Coding Challenge v2 (Platform Team)

Technical exercise

You are being asked to design and build the navigation logic for an elevator in a building.

The building has 6 floors. Each floor has a button or buttons outside of the elevator to summon the elevator and tell it what direction a passenger wants to go. Floors 2-5 have up and down buttons, floor 1 has an up button only, and floor 6 has a down button only.

The elevator has buttons inside for passengers to tell it what floor they want to go, so there are 6 buttons inside, one for each floor.

The navigation logic you write should asynchronously take button presses as input. This simulates the button presses of a “passenger”. To show your navigation logic in use, you should log a simulated elevator’s movements and stops (in terms of the floor numbers) as output. Model the elevator as taking 5 seconds to go from one floor to the next and 10 seconds to make a stop at a floor.

The navigation logic should satisfy these constraints:

- Once started, the elevator should wait for input and stay idle until the first input arrives. It should not produce random passengers or simulate button presses on its own. When there are no requests to take care of, it should again stay idle until more come in.
- Passengers should be able to press an outside button to summon the elevator to stop at their floor and an inside button to be taken to the specified floor. The elevator should accept inside and outside button inputs separately. That is, outside button presses should not be associated with the destination floor of the passenger to be picked up at the outside button’s floor.
  - **Example scenario:** The elevator is idle at the 1st floor. Then the outside-up button at the 3rd floor is pressed.
  - **Expected behavior:** The elevator should go to the 3rd floor, wait there during 10 seconds, and then stay idle and wait for the next button press.
- When the elevator is moving (upward or downward), if an outside button is pressed and the floor of that button is compatible with the elevator’s current direction, the elevator should visit that floor when it passes for the first time without changing its direction.
  - **Example scenario:** The elevator is idle at floor 1. Then the inside button for the 5th floor is pressed. When the elevator is moving upward direction and at the 2nd floor, the outside-up button at the 4th floor is pressed.
  - **Expected behavior:** When going upward direction, the elevator should stop at the 4th floor. Then visit the 5th floor.
- When the elevator is moving (upward or downward), if an outside button is pressed and the floor of that button is NOT compatible with the elevator’s current direction, the elevator should first visit all the requested floors compatible with the current direction, then change its direction and visit that floor.
  - **Example scenario:** The elevator is idle at floor 1. Then the inside button for the 5th floor is pressed. When the elevator is moving upward direction and at the 2nd floor, outside-down button at the 4th floor is pressed.
  - **Expected behavior:** When going upward direction, the elevator should NOT stop at the 4th floor and visit the 5th floor first. And then, it should change its direction and visit the 4th floor.
- When the elevator is moving (upward or downward), if an inside button that is compatible with the elevator’s current direction is pressed, the elevator should visit that floor when it passes for the first time without changing its direction.
  - **Example scenario:** The elevator is idle at floor 1. Then the inside button for the 5th floor is pressed. When the elevator is moving upward direction and at the 2nd floor, the inside button for the 4th floor is pressed.
  - **Expected behavior:** The elevator should stop and wait at the 4th floor. Then it should visit the 5th floor.
- When the elevator is moving (upward or downward), if an inside button that is NOT compatible with the elevator’s current direction is pressed, the elevator should first visit all the requested floors compatible with the current direction, then change its direction and visit that floor.
  - **Example scenario:** The elevator is idle at floor 1. Then an inside button press arrives from the 5th floor. When the elevator is moving upward direction and at the 3rd floor, the inside button for the 2nd floor is pressed.
  - **Expected behavior:** The elevator should visit the 5th floor first. Then it should change its direction and visit the 2nd floor.
- It’s not required to keep the record of passengers, e.g.: their name, button presses, source and destination floors, etc.

Additional Requirements

- The elevator logic should expose an HTTP API for interaction. The API should provide endpoints to simulate button presses (both inside and outside the elevator) and to retrieve the current state of the elevator (e.g., current floor, direction, and whether it is stopped or moving).
- The application should be containerized and deployed into a local Kubernetes cluster using Helm. The Helm chart should encapsulate the deployment configuration of your application, including any necessary Kubernetes resources (Deployments, Services, ConfigMaps, etc.)
- You should provide instructions on deploying your application to a local Kubernetes environment using the Helm chart. Also, include steps to test the elevator's functionality through the provided HTTP API.