CPSC 531 Fall 2017
Assignment 4
Stephanie Ross

Compiling and running

The simulation is run from the class `ICT.java`. The program takes in 4 parameters:

```
$ java ICT numOfElevators schedulingAlgorithm idlingPolicy lambda

numOfElevators: an integer in [1,3]

schedulingAlgorithm: fcfs or sstf

idlingPolicy: stay or bottom

lambda: arrivals per minute
```

Example:

```
$ javac ICT.java

$ java ICT 1 sstf stay 0.5
```

In addition to the statistics that are printed to standard output at the end of the simulation, the program also writes the response times to a file called response.txt, which can then be put into Excel to make a histogram.

The program was tested on the Linux machines in the CPSC labs. The program is fully functional.

Notes

I chose next-event simulation and discrete-time because I thought they were the most appropriate method for the situation. Updating the system state variables (the locations of everyone in the building as well as the locations of all the elevators) one event at a time seemed easier than jumping over fixed time blocks and examining everything that happened during that period. This allowed me to work with a single elevator (or a single person) on a single floor at a time.

CPSC 531 Fall 2017
Assignment 4
Stephanie Ross

Assumptions

- All elevators are initially on floor 1
- A person arrival occurs when someone approaches the elevator area on a floor. On floor 1, this is when a person approaches the elevator area having just entered the building from outside. On any other floor, this is when a person approaches the elevator area after having worked on that floor for some amount of time.
- If a person approaches the elevator area (on any floor) and nobody else is there, they instantly put in a request to summon an elevator (so the timestamp of the request is the same as the person's arrival time).
- If a person approaches the elevator area (on any floor) and there is already somebody waiting, the person who just arrived doesn't need to put in their own request. I chose to model the system like this because in the real world, when you approach an elevator and there is already someone there, the button is already lit up, so pressing it again is unnecessary. Any subsequent presses of the button do not make a difference; however, since the response time is based on the user's perception, the wait + trip time for every passenger still gets recorded when they are dropped off even if it is possible that the passenger wasn't the person who first pressed the elevator button. In the real world, when someone is lucky enough to arrive at the elevators just as the doors are opening to let people in, their perceived response time (wait + trip) would be shorter than the people who had already been standing there for a while, even if they all travel to the same floor.
- Requests are not interruptible. Once an elevator is on its way to pick someone up on a floor, it will not stop at any other floor to pick anyone else up along the way. Even though elevators in the real world would pick people up on the way if they are going in the same direction, I made this assumption to simplify the model so that I wouldn't have to make the elevator stop at every floor on its journey to check for pending requests. Also, one of the scheduling algorithms I tested is FCFS, and if an elevator moving down would pick up requests along the way this might violate FCFS.
- Similar to the previous assumption, once an elevator has picked up passengers, it does not stop to check for pending requests until the elevator is empty. For example, if the elevator picks up a passenger on floor 5 and is bringing them down to floor 1, and on the way a request is made by a person on floor 2 to go down to floor 1, the elevator will pass floor 2 without picking them up.
- If an elevator has dropped off its last passenger and is now empty, it checks for pending requests. If there are none, it goes to its idling location interrupted. When it gets there, it checks if any pending requests have been made. If there are still none, then the elevator becomes idle.
- For $N > 1$, if there is a pending request and more than 1 elevator is idle, an elevator will be chosen at random to service the request.
- The number of requests served by a server is the number of times an elevator opened its doors to let people off. So, if an elevator is carrying 10 people and is traveling up, lets 5 people off on floor 3 and 7 people off at floor 5, 12 user-perceived response times are recorded, but the number of requests serviced by the elevator only increases by 2. Technically the elevator served 12 people, but since everyone who needs to get off on a given floor gets off at the same time, the elevator only had to perform 2 drop off events to service everybody.

Results

The following table outlines the average user-perceived response time for each combination of scheduling algorithm and idling policy when N = 1 and lambda = 0.5 arrivals/minute.

	Stay	Bottom
FCFS	101.979 seconds (1.70 minutes)	106.118 seconds (1.77 minutes)
SSTF	91.255 seconds (1.52 minutes)	92.949 seconds (1.55 minutes)

Using SSTF instead of FCFS as the scheduling algorithm resulted in faster user-perceived response times. This was expected, since one of the problems with FCFS is that there is often an alternate way to service the requests that would result in more jobs being finished in the same amount of time.

When consecutive requests were far apart in distance, it resulted in a lot of unnecessary back and forth traveling. More requests could have been serviced in the same amount of time if they had been picked up in a different order. For example, if the elevator was on floor 6 and there was an "up" request coming from floor 1 followed by a "down" request from floor 5, FCFS would dictate that the elevator goes to pick up the person on floor 1 first. Then the request on floor 5 wouldn't be picked up until the elevator finished dropping off the person it had picked up on floor 1. It would have been more efficient to pick up the person on floor 5 first, then bring them down to floor 1, then the person on floor 1 could have immediately gotten on and traveled to floor 5. That way, 2 requests would have been serviced in that time instead of just 1. The user-perceived response time for the person on floor 1 would have been the same, but the user-perceived response time for the person on floor 5 would have been greatly improved. This is why SSTF is the better choice for this simulation.

Using "stay" instead of "bottom" as the idling policy resulted in faster user-perceived response times. This is particularly noticeable when FCFS was used as a scheduling algorithm. This is probably because I made the decision to have the elevator move to its idling location uninterrupted, so the extra trip to the bottom floor without servicing any down requests (and then the trip back up to where the next request is) results in several extra trips where the elevator is empty when it could have been serving a request.

(More detailed experiment results with increased # of elevators on the next page)

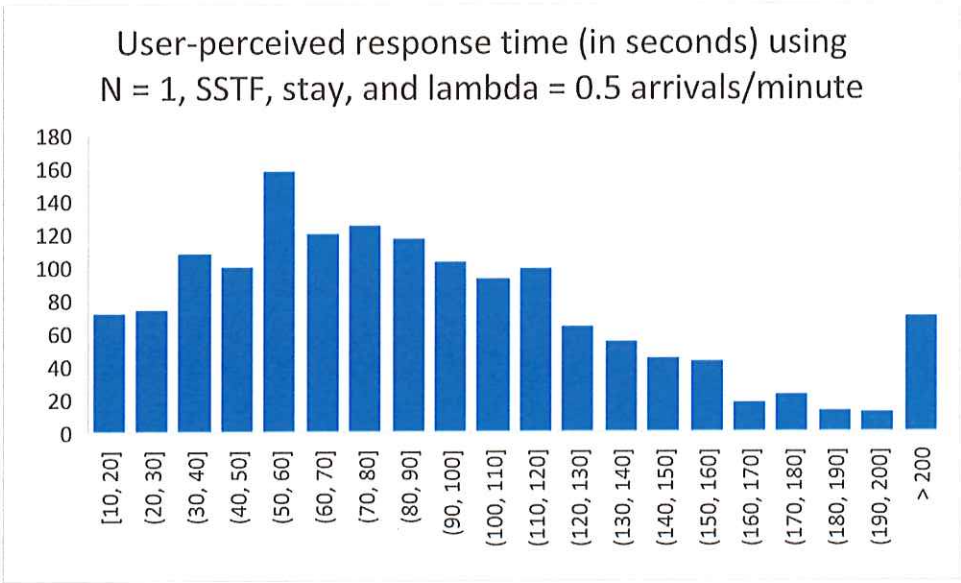
N = 1 with scheduling algorithm = SSTF and idling policy = stay and lambda = 0.5 arrivals/minute

```
Requests served by elevator: 1397

Average user-perceived response time: 91.255 seconds

Average user-perceived response time-- going up:
  floor 1 ---> floor 2:    52.413 seconds (135 users)
  floor 1 ---> floor 3:    60.051 seconds (121 users)
  floor 1 ---> floor 4:    66.842 seconds (142 users)
  floor 1 ---> floor 5:    79.861 seconds (127 users)
  floor 1 ---> floor 6:    92.516 seconds (127 users)
  floor 1 ---> floor 7:    98.311 seconds (123 users)

Average user-perceived response time-- going down:
  floor 2 ---> floor 1:    90.790 seconds (125 users)
  floor 3 ---> floor 1:    82.917 seconds (118 users)
  floor 4 ---> floor 1:    103.606 seconds (133 users)
  floor 5 ---> floor 1:    109.854 seconds (123 users)
  floor 6 ---> floor 1:    125.233 seconds (118 users)
  floor 7 ---> floor 1:    141.317 seconds (120 users)
```



(Note: All values > 200.0 seconds are in the overflow bin)

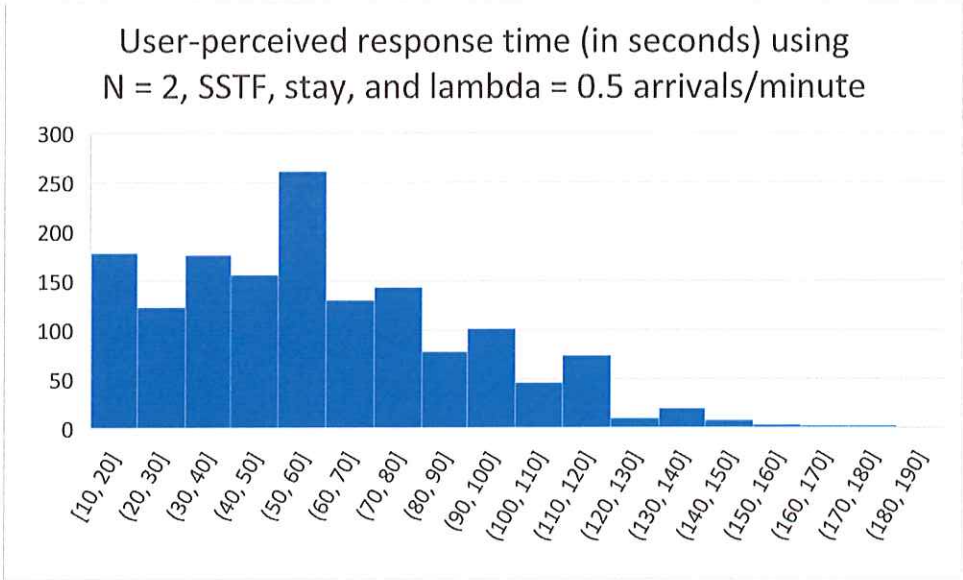
N = 2 with scheduling algorithm = SSTF and idling policy = stay and lambda = 0.5 arrivals/minute

```
Requests served by elevator #1: 734
Requests served by elevator #2: 725

Average user-perceived response time: 61.695 seconds

Average user-perceived response time-- going up:
  floor 1 ---> floor 2:    32.954 seconds (135 users)
  floor 1 ---> floor 3:    38.080 seconds (121 users)
  floor 1 ---> floor 4:    50.943 seconds (142 users)
  floor 1 ---> floor 5:    65.310 seconds (127 users)
  floor 1 ---> floor 6:    71.964 seconds (127 users)
  floor 1 ---> floor 7:    81.329 seconds (123 users)

Average user-perceived response time-- going down:
  floor 2 ---> floor 1:    34.951 seconds (125 users)
  floor 3 ---> floor 1:    43.855 seconds (118 users)
  floor 4 ---> floor 1:    58.680 seconds (133 users)
  floor 5 ---> floor 1:    74.424 seconds (123 users)
  floor 6 ---> floor 1:    89.458 seconds (118 users)
  floor 7 ---> floor 1:    104.137 seconds (120 users)
```



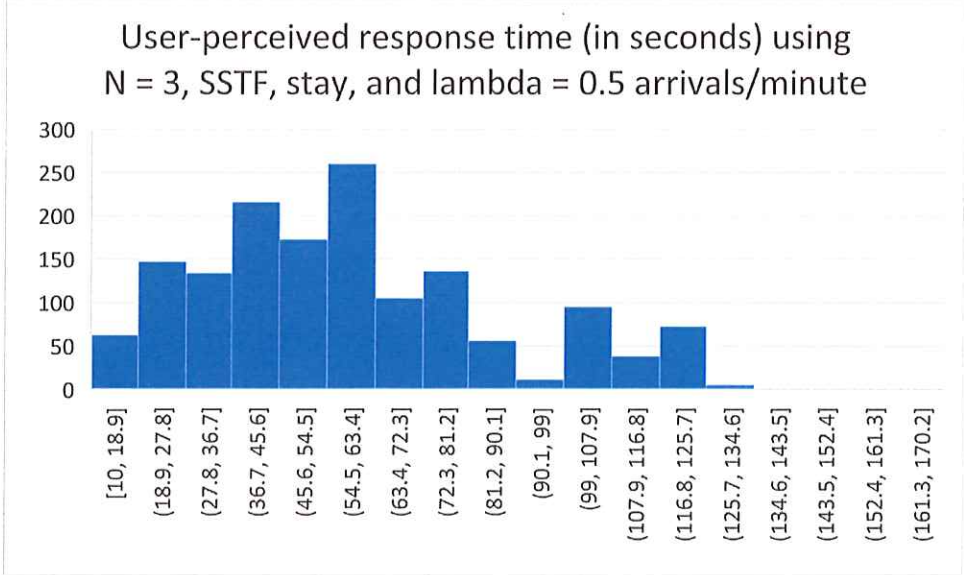
N = 3 with scheduling algorithm = SSTF and idling policy = stay and lambda = 0.5 arrivals/minute

```
Requests served by elevator #1: 479
Requests served by elevator #2: 502
Requests served by elevator #3: 483

Average user-perceived response time: 58.216 seconds

Average user-perceived response time-- going up:
  floor 1 ---> floor 2:    31.180 seconds (135 users)
  floor 1 ---> floor 3:    37.654 seconds (121 users)
  floor 1 ---> floor 4:    48.891 seconds (142 users)
  floor 1 ---> floor 5:    57.111 seconds (127 users)
  floor 1 ---> floor 6:    68.434 seconds (127 users)
  floor 1 ---> floor 7:    80.354 seconds (123 users)

Average user-perceived response time-- going down:
  floor 2 ---> floor 1:    31.444 seconds (125 users)
  floor 3 ---> floor 1:    39.965 seconds (118 users)
  floor 4 ---> floor 1:    52.518 seconds (133 users)
  floor 5 ---> floor 1:    68.326 seconds (123 users)
  floor 6 ---> floor 1:    85.428 seconds (118 users)
  floor 7 ---> floor 1:    103.097 seconds (120 users)
```



The decision to have the elevator not stop to pick up any passengers once it had a load was probably the most impactful assumption I made. I made the decision to model it this way because it simplified the model. The elevator wouldn't have to stop and check if anyone needs to get on at every floor on its journey. This didn't have an impact on the upward trips, because nobody on floors 2-6 is going up. This assumption negatively impacted the downward trips, because the elevators were likely passing floors where people also wanted to go down, and did not pick them up even though it would have been more efficient to do so. This resulted in extra trips to go back up and pick up passengers that it had previously passed.

Increasing the number of elevators from N = 1 to N = 2 decreased the average user-perceived response time by 30 seconds. The improvement was expected, since increasing the number of servers in a system lessens the load of each server. Multiple users can then be serviced simultaneously, resulting in shorter wait times. This is particularly useful for this simulation when there can be requests in opposite directions. Also, if someone arrived at the elevators shortly after one had just left, they likely wouldn't have had to wait as long for the next one to arrive (with N = 1, they would have had to wait for the elevator they had just missed to come back around, which might have taken longer).

Increasing the number of elevators from N = 2 to N = 3 further decreased the average user-perceived response time, but the improvement was small when lambda = 0.5. The user-perceived response time went from to 61.695 seconds with N = 2 to 58.216 seconds for N = 3. That is only an improvement of 3.479 seconds. This is probably because the load never became heavy enough for the third elevator to make a difference, so the user-perceived response time with 3 elevators was not that much better than with 2 elevators. Increasing lambda to examine the elevators at a higher load (e.g. lambda = 5 arrivals/minute) made the benefits of the third elevator more noticeable.

(More experiment results with increased lambda on the next page)

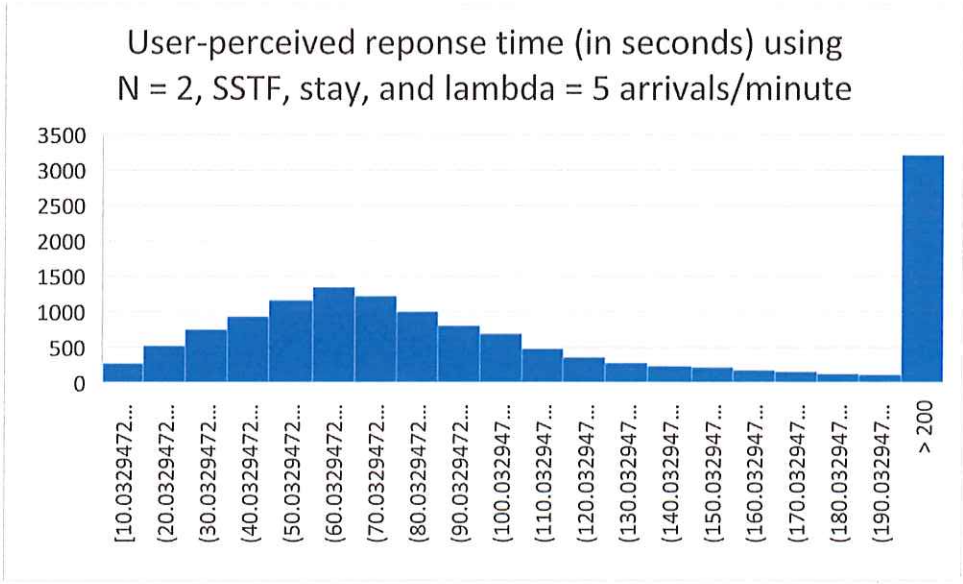
N = 2 with scheduling algorithm = SSTF and idling policy = stay and lambda = 5 arrivals/minute

```
Requests served by elevator #1: 3408
Requests served by elevator #2: 3346

Average user-perceived response time: 195.188 seconds

Average user-perceived response time-- going up:
  floor 1 ---> floor 2:    40.194 seconds (1203 users)
  floor 1 ---> floor 3:    50.148 seconds (1163 users)
  floor 1 ---> floor 4:    60.081 seconds (1236 users)
  floor 1 ---> floor 5:    71.120 seconds (1216 users)
  floor 1 ---> floor 6:    81.026 seconds (1209 users)
  floor 1 ---> floor 7:    90.638 seconds (1183 users)

Average user-perceived response time-- going down:
  floor 2 ---> floor 1:    431.224 seconds (1146 users)
  floor 3 ---> floor 1:    595.574 seconds (1103 users)
  floor 4 ---> floor 1:    474.304 seconds (1176 users)
  floor 5 ---> floor 1:    220.364 seconds (1164 users)
  floor 6 ---> floor 1:    146.083 seconds (1169 users)
  floor 7 ---> floor 1:    126.550 seconds (1130 users)
```



(Note: All values > 200.0 seconds are in the overflow bin)

At a heavy load, the most frequent user-perceived response times were over 200.0 seconds!

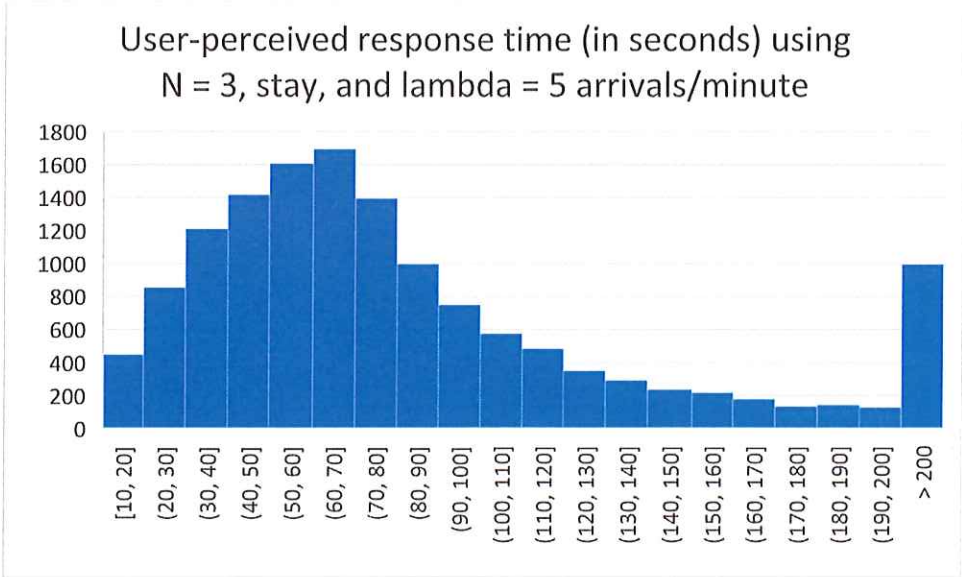
N = 3 with scheduling algorithm = SSTF and idling policy = stay and lambda = 5 arrivals/minute

```
Requests served by elevator #1: 2857
Requests served by elevator #2: 2876
Requests served by elevator #3: 2900

Average user-perceived response time: 88.582 seconds

Average user-perceived response time-- going up:
  floor 1 ---> floor 2:    30.939 seconds (1202 users)
  floor 1 ---> floor 3:    40.252 seconds (1163 users)
  floor 1 ---> floor 4:    49.612 seconds (1237 users)
  floor 1 ---> floor 5:    59.726 seconds (1216 users)
  floor 1 ---> floor 6:    69.996 seconds (1209 users)
  floor 1 ---> floor 7:    80.551 seconds (1183 users)

Average user-perceived response time-- going down:
  floor 2 ---> floor 1:    106.206 seconds (1146 users)
  floor 3 ---> floor 1:    141.086 seconds (1103 users)
  floor 4 ---> floor 1:    144.586 seconds (1183 users)
  floor 5 ---> floor 1:    124.607 seconds (1165 users)
  floor 6 ---> floor 1:    109.377 seconds (1169 users)
  floor 7 ---> floor 1:    115.240 seconds (1130 users)
```



(Note: All values > 200.0 seconds are in the overflow bin)

At a heavier load, now we can see the difference that a third elevator makes. By adding a third elevator, the user-perceived response time improved from 195.168 seconds to 88.582 seconds—a decrease of over 50%! The number of people who had to wait over 200.0 seconds for an elevator massively decreased as well.