1. Main idea

- 현존하는 엘리베이터들의 현실적인 불편함 해소

- 엘리베이터 기다리는 시간을 줄임.

2. Function (상세하게)

\* Algorithm (전처럼 그림으로 보여주고 뒤에 시뮬레이션으로 다시 보여줌)

- Circular Elevator: 엘리베이터 3대가 일정 간격을 두고 한 경로, 한 방향으로 돌아가게 함으로써 엘리베이터가 특정 층에 올라갔다가 다른 탑승객을 태우기 위해 다시 내려오는 시간을 줄여 전체 waiting시간을 줄임

\* 층 별로 올라가는 passenger와 내려가는 passenger를 위한 circular queue를 두 개를 두어 passenger가 생성되면 queue에 enqueue된다.

\* passenger가 queue에 들어가면 elevator의 state가 run으로 바뀌어 엘리베이터가 운행하게 된다.

\* elevator가 운행하면서 floor별로 queue에 passenger가 있는지 확인하고 있다면 queue에서dequeue되고 elevator의 list에 도착 층에 가까운 순서로 insert된다. (올라갈 땐 increasing order / 내려갈 땐 decreasing order)

\* elevator가 다시 운행하면서 도착 층에 도착하면 list에서 passenger를 delete한 후 엘리베이터들의 현재 state를 확인해 계속 운행을 할지 멈출지 결정한다.

- full elevator: elevator의 maximum capacity를 넘는다면 capacity의 여유가 생기기 전까지passenger를 태우지 않음

\* elevator가 돌면서 floor별 queue에 있는 승객을 태울 때 승객의 weight를 확인하고 현재 elevator weight와 비교해 태울지 안태울지 확인한다.

- Estimated arrival time: 엘리베이터 버튼을 내부가 아닌 층 별로 외부에 두어 passenger가 가고자 하는 층의 버튼을 누르면 도착 예정 시간을 보여줌

\* passenger가 생성된 후 생성된 passenger 위치에서 3대의 엘리베이터의 각도를 계산해 가장 빨리 오는 엘리베이터를 찾는다.

\* 해당 엘리베이터가 passenger 위치에 도착하는 시간과 passenger를 태워 목적 층까지 도착하는 시간을 결정해 총 도착 예정 시간을 계산한다.

\* UI Programming **(빨간 것들만 요약해서 발표하고 나머지는 필요할 것 같으면 언급하기)**

GUI: 버튼으로 엘리베이터를 호출하고 엘리베이터가 동작하는 모습을 그림으로써 알고리즘을 직관적인 이해를 돕기 위해 GUI를 선택했다.

연결: 두 개의 프로그램을 연결하기 위해 네트워크를 사용함.

(command [argument1] [argument2] …)

1) Server to Client

- 엘리베이터 상황: elevator [id] [icon\_id] [current\_floor] [horizontal\_position] [elevator\_state]

: 엘리베이터를 그리기 위해 현재 위치와 상태, 이미지 번호를 송신한다.

- 남은 시간: remaining\_time [minutes] [seconds]

: 남은 시간을 표시하기 위해 분, 초 단위 시간을 송신한다.

- 탑승: getin [source\_floor] [destination\_floor]

: 탑승하는 상황을 그리고 호출 버튼을 해제하기위해 출발 층과 도착 층을 송신한다.

- 하차: getout [destination\_floor]

: 하차하는 상황을 그리기 위해 도착 층을 송신한다.

2) Client to Server

- 엘리베이터 호출: NONE [source\_floor] [destination\_floor]

: 엘리베이터를 호출하기 위해 출발 층과 도착 층을 송신한다.

3. Simulation (상황 별 simulation 보여줌)

- running elevator + estimated arrival time

- full elevator

4. Role

알아서