Code Structure

Split the application into different projects, each with a specific purpose. Allow for possible future uses for the components.

|  |  |
| --- | --- |
| Project | Description |
| Hub | Main console application. Used to capture people at floors, and also send requests. |
| Entities | Entities that are used in the application. Entities could be in the Domain, however splitting them here means that entities could be used in a separate project where some other aspect involving elevators, requests and floors needs to be used. |
| Domain | Forms the business logic (domain knowledge), determining how the elevators are allocated. |
| Challenge.Tests | Testing key application logic such as moving, requests, and boarding and unboarding of the elevator. |

References: **Hub** -> **Domain** -> **Entities**

Tests can be done on Domain to check the calculations on shafts and elevators.

Entities

The entities project forms the core entities, enumerations and events that are in the system.

1. Floor – track elevators at current floor (list), handle events for elevator arrived
2. Elevator – id (for display), weight limit. Weight limit used to determine the number of people that will board the elevator. Accommodates loading some of the people and alerting UI of number left. Two types added, fast and slow
3. Person – has a destination (not used in implementation), would be useful when extending to allow different destinations for the people on an elevator.
4. Request – links to person, source floor, destination floor, direction

# Notes:

The entities in this solution can be moved to be part of the domain. This implementation started with them separate to initially accommodate any future uses, e.g. a maintenance application that needs to use the same entities without the same business rules.

## Interface

Interface added to the Elevator. This will accommodate different elevator types. In this implementation **Speed** is a differentiator on the elevators, although the weight can also be set differently.

Domain

The shaft has the core business rules, determining which is the closest elevator. Domain also raises events from the objects to the calling UI (console application **Hub**).

Hub

The hub application is a console application used to take input from the user, process using the domain and entities and display the result.

Instructions that can be captured are:

1. Floor where elevator is required.
2. Number of people waiting at the floor
3. Direction – can be used in future business rules to check that the destination floor is in the correct direction. Not used in this implementation.

The instruction is added to the shaft as a request, and the shaft determines the nearest elevator, and starts moving it to the floor.

Once at the floor, an additional instruction is captured as to the destination floor.

# Notes:

1. Input can be improved to capture floor, number of people waiting and direction as a comma separated input. Will require more validation as user can capture combination which may be invalid.
2. Multiple request input can be added if each floor’s request is captured as a single input (see above). This will allow multiple lifts to move at the same time. Would need time to implement point 1 above as well as group multiple requests and fire off at the same time.
3. Extension would be possible to allow input of destination for individual passengers. Would be able to implement **disembark** to accommodate the different floors, as well as proceed to the final destination.

# Challenges:

I was not able to implement receiving a new request while the current request was being processed. I would need to investigate how to achieve this using a console application, without having a confusing interface/user experience.