updateTime(maxExecutionTime) |  |
| generateMessage() |  | none | Generates a message of the form  "Process # executed for x seconds" |  |  |

Module: GrimReaper

Uses: ALL SCREENS

Extends:

INTERFACE:

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Meaning |
| currentThread | Process | Current thread to have its fate decided |
| readyQ | ReadyQueue |  |
|  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
| GrimReaper | (ReadyQueue readyQ) | GrimReaper | Constructor | this.readyQ=readyQ  currentThread=null; |  |
| checkTime(Process) |  |  |  | currentThread=Process  judge(); |  |
| give(Process p) |  |  |  | currentThread=p |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
|  |  |  |  |  |  |
| judge() |  |  |  | int timeLeft=currentThread.getExecutionTime()  if(timeLeft>0){  backInQueue(currentThread)  else{  dequeue();  currentThread=null;  } |  |
| OVERLOADED  generateMessage(id,x)  generateMessage(id) | int id  int x  x represents how many seconds a process with id number id has executed  id represents a processID | none | Generates a message of the form either  "Process id finished executing"  "Process id recycled into ready queue with x seconds remaining" |  | Generates a message and prints it  Integer.toString |
|  |  |  |  |  |  |

Module: RRScheduler

Uses: ALL SCREENS

Extends:

INTERFACE:

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Meaning |
| readyQueue | ReadyQueue |  |
| generator | Generator |  |
| dispatcher | Dispatcher |  |
| cpu | CPU |  |
| grimReaper | GrimReaper |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

IMPLEMENTATION

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Meaning |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name | Input | Output | Description | State Transitions | Dependencies |
| run() |  | Runs all the processes! | Runs all the processes! |  | Generator  GrimReaper  Dispatcher  ReadyQueue  CPU |
|  |  |  |  |  |  |
|  |  |  |  |  |  |