Davis Lewis, Rashminder Randhawa, and Anastasia Reder
Professor Han
CS 2028C
April 20th, 2021

Homework 3: Elevator Simulator

Group and Contributions of Each Member:
Anastasia Reder – Programmer, Report Writer
David Lewis – Main Programmer, Report Writer
Rashminder Randhawa – Main Programmer

Part A:
Object oriented analysis (OOA):

Object Oriented Design (OOD):

Problem Statement:

Build an elevator simulator to measure its performance utilizing classes in C++.

Assumptions and Requirements:

We know that there should be six main classes based on some of the background information given in the assignment. The six classes as follows: elevator, building, floor, door, clock, person, button, and scheduler. Elevator cannot exceed the floor limits such as 1-10 or 3-8 as in the background given to us. Students, teachers, staff, and the general public will call the elevator from different floors to enter and, once on the elevator, select their desired floor to exit from. We will need to create a set of classes to simulate the movement of an elevator. The elevator will go in one direction until all passengers have exited or the elevator has reached the top/bottom floor where it then can be called in the opposite direction. There can be more than one person entering/exiting the elevator. The queue for people on the elevator adapts when people enter or exit from the elevator (priority queue).

Rationale:

Elevator class:

We determined that it should be able to know the “capacity”: the number of people that it should be able to contain. The elevator, as we all know, operates going up or down, which means the class must know the direction that it needs to travel. The class should also know what floor it is on as well as what floor it needs to reach. The elevator class should know if the button of a floor was pressed so it can know in which direction to travel. This starts the movement of

the elevator, and the class should be able to ascertain whether the door is open or closed so the people in this simulation are not trying to walk through closed doors. We know the door, button, and the floor parts can be determined from the three classes we labeled as door, button, and floor respectively. We also needed a queue system for the elevator because if someone on the elevator (on floor 8) presses 1st, 4th, 3rd, and the 2nd floors, the elevator is not going to go in that order. Thus, we needed a queue system to be able to sort them in order of descending or ascending order, and be able to stop at the floors being called if it is “on the way” towards its final destination.

Building class:

The building class we determined needed to know the number of floors that exist, such as the floors from 1-100, or 3-8 so that the elevator simulator does not exceed the floor limits. The building would also be able to know the schedule such as when people arrive and student class times. The building class is also aware of the elevator since the elevator exists inside a building.

Floor class:

This class needs to know when a button is pressed on a floor to call the elevator. The floor class needs to house the height of each floor which we used to calculate the time elapsed between floors. The last part of this class needs to know the number of people wanting to get on/off on each floor since the building will have people entering and exiting on the same or different floors.

Door class:

We initially thought it would be useful to know the amount of time required for opening the door so we could measure it later, but we later opted for a single unit of time to determine the elevator traveling from one floor to the next without measuring the individual time it would take for each task of opening/closing the door and the elevator raising between levels. Thus, the final class was used as a way to make sure the door was open for the people to enter or get off on the desired floor.

Clock class:

This class needs a counter and a way to track the progression of time since the elevator would be operating 24 hours a day. This is also connected to the scheduler as a way to make sure time is being tracked throughout the day. We need the class to be able to advance time quickly because having the simulator operate at regular time intervals makes observing the simulation too long to be useful.

Person class:

This class is important for the setup of the people getting on and off the elevator because the “person” would have a floor they are entering from and a target floor they are trying to reach.

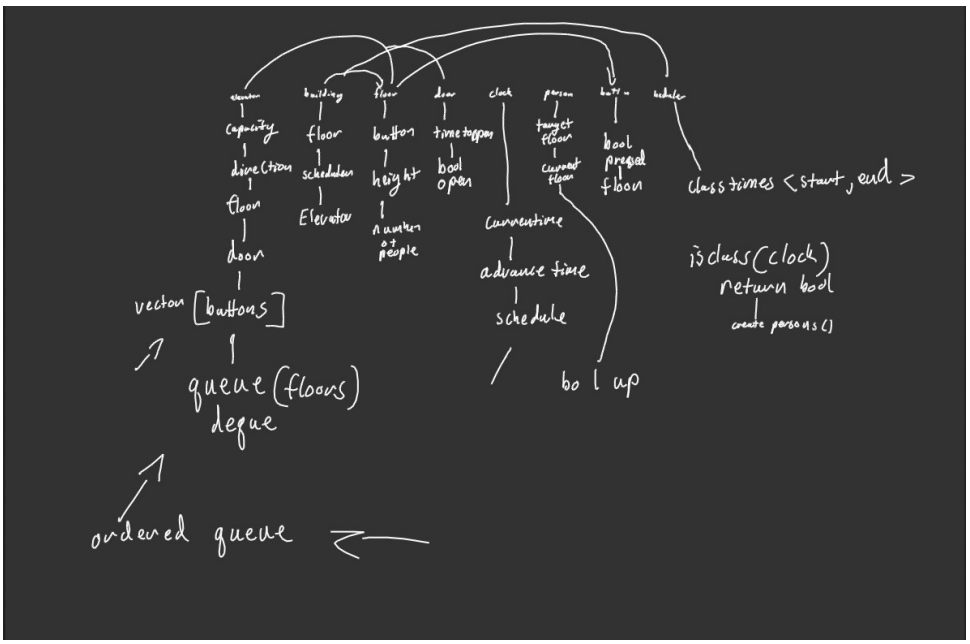
Button class:

This class confirms the “button” was pressed, which in the real world would signify to the elevator to raise or lower to the correct floor.

Scheduler class:

It is important that the class has access to the class times and when the classes would take place (or not). For example, when a class is in session, the elevators are not used as much, but when the class is over, you have students

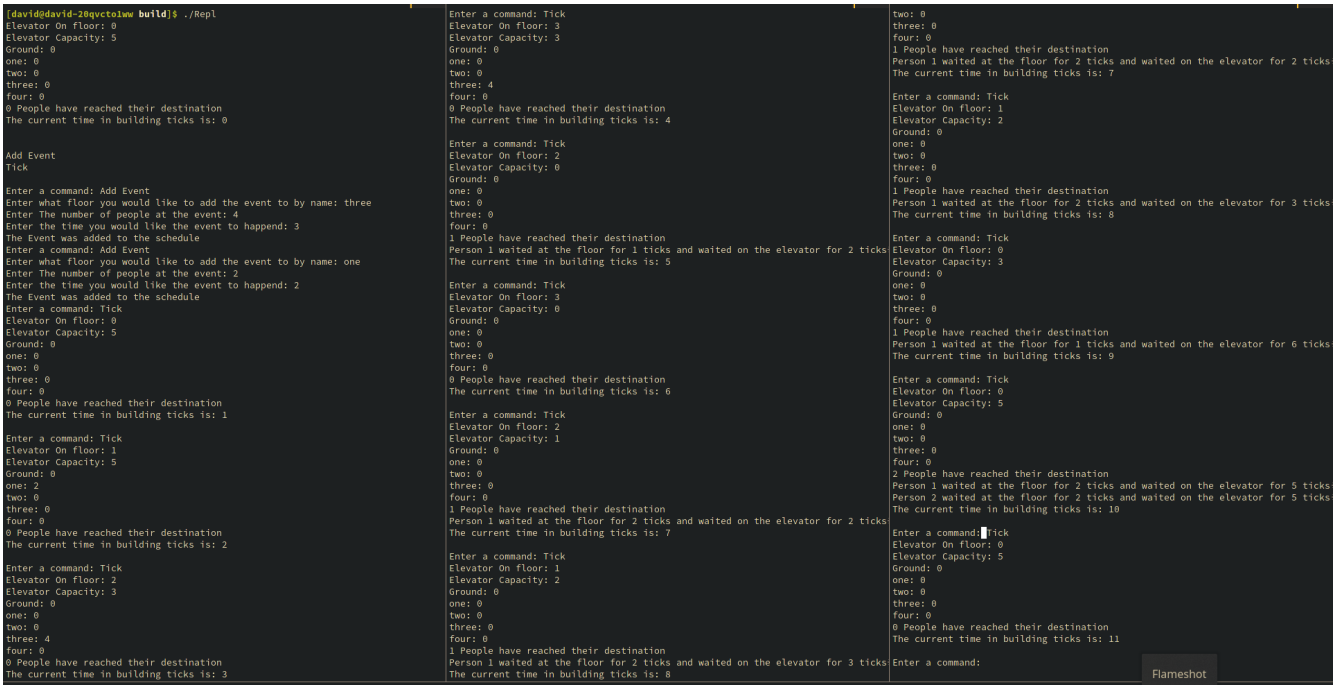
that want to use the elevator to get to a different floor, whether they are going to their next class or heading home for the day.



Part B:

Object Oriented Programming:

Compiling Instructions:
Make sure GNU make is installed. In the root directory (hw03), run the shell command “make -k”.
Executing Instructions:
In the build directory, run the Repl binary with ./Repl. There are two commands (Add Event : adds an entry in the scheduler that will spawn people at specific times) and (Tick : advances the building time by 1 unit of time). Use the ScreenShot as an example.
ScreenShot:



Submission:
Submit a report in two parts, Part A and B. Part A includes the OOA and OOD -- the problem statement, the requirements, the assumptions made, and the rationale why and how your classes are derived.
Part B is OOP which should include sample results printout and instructions for compiling and running the programs, the source code files and any required data files.

This homework can be done in group. Make sure to include the name of all group members (up to four members), and what each member contributed.