CPSC 3333 Programming Assignment 2

Overview

It is important that you name the required classes and methods exactly the same (case included) as specified. Testing will fail if you do not.

This week we will build upon our simple FIFO queue and create a shortest-job-first queue and a priority-scheduling queue.

Deliverables

You must submit the following files:

MasterQueue.java

Pcb.java

Specifications

For this assignment you can use any library provided by java (e.g. ArrayList, LinkedList, etc.) I will give you the freedom to select what you think works best. You may also have as many private variables and private methods as you see fit. You are only allowed to have the public methods specified below. You are not allowed to have any public variables.

Please use the same files you used in Programming Assignment 1. We will use many of the same methods we used last week.

New additions will be in **bold**.

- Pcb.java
  This will implement a simple PCB. It will hold the following attributes, Low memory bound (integer), High memory bound (integer), state (integer), process id (pid) (integer), **priority (integer), and bursts (integer)**.

  Please leave your four-parameter constructor and create a new six-parameter one.

  The constructor will take the **six** parameters specified above.

  This class must have **tweleve** public methods:

  void setLowMem(int)

  this method will set the value of the low memory bound variable.

  Int getLowMem()

  This method will return the integer value of the low memory bound variable.

  void setHighMem(int)

  this method will set the value of the high memory bound variable.

  Int getHighMem()

  This method will return the integer value of the high memory bound variable.

  void setState(int)

  this method will set the value of the state variable.

  Int getState()

  This method will return the integer value of the state variable.

  void setPid(int)

  this method will set the value of the pid variable.

  Int getPid()

  This method will return the integer value of the pid variable.

  void setPriority(int)

  this method will set the value of the priority variable.

  Int getPriority()

This method will return the integer value of the priority variable.

void setBursts (int)

this method will set the value of the bursts variable.

Int getBursts ()

This method will return the integer value of the bursts variable.

- MasterQueue.java

  This will hold all the queues for the system**. This week the name of each queue will be what time of queue it is, either a shortest-job-first queue or a priority-scheduling queue.**

  The names you MUST use are "SJF" or "PS" for the two queues.

  The names you MUST use are "SJF" or "PS" for the two queues. (repeated on purpose)

  The constructor will take no parameters.

  This class must have three public methods:

  boolean addQueue(String)

  This method will have a String parameter. It will return a boolean.

  It will add a new queue specified by the parameter. E.g. addQueue("ready") would create a new queue called "ready" to the list of queues. If there is already a queue by the specified name, then this method will return false. E.g. If you already have a queue named "ready" and you call addQueue("ready"), it will return false. Otherwise, it will create the queue and return true.

  boolean addPcb(Pcb, String)

  This method will have two parameters, a Pcb (class created by you) and a String. It will return a boolean.

  It will add the specified Pcb to the specified queue. **This addition must follow the rules of the two different queue types. E.g. Assume we are adding a PCB to an SJF queue. Assume the new PCB will use 8 bursts and the current PCBs in the queue have bursts of 3 and 9. After the new PCB has been added the queue will have PCBs with bursts of 3, 8, then 9, in that order. You can assume that there will be jobs with identical bursts or priorities. If so, the new job will move in line *behind* all current jobs in the queue with the same burst / priority.**

Also new this week, we must ensure that all PCBs have a unique process ID. If a PCB is submitted with a repeated process ID, then nothing should be added to the queue and this method should return false.

If the process id is unique and the new PCB has been added into the queue, this method should return true.

Pcb removePcb(String)

This method will have a String parameter. It will return a Pcb.

It will remove the first Pcb from the queue specified by the String parameter and return that Pcb.

If the specified queue does not exist, then this method will return a Pcb with pid = -1 (the values of the other fields do not matter as long as the pid = -1).

If the specified queue does not have any Pcbs (i.e. the queue is empty) then it will also return a Pcb with pid = -1. E.g. Assume we created our MasterQueue, and then we added the queue “ready”, but we have yet to add a Pcb to “ready”. Calling removePcb(“ready”) would return a Pcb will pid = -1.