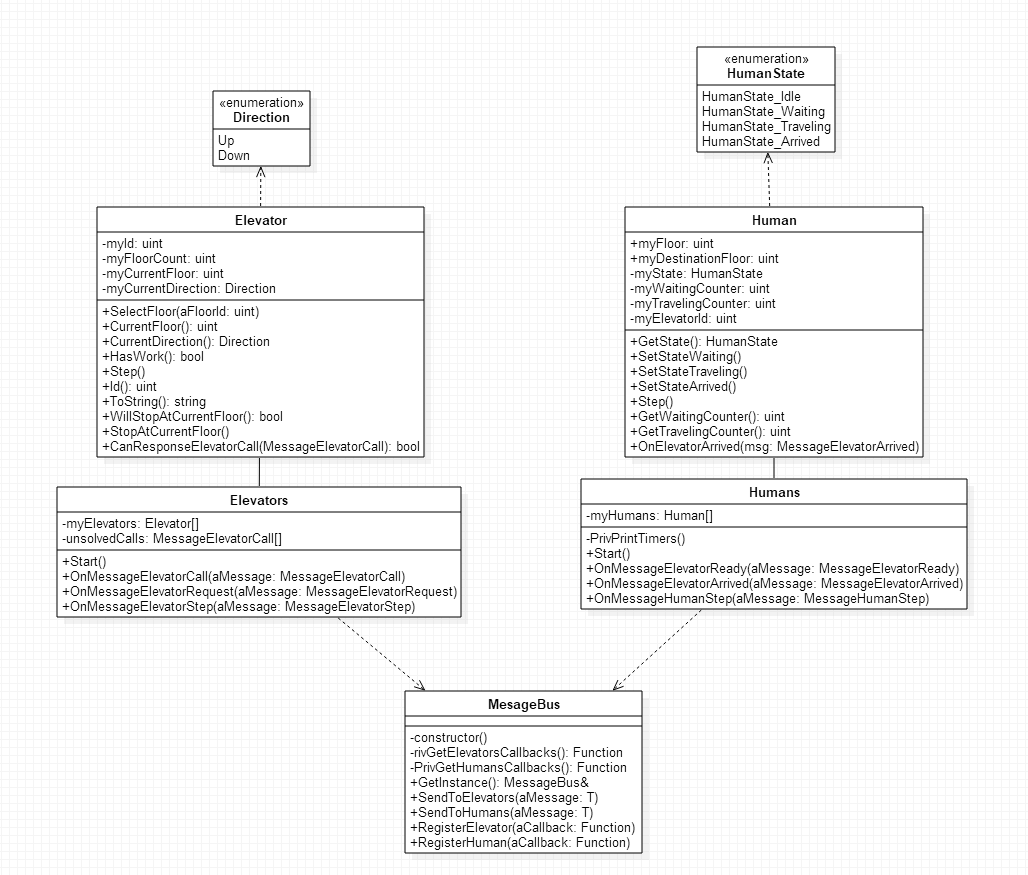
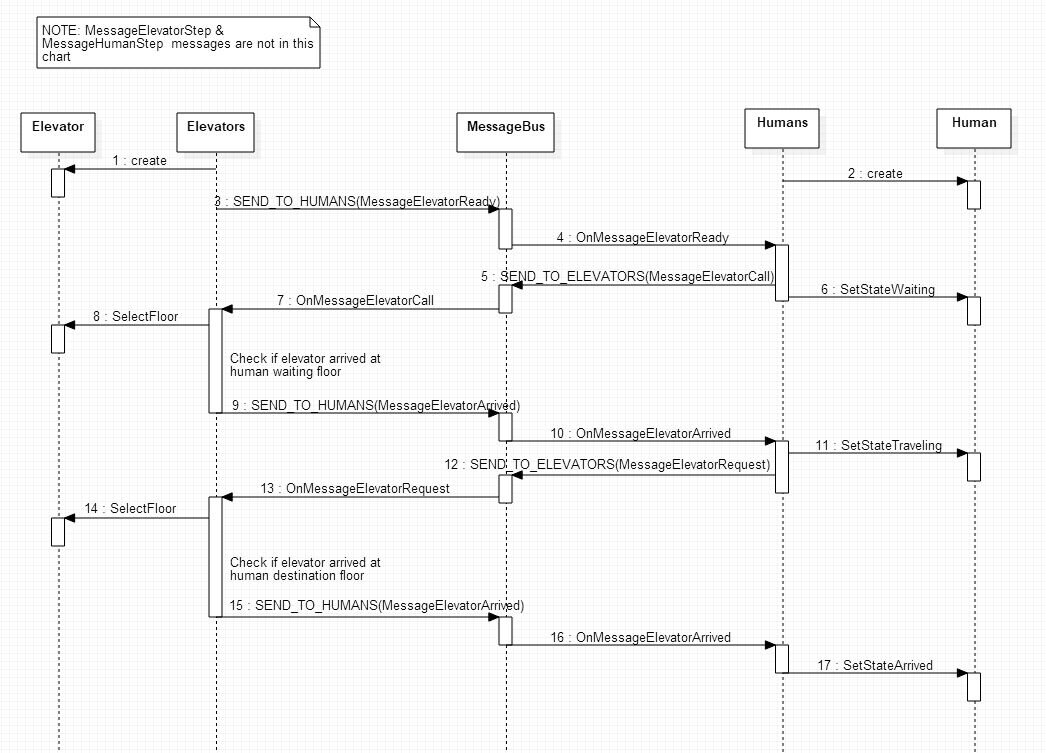
**Elevator System**

1. **Introduction**

**1.1 main classes diagram of the system**

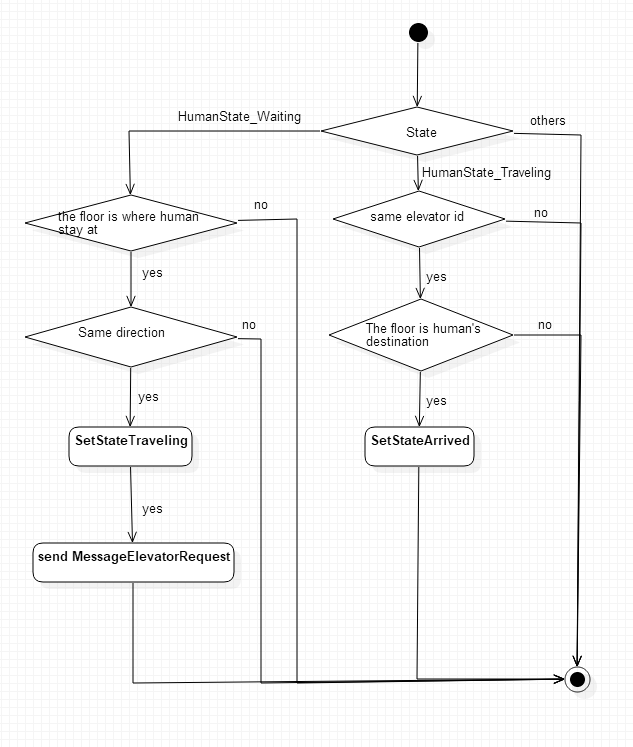


**1.2 Basic message and call flow between an elevator and a human**



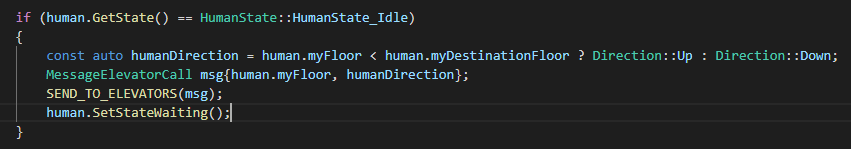
1. **Humans module design and implementation**
   1. **Handle MessageElevatorArrived**

For class Human, I added a private member myElevatorId(unsigned int) to indicate which elevator the human is in when human in travel. And added a public method “void OnElevatorArrived(const MessageElevatorArrived&)”, This method is to handle message MessageElevatorArrived. The general procedure for Humans handle MessageElevatorArrived is as below:



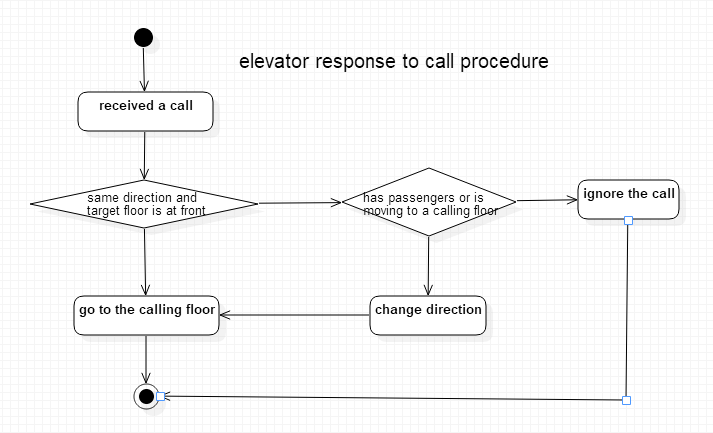
* 1. Handle MessageHumanStep

Human only care if the elevator is arrived. So in Humans::OnMessageHumanStep, what I added is only to change Human Idle state to waiting state, and send the message MessageElevatorCall. The related code as below:



1. **Elevators module implementation**
   1. **Handle MessageElevatorCall**

For one elevator scenario, I choose LOOK algorithm to schedule it. The basic strategy is the elevator only response to the call that on the same direction and at front, when no more call on current direction, change the elevator’s direction and response to calls on new direction.



For multiple elevator scenario, I choose the closest elevator from all elevators that can respond the call to respond the call.

* 1. **Handle MessageElevatorRequest**

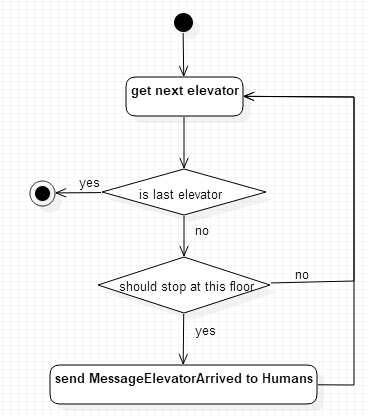
I added a private member “floorsNeedStop (std::set<unsigned int>)” in class Elevator, it represents this elevator has to go to these floors. When elevator is received the request, the destination floor of the request should be stored in floorsNeedStop. Detail code as below:



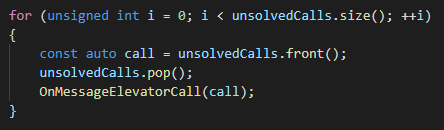
* 1. **Handle MessageElevatorStep**

Elevators mainly do two things here, first one is to check whether any elevator needs to stop at this floor, and do relate operations. Another one is to check if unsolved calls can be responded, and do relate operations.

For first thing, the procedure is:



For second thing, I added a queue to save the unsolved calls, if the call is responded, the call will be deleted in the queue, otherwise, the call will be re-added at the end of the queue. Detail code as below (if call cannot be responded, this call will be added again in OnMessageElevatorCall):



1. **Tests**

For detail test result, please see: tests.xlsx