2root2: Clive Johnston and Caleb Smith-Salzberg (Period 5)

Smarty Elevator

Idea: We want to code a smart elevator that maximizes efficiency, by packing people that are going to the same or a nearby floor into the same elevator. This is in stark contrast to regular patterns of elevator usage, where a user just goes to the first available elevator. Smart elevators are already in use in many locations across the world because of their proven value.

Implementation:
-Passenger class

- int _currFloor
- int _desiredFloor
- Int _VIP

-Elevator class

- int _capacity
- int _speed
- PriorityQueue<Passenger> _route
    - Will hold passengers, and will go to floors with highest priority, with priority designated by how many Passengers have that as their _desiredFloor. Also prioritize higher-level employees. These fields will be weighted.
- int _time
    - How long it will take for an Elevator to return to Floor0
- ArrayList_zone
    - Contains ints designating which Floors it covers
- Methods that determine when an elevator can depart
    - When an elevator can depart is affected by how many Passengers it currently holds, and how busy it currently is. If the business increases, elevators will wait until they are full before they depart.
    - In addition, there will be 10 floors.
    - (increased business meaning the elevator will wait longer)

-SmartElevator class (driver class)

- Int _business
    - Works as described above
- Methods to facilitate adding Passengers to elevators

User Interaction: We hope to have a processing sketch that will show a depiction of a regular elevator, and one of a smart elevator, side-by-side. The sketch will show dots (Passengers) entering elevators on both sides. It should become apparent to the user that the smart elevator is more efficient than the regular elevator. A user can change the business, and speed of the elevators, and we also hope to add interactions where users can add people and desired floors. This information will be processed and added to the PriorityQueue, and the elevator class will adjust accordingly.

Algorithm for determining which Elevator a Passenger is assigned to:
If Passenger (bob) wants to go to Floor5, every _available Elevator will be scanned. An Elevator is considered _available if its _time <= 6. _time is defined as how long it will take the Elevator to reach Floor0, and it is calculated as soon as the Elevator departs from Floor0

The _available Elevators will be spilt up to cover a _zone of the building. So if there are 3 _available Elevators in a 10 floor building:
One Elevator will cover floors [2, 4]
One Elevator will cover floors [5, 7]
One Elevator will cover floors [8, 10]

If the _zone of the Elevator's have already been defined, because of a previous Passenger, then bob will simply wait for, or enter the Elevator whose _zone contains Floor5.

If the _zones have not yet been defined, the Elevator (ellie) with the smallest _time will be assigned the _zone that covers Floor5. bob will be entered into ellie's _PriorityQueue

Based on how _busy it is, ellie will wait for a certain amount of time. After that time has elapsed, ellie will take off and drop off people at their designated floors, in the order of the _PriorityQueue. The order is defined by _VIP of the Passengers, and how low the Passenger's _desiredFloor is. (ie a Passenger who is going to Floor5, will get dropped off before one going to Floor6).

After dropping of its final Passenger, the Elevator will return to Floor 0, and will become _available.