# Getting Ready: Elevator System

Understand the elevator system problem and learn the questions to simplify this problem.

**We'll cover the following**

* [Problem definition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Problem-definition)
* [Expectations from the interviewee](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Expectations-from-the-interviewee)
  + [Multiple elevators](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Multiple-elevators)
  + [Display](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Display)
  + [Optimization](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Optimization)
* [Design approach](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-approach)
* [Design pattern](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-pattern)

## Problem definition

An **elevator** is an integral part of buildings that have multiple floors. The elevator car could be in different states, either up or down, or could be stopped on some floor. Anyone can request an elevator car from any floor using the buttons on the panel. The elevator car’s algorithm will set the priority and take action accordingly, so the wait time is minimum. Inside an elevator, there will be a panel for passengers to select the floor on which they want to go. The elevator car will have a fixed capacity for the number of passengers and a display to show on which floor the elevator car is currently located.

The elevator system

## Expectations from the interviewee

Numerous components are present in a typical elevator system, each with specific constraints and requirements placed on them. The following provides an overview of some of the main expectations that the interviewer will want to hear you discuss in more detail during the interview:

### Multiple elevators

You can also ask the interviewer whether the system should handle multiple elevators or just the single one. For this you can ask the following questions:

1. Can there be multiple elevator cars in the building?
2. How could a one-elevator system be different from a multi-elevator system in terms of user wait time and running cost?

### Display

You may want to ask the interviewer about the display of the elevator system:

1. How can passengers see the status of the elevator car and request an elevator car?
2. Would the display be the same inside and outside of the elevator?

### Optimization

An interviewer would expect you to ask questions about the optimization of the elevator system in terms of wait time, maintenance, throughput, etc. You can ask questions like:

1. What would be an optimized solution to minimize the wait time of the passengers?
2. How are we going to minimize the running cost of the elevator system?

## Design approach

We’ll design this elevator system using the bottom-up design approach. For this purpose, we will follow the steps below:

* Identify and design the smallest components first, like, the button and door.
* Use these small components to design bigger components, for example, the panel, elevator car, and building.
* Repeat the steps above until we design the whole system.

## Design pattern

It is always a good practice to discuss the design patterns that an elevator system falls under, during the interview. Stating the design patterns will give the interviewer a positive impression and shows that the interviewee is well-versed in the advanced concepts of object-oriented design.

The following design patterns are used to design the elevator system:

* Strategy design pattern
* State design pattern
* Delegation design pattern

Let’s explore the requirements of the elevator system in the next lesson.

Back

**Requirements for the Elevator System**

Learn about all requirements of the elevator system.

**We'll cover the following**

* [Requirement collection](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Requirement-collection)

In this lesson, we’ll list the requirements of the elevator system. This is a very crucial step as requirements define the scope of a problem, so getting them right from the interviewer and understanding them well will make the design of the rest of the system smooth and easy.

We’ll use the notational convention to identify each requirement with a unique label "Rn", where "R" is short for Requirement and "n" is a natural number.

**Requirement collection**

For the elevator design problem, the requirements are defined below:

**R1:** There exist multiple elevator cars and floors in the building.

**R2:** The building can have a maximum of 15 floors and three elevators.

**R3:** The elevator car can move up or down or be in an idle state.

**R4:** The elevator door can only be opened when it is in an idle state.

**R5:** Every elevator car passes through each floor.

**R6:** The panel outside the elevator should have buttons to call an elevator car and to specify whether the passenger wants to go up or down.

**R7:** The panel inside the elevator should have buttons to go to every floor. There should be buttons to open or close the lift doors.

**R8:** There should be a display inside and outside the elevator car to show the current floor number and direction of the elevator car.

**R9:** The display inside the elevator should also show the capacity of the elevator car.

**R10:** Each floor has a separate panel and a display for each elevator car.

**R11:** Multiple passengers can go to the same or different floors in the same or opposite direction.

**R12:** The elevator system should be able to control the elevator car movement and the door functioning and monitor the elevator car.

**R13:** The elevator control system should be able to send the most appropriate elevator to the passenger when the passenger calls the elevator car.

**R14:** The elevator car can carry a maximum of eight persons or 680 kilograms at once.

We’ve identified our requirements for the problem. In the next lesson, we will define different use cases for the elevator control system.

# Use Case Diagram for the Elevator System

Learn how to define use cases and create the corresponding use case diagram for the elevator system.

**We'll cover the following**

* [System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#System)
* [Actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actors)
  + [Primary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Primary-actors)
  + [Secondary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Secondary-actors)
* [Use Cases](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-Cases)
  + [Passenger](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Passenger)
  + [System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#System)
* [Relationships](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationships)
  + [Generalization](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Generalization)
  + [Associations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Associations)
  + [Include](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Include)
* [Use case diagram](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-case-diagram)

Let’s build the use case diagram of the elevator system and understand the relationship between its different components.

First, we will define the different elements of our elevator, followed by the complete use case diagram of the system.

## System

Our system is an "elevator."

## Actors

Next, we will define our elevator system’s main actors.

### Primary actors

* **Passenger:**This actor is the passenger in an elevator. It can request an elevator, open and close the door of an elevator, move up and down using the elevator, and press the emergency button.

### Secondary actors

* **System:** It can open and close the elevator door, display floor level, and move the elevator according to the dispatcher algorithm.

## Use Cases

In this section, we will define the use cases for the elevator. We have listed the use cases according to their respective interactions with a particular actor.

**Note:**You will see some use cases occurring multiple times because they are shared among different actors in the system.

### Passenger

* **Press elevator panel button:**To press the button on the elevator panel to select the destination floor, request to open/close the elevator door while it is stopped, or call an emergency
* **Press hall panel button:**To press the button on the hall panel to select the request for the elevator

### System

* **Move/stop elevator:**To move up or down or to stop the elevator on a specific floor
* **Dispatcher algorithm:**For proper elevator functionality according to the algorithm
* **Display (inside/outside):**To display the floor number on the screen inside and outside the elevator
* **Open/close door:**To open and close the elevator door

There are some use cases that are not directly related to any actor. These are elaborated below

* **Request for elevator:**To request the elevator
* **Floor request:**To submit a request for the destination floor
* **Door open/close request:**To submit a request to open/close the elevator door
* **Call Emergency:**To call the support team in case of emergency

## Relationships

This section describes the relationships between and among actors and their use cases.

### Generalization

We can press the elevator panel button to select the destination floor, request to open/close the elevator door while it is stopped, or call an emergency. This demonstrates that the “Press elevator panel button” use case has a generalization relationship with “Floor request,” “Door open/close request,” and “Call emergency” use cases.

### Associations

The table below shows the association relationship between actors and their use cases.

|  |  |
| --- | --- |
| **Passenger** | **System** |
| Press elevator panel button | Dispatcher algorithm |
| Press hall panel button | Open/close door |
|  | Display (inside/outside) |
|  | Move/stop elevator |

### Include

* When a floor request is submitted, the system will move the elevator and stop it at the requested floor. Hence, the “Floor request” use case has an include relationship with the “Move/stop elevator” use case.
* When a door open/close request is submitted, the system will open/close the elevator door. Hence, the “Door open/close request” use case has an include relationship with the “Open/close door” use case.
* When any button on the hall panel is pressed, a request for the elevator is submitted. Hence, the “Press hall panel button” use case has an include relationship with the “Request for elevator” use case.

## Use case diagram

Here’s the use case diagram of the elevator system:

The use case diagram of elevator system

In the next lesson, we will discuss the class diagram with a detailed explanation of all classes and their relationship.

A diagram of an elevator

Description automatically generated

# Class Diagram for the Elevator System

Learn to create a class diagram for the elevator system problem using the bottom-up approach.

**We'll cover the following**

* [Components of an elevator system](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Components-of-an-elevator-system)
  + [Button](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Button)
  + [Elevator panel and hall panel](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Elevator-panel-and-hall-panel)
  + [Display](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Display)
  + [Door](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Door)
  + [Elevator car](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Elevator-car)
  + [Floor](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Floor)
  + [Building](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Building)
  + [Elevator system](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Elevator-system)
  + [Enumerations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumerations)
* [Relationship between the classes](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationship-between-the-classes)
  + [Aggregation](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Aggregation)
  + [Composition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Composition)
  + [Inheritance](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Inheritance)
* [Class diagram of the elevator system](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Class-diagram-of-the-elevator-system)
* [Design pattern](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-pattern)
* [Additional requirements](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Additional-requirements)
  + [FCFS](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#FCFS)
  + [SSTF](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#SSTF)
  + [SCAN](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#SCAN)
  + [LOOK](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#LOOK)

In this lesson, we’ll identify and design the classes, abstract classes, and interfaces based on the requirements that we have previously gathered from the interviewer in our elevator system.

## Components of an elevator system

As mentioned earlier, we should design the elevator system using a bottom-up approach. Therefore, we will first identify and design classes of the smaller components like button, door, and floor. Then, we will create the class of the entire elevator control system, which will contain these smaller components.

### Button

Button is an abstract class. There can be two types of buttons i.e., the elevator button and the hall button. The status of the button determines whether the button is pressed or unpressed. We can press the button or check the status of the button through the Button class.

The ElevatorButton subclass is inherited from the Button class and represents the buttons that are inside the elevator. When the elevator button is pressed, it specifies what would be the destination floor of the elevator car or where the passenger wants to go.

Similar to ElevatorButton, HallButton is also a subclass of the Button class. This class represents the buttons that are outside the elevator. This class used the enumeration Direction to specify whether the button is for going up or down. The hall button has two important pieces of information, the floor from where the button is pressed and the direction in which the passenger wants to move.

The UML representation of these classes is shown below:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The following classes show an inheritance relationship:

* Both ElevatorButton and HallButton extend the Button class.

**Note:** We have already discussed the inheritance relationship between classes in the component section.

## Class diagram of the elevator system

Here’s the complete class diagram for our elevator system:

A diagram of a building

Description automatically generated

state and delegates all the state-specific tasks to that object. For example, elevators have multiple states like working or idle, etc. Based on the state, the system infers which method or behavior of the elevator should be invoked.

## Additional requirements

The interviewer can introduce some additional requirements in the elevator control system, or they can ask some follow-up questions. The additional requirement can be about how the dispatcher works in the elevator system. The interviewer can ask to devise an algorithm to optimize any of these:

* To minimize the wait time of the system
* To minimize the wait time of the passenger
* To maximize throughput
* To minimize the power usage or cost

To optimize the elevator system, we have different dispatching algorithms.

### FCFS

**First Come First Serve (FCFS)**is a scheduling algorithm by which the passenger who comes first gets the elevator car and reaches the destination. There are four states of an elevator car with respect to the passenger:

* The elevator car is in an idle state.
* The elevator car is moving towards the passenger and in the same direction the passenger wants to go.
* An elevator car is moving towards the passenger but in the opposite direction the passenger wants to go.
* The elevator car is moving away from the passenger.

In this algorithm, the dispatcher will try to find elevators that are in either of the first two states and ignore those elevators which are in either of the last two states.

The FCFS algorithm

The advantage of this algorithm is that it is simple and easy to implement. The drawback of this algorithm is that extra elevator movements occur by this algorithm which results in more power usage and cost. To implement FCFS, we can use a queue data structure to keep track of which passenger comes first.

### SSTF

**Shortest Seek Time First (SSTF)** is an algorithm in which the passenger who is closest to the elevator car would get the elevator car. This algorithm is considered better than FCFS since less elevator movement is required as compared to the FCFS algorithm. This algorithm also results in an increased throughput. However, there is a loophole in this method where it always chooses the minimum distant passengers and ignore the farther ones completely. To implement this algorithm, we can use a priority queue, min-heap, or an array data structure.

### SCAN

**SCAN** is also known as the **Elevator Algorithm**. The elevator car starts from one end of the building and moves towards the other end, servicing requests in between. The advantage of this method is that it serves multiple requests in parallel. However, it results in increased cost as the elevator car only changes its direction at either the top floor, or the lowest floor. The implementation of SCAN can be done using two boolean arrays or a single HashMap, or two priority queues data structures to track the floor where the elevator should stop.

### LOOK

**LOOK** is also known as the look-ahead SCAN algorithm. It is an improved version of the SCAN Algorithm. In this algorithm, the elevator car stops when there is no request in front of them. It will move again on the basis of the request. The advantage of this algorithm is that the elevator car does not always go till the end of the building but can change its direction in between. This algorithm can be implemented using a HashMap, TreeMap, or binary search tree data structure.

The LOOK algorithm

**1** of 12

We have completed the class diagram of the elevator system according to the requirements. Let’s design the sequence diagram of the elevator control system in the next lesson.

Back

**Sequence Diagram for the Elevator System**

Visualize the sequence diagram for calling the elevator, and solve a challenge.

**We'll cover the following**

* [Elevator call](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Elevator-call)
* [Sequence challenge: Elevator ride](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Sequence-challenge:-Elevator-ride)

A sequence diagram is a great way to understand the interactions between different entities and objects in the system. There can be different sequence diagrams that we can create for our elevator system. In this lesson, we will create sequence diagrams for the following two interactions:

* **Elevator call:** The passenger calls the elevator.
* **Sequence challenge:** The passenger rides the elevator to a floor.

**Elevator call**

The sequence diagram for an elevator call should have the following actors and objects that will interact with each other:

* **Actor:** Passenger
* **Objects:** HallButton, HallButton, ElevatorSystem, Dispatcher, ElevatorCar, and Door

Here are the steps in the elevator call interaction:

1. The passenger presses the hall button to call the elevator.
2. The hall button signals the elevator system to call an elevator car to the passenger's floor.
3. The elevator system informs the dispatcher to select the best car.
4. The dispatcher returns the best car to the system.
5. The elevator system signals the elevator car to move to the passenger's floor.
6. The elevator car signals the system when it arrives on the floor.
7. The system signals the hall button that the elevator has arrived.
8. The hall button is unpressed.
9. The elevator system signals the doors to open.
10. The door opens for the passenger.

Based on the order above, the sequence diagram for an elevator call in the elevator system is given below.

A diagram of a lift

Description automatically generated

You will complete a sequence diagram for an elevator ride from one floor to another. A skeleton of the sequence diagram for an elevator ride is given below:

A diagram of a elevator

Description automatically generated

Notice that the arrows in the diagram above are numbered from 1 to 9. The message boxes shown below are the messages to be exchanged between the actor(s) and object(s). Can you rearrange the messages below in the correct sequence of order they should appear in the skeleton of the sequence diagram above?

**Note:** If you get stuck, just click the “Show Solution” button to check the correct answer.

Rearrange the sequence.

pressDown(floor)

door closes

door opens

destFloor(floor)

stop at floor

openDoor()

closeDoor()

unpress

car arrived

move(floor)

ResetShow SolutionSubmit

Alternatively, you can also click the "Show complete diagram" button below to see the complete sequence diagram for the elevator ride interaction.

Show complete diagram

Next, let’s look at the activity diagrams for the elevator system to understand the control flow of the system.

Back

# Activity Diagram for the Elevator System

Create some activity diagrams for the elevator system problem.

**We'll cover the following**

* [The passenger arrives at the desired floor](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#The-passenger-arrives-at-the-desired-floor)
  + [States](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#States)
  + [Actions](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actions)
* [Activity challenge: The passenger calls for the elevator](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Activity-challenge:-The-passenger-calls-for-the-elevator)

An activity diagram is a great way to visualize the flow of messages from one activity to the other in the system. There can be different activity diagrams that we can create for our elevator system. In this lesson, we will create activity diagrams for the following two activities:

* The passenger arrives at the desired floor.
* **Activity challenge:**The passenger calls for the elevator.

## The passenger arrives at the desired floor

The following are the states and actions that will be involved in this activity diagram.

### States

**Initial state:**The passenger enters the elevator car.

**Final state:**There are two final states present in this activity diagram. These are shown below:

* The passenger arrives at the destination floor.
* The passenger is not allowed due to max load/capacity issues.

### Actions

The passenger enters the elevator car. The elevator car checks if the safety limits are met. The elevator car stops at other passengers’ floors. Finally, the elevator car reaches the passenger’s desired floor.

Based on the order above, the activity diagram of a passenger arriving at their desired floor is given below.

**Note:** Here, the passenger is just entering the elevator, so either the up or the down button can be pressed.

A diagram of a passenger

Description automatically generated

## Activity challenge: The passenger calls for the elevator

You will create an activity diagram of a passenger calling for the elevator.

A skeleton of the activity diagram is given below.

A diagram of a passenger calling

Description automatically generated

Notice that the actions in the diagram above are numbered from 1 to 10. The slots shown below represent the activities, and the arrows represent the flow from one activity to the other.

Can you rearrange the slots below in the correct order they should appear in the activity diagram above?

**Note:** If you get stuck, just click the “Show Solution” button to check the correct answer.

Fill the missing slots with the correct actions for a passenger calling for the elevator.

Up direction

Drag and drop the cards in the blank spaces.

The elevator door opens

Floors button light is turned on

Screen shows approaching elevator cars current floor

Passenger enters the elevator

Elevator reaches passengers floor

The elevator door closes

Passenger presses the button

Floors button light is turned off

Down direction

ResetShow SolutionSubmit

Alternatively, you can also click the "Show complete diagram" button below to see the complete sequence diagram.

A diagram of a elevator system

Description automatically generated

We've looked at some of the activity diagrams of our elevator system. In the next lesson, we will present the code for our designed classes in some of the most popular languages.

Back

# Code of Elevator System

Write object-oriented code to implement the design of the elevator system problem.

**We'll cover the following**

* [Elevator system classes](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Elevator-system-classes)
  + [Enumerations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumerations)
  + [Button](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Button)
  + [Elevator panel and hall panel](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Elevator-panel-and-hall-panel)
  + [Display](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Display)
  + [Elevator car](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Elevator-car)
  + [Door and floor](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Door-and-floor)
  + [Elevator system and building](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Elevator-system-and-building)
* [Wrapping up](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Wrapping-up)

We’ve discussed different aspects of the elevator system and observed the attributes attached to the problem using various UML diagrams. Let’s explore the more practical side of things, where we will work on implementing the elevator system using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in the elevator control system:

* Java
* C#
* Python
* C++
* JavaScript

## Elevator system classes

In this section, we’ll provide the skeleton code of the classes designed in the class diagram lesson.

**Note:** For simplicity, we aren’t defining getter and setter functions. The reader can assume that all class attributes are private and accessed through their respective public getter methods and modified only through their public methods function.

### Enumerations

First of all, we will define all the enumerations required in the elevator system. According to the class diagram, there are three enumerations used in the system i.e., ElevatorState, Direction and DoorState.The code to implement these enumerations is as follows:

**Note:** JavaScript does not support enumerations, so we will be using the Object.freeze() method as an alternative that freezes an object and prevents further modifications.

Java

// definition of enumerations used in elevator system

enum ElevatorState {

IDLE,

UP,

DOWN

}

enum Direction {

UP,

DOWN

}

enum DoorState {

OPEN,

CLOSE

}

### Button

This section contains the implementation of a Button class and its subclasses which are HallButton and the ElevatorButton. The Button class has a pure virtual function isPressed() in it. The code to implement this relationship is given below:

public abstract class Button {

private boolean status;

public pressDown();

public abstract boolean isPressed();

}

public class HallButton extends Button {

private Direction buttonSign;

public boolean isPressed() {

// definition

}

}

public class ElevatorButton extends Button {

private int destinationFloorNumber;

public boolean isPressed() {

// definition

}

}

### Elevator panel and hall panel

ElevatorPanel and the HallPanel are classes which use the instance of ElevatorButton and HallButtonrespectively. The code to implement these classes is provided below:

public class ElevatorPanel {

private List<ElevatorButton> floorButtons;

private ElevatorButton openButton;

private ElevatorButton closeButton;

}

public class HallPanel {

private HallButton up;

private HallButton down;

}

### Display

This component shows the implementation of the Display class. This class is responsible for showing the display inside and outside of the elevator cars. The code to implement this class is shown below:

public class Display {

private int floor;

private int capacity;

private Direction direction;

public void showElevatorDisplay();

public void showHallDisplay();

}

### Elevator car

This section contains the definition of the ElevatorCar class. An elevator car contains the instance of Door, Display, and ElevatorPanel. The implementation of this class is represented below:

public class ElevatorCar {

private int id;

private Door door;

private ElevatorState state;

private Display display;

private ElevatorPanel panel;

public void move();

public void stop();

public void openDoor();

public void closeDoor();

}

### Door and floor

This section contains the code for the Door and Floor classes. In the Door class, the enumeration DoorState is used and the Floor class contains the instances of Display and HallPanel. The implementation of this class is given below:

public class Door {

private DoorState state;

public boolean isOpen();

}

public class Floor {

private List<Display> display;

private List<HallPanel> panel;

public boolean isBottomMost();

public boolean isTopMost();

}

### Elevator system and building

The final class of an elevator system is the ElevatorSystem class which will be a Singleton class, which means that the entire system will have only one instance of this class. Moreover, there is a Building class that contains the instances of Floor and ElevatorCar. The implementation of these Singleton classes are provided below:

public class ElevatorSystem {

private Building building;

public void monitoring();

public void dispatcher();

// The ElevarSystem is a singleton class that ensures it will have only one active instance at a time

private static ElevatorSystem system = null;

// Created a static method to access the singleton instance of ElevatorSytem class

public static ElevatorSystem getInstance() {

if (system == null) {

system = new ElevatorSystem();

}

return system;

}

}

public class Building {

private List<Floor> floor;

private List<ElevatorCar> elevator;

private static Building building = null;

public static Building getInstance() {

if (building == null) {

building = new Building();

}

return building;

}

}

## Wrapping up

We've explored the complete design of an elevator control system in this chapter. We've looked at how a basic elevator system can be visualized using various UML diagrams and designed using object-oriented principles and design patterns.