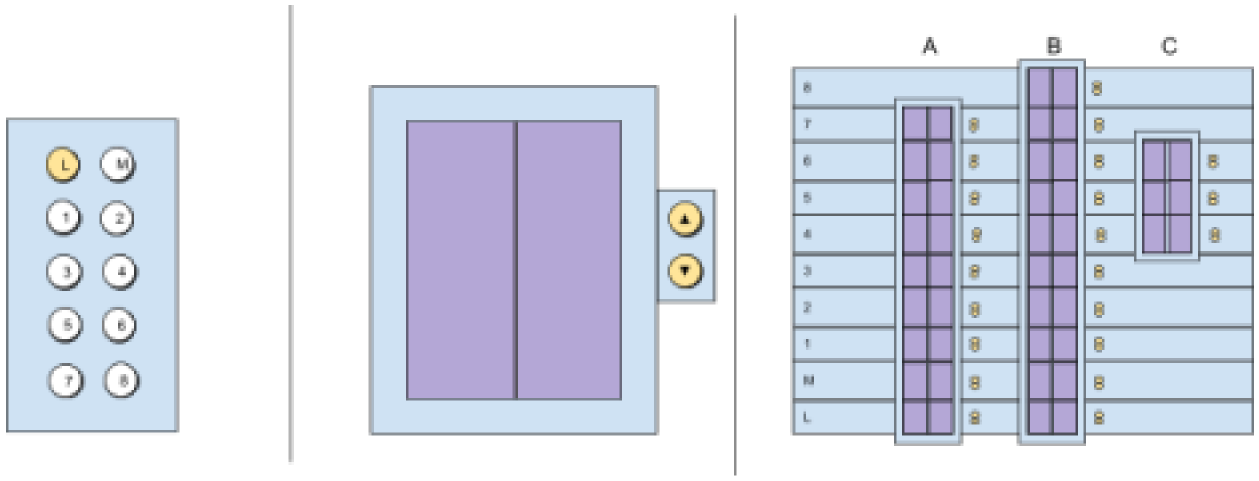
Part 1

You are designing and building the software controller for a new building elevator system that uses cars to move riders from floor to floor.

A single elevator controller is responsible for managing a group of elevator shafts (the vertical space in a building in which the elevator travels) and cars. **Each shaft has a two directional button that allows a rider to call the elevator <u>in that shaft</u>** in a direction by pressing the up or down button.

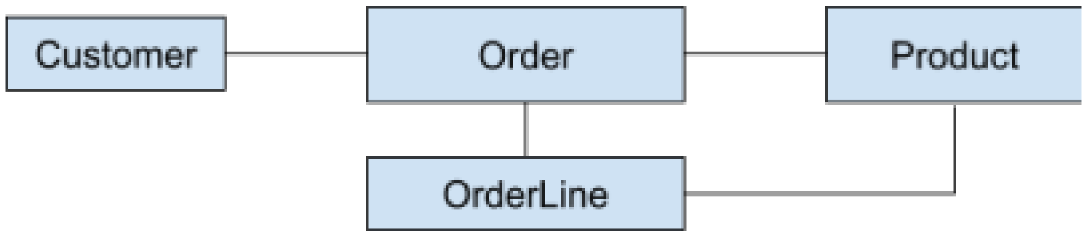
All cars are fitted with a high-precision stepper motor with a smart sensor that can track the location of a car and move it exactly to a specified height at a given speed when instructed by the controller.

Figure 1. Elements



Produce a conceptual class model that describes relationships between object types, no need to assign members at the moment, e.g:

Figure 2: Class model example



Describe the relationship algorithm->model in the simple happy path scenario of someone at L pressing the up button while the elevator is on level 5.

Part 2

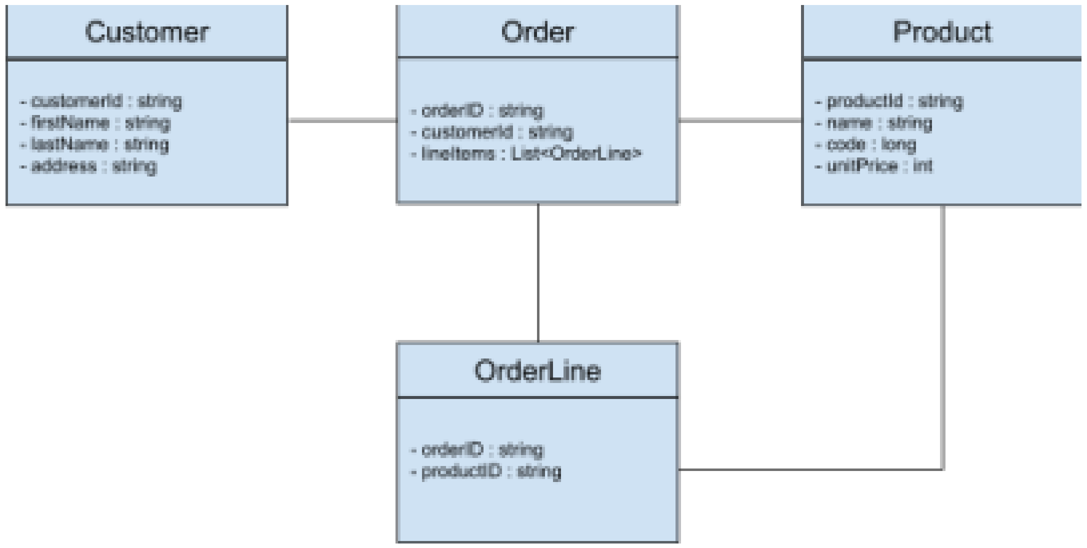
After some testing, our QA have found out the elevators are performing poorly. For example, the elevator starts in the Lobby. Someone press the down button on floor 8 (with the intention to go to Lobby) and right after, someone presses up on floor 1. The elevator will go all the way up to 8, then serve the request and only then go to 1.

At this point, our product specialist has added a new requirement: The system should be able to serve new requests when those requests are compatible with the current state, i.e: in the previous example the elevator should have stopped on 1 on its way down to Lobby.

Create a new version improving the previous model by adding members (including data types) and honoring the requirement described above. One example of such diagram can be found in figure 3.

You can use as data types any construction simple or not, present in any of the most common programming languages: strings, longs, lists, linkedLists, sets, maps, queues, etc.

Figure 3: Class model example with members



Describe the relationship algorithm->model in the simple happy path scenario. The elevator is at L, and someone presses the down button on level 8 (with the intention to go to L). Immediately afterwards, someone else presses the up button on level 1.

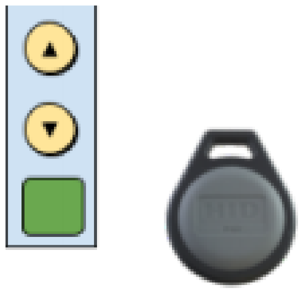
Part 3

Some building-wide services, like the security team, are located in the lobby. This presents some problems when security needs to arrive to a distant floor in rush hours like lunch time.

After some consideration, it has been proposed to have a FOB reader set below the directional buttons. When a given FOB is read and a button is pressed, it would mean a priority request should be served sooner than any non-priority requests with the first elevator available with no passengers. This does not mean that any pre-existing trip will be cancelled to serve this priority request. When a priority request is served, no stops will be made to serve other requests.

Our product designers would like to have multiple configurable priority levels, where for instance: security will have priority 10 and cleaning priority 5.

Figure 4: Floor console with fob reader



Create a new version improving the previous model by adding its members (including data types) and honoring the requirement described previously. One example of such diagram can be found in figure 3.

Describe the relationship algorithm->model in the simple happy path scenario of a security person trying to get to level 5 from L in rush hour.

Extra

The company is doing great, your solutions are being used in many different buildings across the country. You have been promoted to technical leader of the product and as part of the promotion you meet the CPO of the company. He asks you:

Could you briefly describe, in a single line, your top 3 improvement proposals to the system?