In this lab you will write a print server. A print server is a program that accepts print jobs, sends print jobs to printers, and queues print jobs if there is no printer available. A real print server gets commands from the operating system (such as: "here is a print job") and printers (such as: "print job complete"). Our print server will get print jobs from an input file and will calculate the time when a given print job is done.

The input to the program will be a file that contains commands for the print server. Your file should be named **printjobs.dat**. Each command also has a time, which is the time at which the command is received by the print server. Our times will be simple numbers, like time 1, 2, 3, etc. The commands are in the file in increasing time order. There are commands for a printer coming online, a print job being submitted, a printer going offline, and a print job being cancelled.

In your program you will have classes for the print server, the printers, the print queues (one queue for each printer), and the print job itself. I will give you the queue class.

**Print Job**

A print job has a job number (int), the number of bytes in the job (int), and the logon id of the owner of the job (String). A real print job would also contain the data to be printed, but we won't have this in our similation.

The print job needs an equals method that compares the job numbers of two print jobs.

**Printer**

A printer has:

* the speed of the printer (in bytes per second)
* the number of the print job currently being processed
* the time at which the current job will finish printing
* a queue to hold print jobs waiting to print on this printer
* the total size of all the jobs waiting to print
* the printer number
* flags for the printer status (online/offline, printing/idle)

The methods of a printer object will partly depend on how you write your print server. Each printer has a speed in bytes/sec that is fixed for that printer. The printer number is assigned to the printer by the print server. You may need some sets and gets but create them only as needed. Note the following:

* the constructor will set the speed of the printer
* provide an equals method that compares the job number
* the print server will set the printer number when it brings the printer online
* when adding a print job to the printer's queue you must update the total number of bytes of all jobs in the queue
* when removing a print job from the printer's queue you must update the total number of bytes of all jobs in the queue and return the print job removed from the queue to the print server
* when starting a print job you must set the printer to "printing", calculate the end time of the job, set the number of the job being printed, and update the total number of bytes of all jobs in the queue; you need the current time so you can calculate the end time

**Print Server**

The print server object has an array of printers. Read in the number of printers and the speed of each printer from the printers.dat file, then initialize the array and initialize each printer with the appropriate speed.

The print server has to handle the following operations:

* bring a printer online: activate a printer, given the printer number and the current time.
* print: given a print job and the current time, choose a printer to print the job (send the job to the printer which has the smallest (least number of bytes to print) queue).
* cancel a print job: given a job number and the current time, try to remove the job from each print queue.
* take a printer offline: the job (if any) that was printing is terminated, and any jobs waiting in that printer's queue are distributed to the other printers. Each job that is deleted from the offline printer's queue should be sent to the printer that has the shortest queue. (Make sure you calculate which queue is the shortest for each print job, don't just calculate it once and move all print jobs to that queue.)
* check all printers: given the current time, check each printer to see if it has a print job finishing at the current time. If so, then check each printer to see if it has a print job in queue that can start at the current time.

Your print server should stay active until there are no more commands to process and all printers are idle with empty queues.

Make sure that your methods do not get too long; use private methods to make your public methods shorter.

**Queue**

A canceled a print job must be removed from the print queue, even if it is not at the front of the queue. Thus we need to add a delete method to the queue class. The delete method must find the item and remove it. If the print job is not found, throw a NoSuchElementException.

**Input**

The input to the program is read from two files. One file contains the initialization info for the printers. The other file contains the print server commands.

The printer initialization file is named **printers.dat**. The first line of the file contains the number of printers. Following that is the speed for each printer.

The print server file is named **printjobs.dat**. Each line of the file contains one print server command. Each command starts with the time the command is submitted to the print server. The commands are in the input file in order by time. The commands that the program will read from the input file are:

* time o printernum  
  In this command printernum is the number of the printer. The print server should bring the printer online, so print jobs can be added to its queue and it can start printing them. When this command is executed the program should print a message giving the time and saying that the printer is now online.
* time p jobnum size owner  
  In this command jobnum is the number assigned to the print job, size is the number of bytes being printed, and owner is the logon id of the user who submitted the print job. The print server should put the new print job into a print queue so that it will be printed when the printer is free. The program should put the job in the "shortest" queue (the one that contains the fewest total bytes). When this command is executed the program should print a message listing the job number, the time, and which printer the job is queued for.
* time f printernum  
  In this command printernum is the number of the printer which is going offline. The print server should terminate the job currently printing (if any) on this printer, remove each job from the print queue for this printer and put it in another print queue, and make the printer inactive so no more jobs will be sent to this printer. When this command is executed the program should print a message giving the time and saying that the printer is now offline. If a print job was terminated then a message saying the job was terminated should also be printed.
* time c jobnum  
  In this command jobnum is a print job that is canceled. The print server should attempt to remove the job from each printer, since it could be in the queue of any printer. If the job is removed from a queue the program should print a message giving the time and saying that the job was canceled. If the job is not found in any print queue, the program should print a message giving the time and saying that the job could not be canceled because it was not found.

**Client Code**

The simulation has a loop based on time. Each pass through the loop represents a time. In our program we will start at time 0, and each time through the loop we increment the time by 1. When the current time is the same as the time of an event, then the event is processed. Events are controlled by commands (printer online, printer offline, print a job) and by the jobs (job is done printing). Write your loop to read and process any commands for the current time, and then check the printers to see if any printers have a job ending. Process any ending jobs.

Read the time of the first command before the loop. When the current time (incremented each time through the loop) equals the time of the command, read the rest of the command and process it. Then read the time of the next command. Since there can be multiple commands at the same time, create an inner loop to read and process commands as long as the time of the commands is the same. (See the sample input below for an example of this.) Exit this loop when you read a command for a later time. Then check the printers to process any print jobs that are ending at that time.

Make sure that your loop continues until all commands are processed and all jobs are finished printing. It may be easier to have one loop which ends when all commands are processed followed by another loop to finish the jobs that are still printing after all commands have been read in.

**Output**

As described above, the program should print a message when a printer comes online, when a printer goes offline, when a job is sent to a printer, when a job starts printing, when a job finishes printing, when a job is terminated during printing because the printer is going offline, and when a job is canceled. You can print the messages in the print server functions or in the printer functions. Each message should include the current time. For example:

Time 2: Printer 1 online

Time 3: Print job number 100 queued for printer 0

Time 3: Printer 0 printing job 100

Time 11: Printer 0 offline, Print job 101 not completed

Time 14: Printer 1 completed job 103

Time 16: Print job number 150 canceled

Time 20: Print job number 220 not found/not canceled

**Sample Input**

printers.dat

3

800 1600 1200

printjobs.dat:

1 o 0 800

2 o 1 1600

3 p 100 3000 davidson

6 p 101 4096 berrios

8 p 103 8888 berrios

11 f 0

12 p 108 12800 deshpande

13 p 111 16384 khan

18 p 202 16000 davidson

20 o 0 800

21 p 102 7462 lee

22 c 333

23 p 114 5000 khan

23 p 115 12000 lee

23 p 116 8192 davidson

24 p 117 11000 clipper

25 p 118 6000 lee

26 c 116