Questions

How many elevators will be involved?

- 1 to start, 2 to give another perspective with time

Will there be a capacity?

- Yes, 8 people limit

How many floors will be in the building?

- 8 floors

Can the amount of floors and capacity be changeable?

- Yes. This is only possible at the start of running where they are defined from the text file and then stay constant

What happens if an elevator is full?

- It will hold off on that passenger and come back for it after taking care of those on elevator currently
- How does it know to come back for that passenger?
    - The request remains in the system's pickup queue
    - Would it not take more time for other passengers that are arriving or have been waiting?
        - It will stop and pick up those passengers on the way to the original passed passenger
        - What if the elevator is at max capacity again when it makes its way to that passenger?
            - The elevator could follow a scheduling rule such as oldest unserved request gets priority

        - Does that mean it will skip floors that passengers need to get off of to prioritize this one individual to get on?
        - Well, no we would want them to get off on their respective floors
        - But you cannot prevent other passengers waiting on those floors to not get on? They will still load themselves on as other passengers are trying to unload?
            - Then let us wrap back to the original algorithm of coming back for passengers as soon as possible. Let us establish another elevator to mitigate this situation.

How will you simulate the feeling of an elevator?

- Adding wait times for going from floor to floor, opening and closing doors.
- Additionally, I want to add arrival times for the passengers.

How will you handle requests with 1 or 2+ elevators?

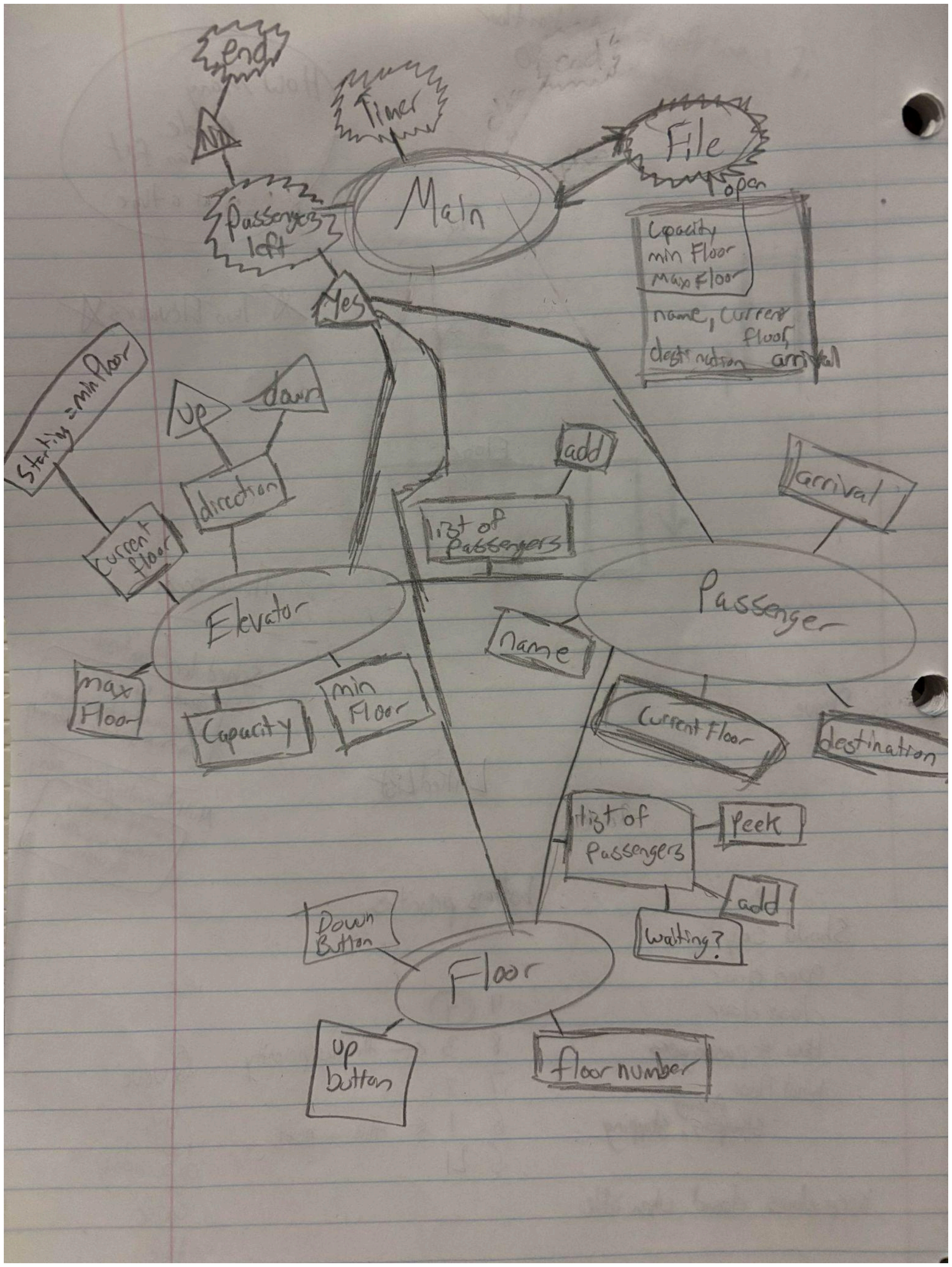
- With 1 elevator, I will start by having the elevator just go up and down
    - I will want the elevator to rest if not needed, so implementing a way for it to be idle.
    - I want to optimize the time it takes to pick up passengers, so I will have it change directions if nobody has a request above/below to pick up a request. If it does have a request in that direction it is already traveling, it will continue in that direction.
- With 2 elevators, I want to try and save time by having one elevator possibly look for passengers going up and the other looking for passengers going down based on button press.

- Elevators could communicate if one is being overloaded or underutilized as well.

Ideas

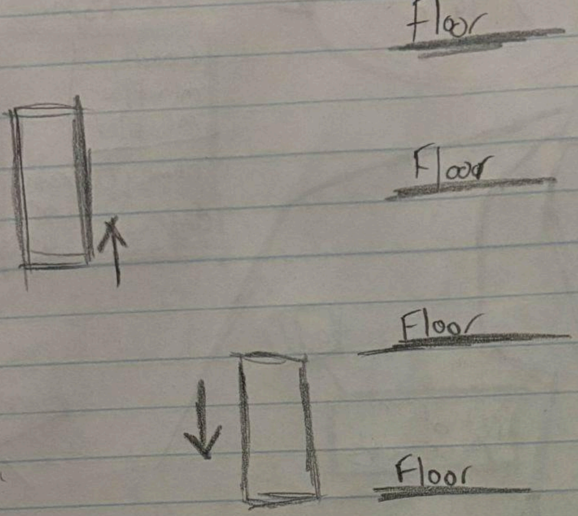
- Instead of updating the system every round, passengers could announce when they press the button in real time. This would make the simulation more dynamic and could also help organize the flow of requests.
- Implement logic to prevent the elevator from traveling all the way to the top if no one has requested a ride above. This would save time and energy.
- Add custom buttons for specific floors, such as "Lower Floor 1" or "Lobby," to make the elevator system more flexible and user-friendly.
- Make the elevator more complex by adding dual doors that open from the left and right. Passengers could then choose which side to exit from, depending on whether they're going up or down.
- Find a way for multiple elevators to communicate with each other, instead of having one focus solely on up calls and the other on down calls. This would improve efficiency in handling requests and prevent idle elevators.
- Implement priority queues for emergencies or elderly passengers, which could push them to the front of the queue. This might require a third elevator dedicated to handling urgent requests.
- Have the elevator speed adjust based on factors like the number of passengers or time of day. For example, during off-peak hours, the elevator could move more quickly if fewer passengers are inside.

Designs


end
No
Timer
File
Main
open
Capacity
min Floor
Max Floor
name, current floor, destination arrival
Passengers left
Yes
Starting = min Floor
up
down
direction
current floor
add
list of Passengers
arrival
Elevator
Passenger
name
max Floor
Capacity
min Floor
Current Floor
destination
list of Passengers
Peek
add
Waiting?
Down Button
Floor
up button
floor number

if current floor = destination floor
destination floor = 0

How many people can fit at a time



☆ Two Elevators ☆

Things needed

Scheduler/controller
- elevator requests,
- assign elevators to passengers
- optimize elevator movement

Multithreading/Concurrency
- multiple elevators
- handle concurrent requests

Deque?

LinkedList

buttons priorities

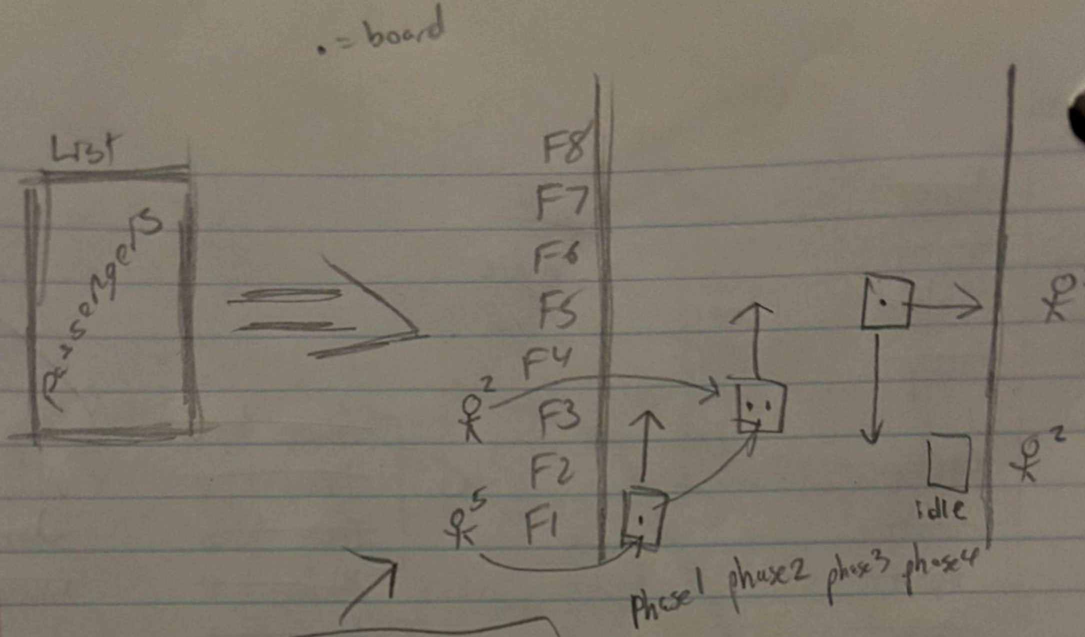
Should say
- open door
- close door
- How # passengers
- What Floor
  - ~~skipping~~ passing, stopping

Keep doors closed when idle

9 (4)
8 3 ← this has priority
7 2
6 1 ← this is next
5 L1

Queue by descending order Here

Capacity = 8
Min Floor = 1

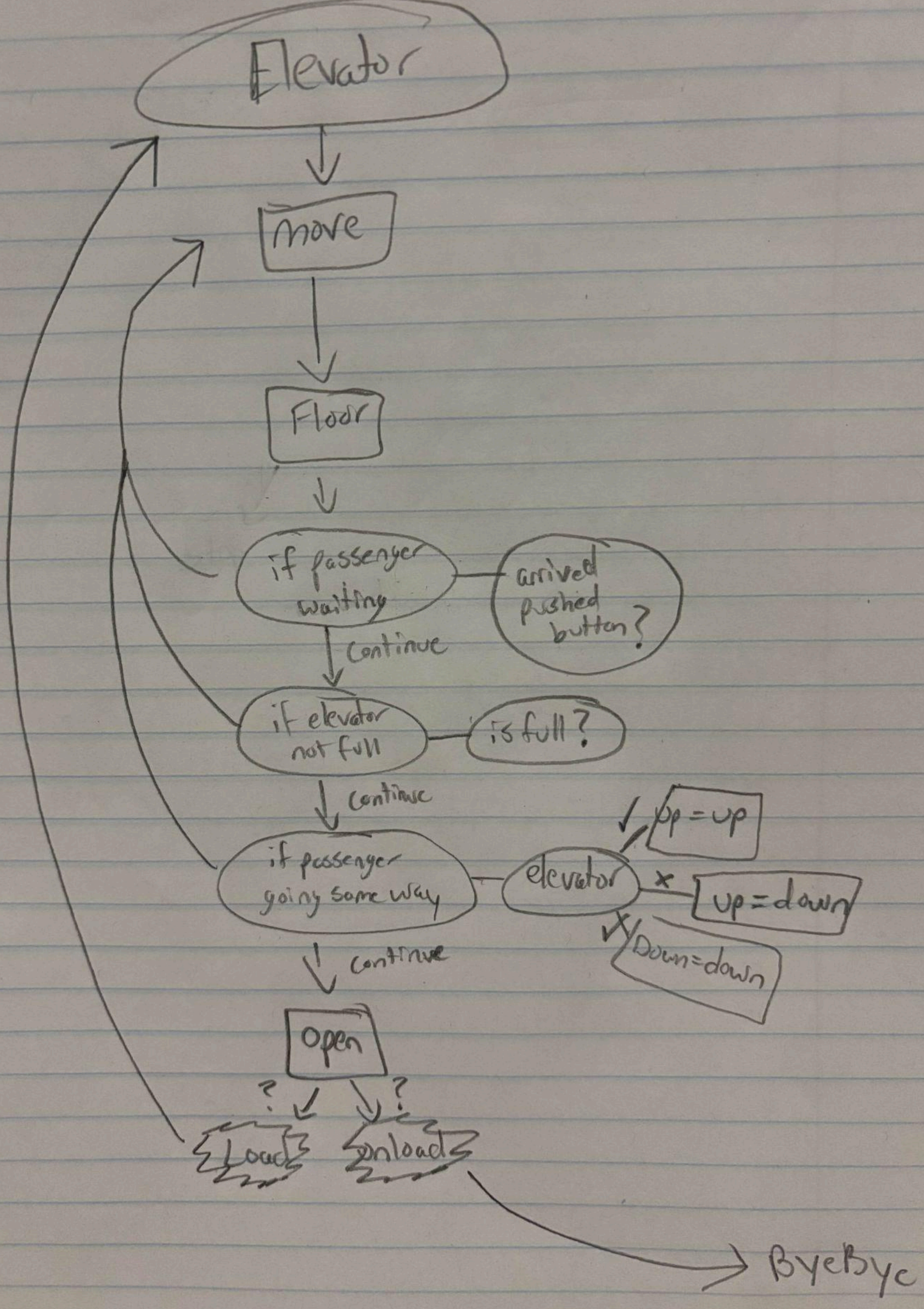


if 3rd one appears at F7 after pickup
what to do?
2

Go to F2, then prioritize back up for F7
pick up any along the way

2nd elevator is communicated to by 1st elevator

Should we have 2 elevators where one focuses on residents going down and the other focuses on residents going up?


Elevator
move
Floor
if passenger waiting
arrived pushed button?
continue
if elevator not full
is full?
continue
if passenger going same way
elevator
✓ up = up
× up = down
✓ Down = down
continue
Open
?
?
Load
Unload
ByeBye

Things I Learned

Ternary operator:

- ? is like an if-else expression
- condition ? valueIfTrue : valueIfFalse

AtomicBoolean

- Lets you store a boolean value
- Lets you share/update value across methods
- It is a mutable object

Control Flow with States (ElevatorActionState):

- The ElevatorActionState class is used to manage the elevator's actions at different stages
- Being able to handle transitions between different states systematically

Final Notes/Thoughts

This coding challenge has been a great learning experience. I started out with a well-organized approach, but I eventually got too comfortable and ended up writing everything in the main method. Looking back, I realize it would have saved me time and effort to build methods early on rather than refactoring later in development. One area I did not fully explore was threading. I encountered it when considering how to manage multiple elevators at once. I believe using multi-threading would have allowed the elevator to move and process passenger requests more efficiently than with a single elevator. I would like to spend more time exploring how to make two elevators work simultaneously in my free time. Additionally, I am interested in adding a front-end to the project so I can visually show the elevators moving. I see it as a fun way to bring the simulation to life. Moving forward, I plan to focus on keeping my code organized and modular and seeing if I can learn some new features to optimize this project.

Resources Used

Understanding Elevator Algorithm

https://youtu.be/xOayymoIl8U?si=OyBzwtHUOcRL_nBm

Ways to Look at the Elevator Algorithms

Elevator Scheduling Algorithms: FCFS, SSTF, SCAN, and LOOK - DEV Community

Example 1: Elevator

https://youtu.be/FptCbX7fRHw?si=6uaONFTpMhV_eHqm

Example 2: Elevator

GitHub - PyAntony/java-elevator-simulation: Elevator simulation

Example 3: Elevator

GitHub - swapniltake1/Elevator: This is a Java-based elevator simulator program that demonstrates the functionality of a single-threaded elevator system. The program is designed using core Java concepts, including object-oriented principles and the collection framework. The simulator allows users to call the elevator, board passengers, and move between floors.

Understanding Deque

https://youtu.be/gXZt4P97UW4?si=jGidLF31RjCeLcGQ

Deque Interface in Java - GeeksforGeeks