CodeCrunch

CS2030 Discrete Event Simulator (Level 6)

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

Discrete Event Simulator Version 2

Requirements

- OOP Design consideration: continuation with switch-based event dispatching by event type
- Program documentation: adherance to CS2030 Javadoc

Problem Description

Let's make some changes to the queuing system in the previous lab. Specifically,

- there are now $k \geq 1$ servers, arranged in order from 1 to k;
- only one waiting customer is allowed per server;
- when a customer arrives at the shop:
    1. he/she scans the servers from 1 to k, and approaches the first idle server to be served immediately;
    2. if there is no idle server, the customer scans the servers from 1 to k, and waits at the first busy server without any waiting customer;
    3. if every server is busy and already has a waiting customer, he/she leaves the shop.

As a result of this, you might realize that there is a better way to encapsulate the data and the behavior of the various entities in the program. In which case, you may want to consider re-organize=ing your solution.

Keep in mind that there are still five states of the system, namely `ARRIVES`, `SERVED`, `WAITS LEAVES` and `DONE`. and only one of three possible state transitions for each customer:

- `ARRIVES` → `SERVED` → `DONE`

- `ARRIVES` → `WAITS` → `SERVED` → `DONE`
- `ARRIVES` → `LEAVES`

Statistics of the system that we need to keep track of are:

1. the average waiting time for customers who have been served
2. the number of customers served
3. the number of customers who left without being served

The Task

Input now consists of the following (in order of presentation)

- number of servers;
- set of customer arrival times in chronological order

Output comprises the individual discrete events, and also the statistics at the end of the simulation.

Take note of the following assumptions:

- The maximum number of events is 100 **at any one time**;
- The format of the input is always correct;
- Output of a `double` value, say `d`, is to be formatted with `String.format("%.3f", d);`

As this lab is a continuation form the previous one, this task will be treated as Level 6. Your programs for `sim1` to `sim5` in the previous lab has been made available to you. Just remember to

- define a `Main` class with the `main` method to handle input and output;
- check for output format correctness using the `diff` utility (see specific level for usage details). Note that only **one** test case is provided for this;
- check for styling errors by invoking `checkstyle`. For example, to check styling for all java files

```
$ checkstyle *.java
```

- save a copy of all source files into the appropriate level directory (see specific level for usage details).

Level 6

Discrete event simulation with k servers

You might want to first include the `k` servers, but only work with the first server, so that the output you get is consistent with the single server simulator.

Thereafter, whenever an event is picked from the queue, process that event by consider all `k` servers and generate the new event.

The following is a sample run of the program. User input is underlined.

```
1
0.500
0.600
0.700
1.500
```

```
1.600
1.700
0.500 1 arrives
0.500 1 served by 1
0.600 2 arrives
0.600 2 waits to be served by 1
0.700 3 arrives
0.700 3 leaves
1.500 1 done serving by 1
1.500 2 served by 1
1.500 4 arrives
1.500 4 waits to be served by 1
1.600 5 arrives
1.600 5 leaves
1.700 6 arrives
1.700 6 leaves
2.500 2 done serving by 1
2.500 4 served by 1
3.500 4 done serving by 1
[0.633 3 3]
```

```
2
0.500
0.600
0.700
1.500
1.600
1.700
0.500 1 arrives
0.500 1 served by 1
0.600 2 arrives
0.600 2 served by 2
0.700 3 arrives
0.700 3 waits to be served by 1
1.500 1 done serving by 1
1.500 3 served by 1
1.500 4 arrives
1.500 4 waits to be served by 1
1.600 2 done serving by 2
1.600 5 arrives
1.600 5 served by 2
1.700 6 arrives
1.700 6 waits to be served by 2
2.500 3 done serving by 1
2.500 4 served by 1
2.600 5 done serving by 2
2.600 6 served by 2
3.500 4 done serving by 1
3.600 6 done serving by 2
[0.450 6 0]
```

Check the format correctness of the output by typing the following Unix command

```
$ java Main < test.in | diff - test6.out
```

Make a copy of your Java programs to the level directory by typing the Unix commands

```
$ mkdir sim6
$ cp *.java sim6
```

Submission (Course)

Select course: CS2030 (2018/2019 Sem 2) - Programming Methodology II ▼

Your Files: BROWSE

SUBMIT (only .java, .c, .cpp, .h, and .py extensions allowed)

To submit multiple files, click on the Browse button, then select one or more files. The selected file(s) will be added to the upload queue. You can repeat this step to add more files. Check that you have all the files needed for your submission. Then click on the Submit button to upload your submission.