# Getting Ready: Amazon Locker Service

Understand the Amazon Locker service problem and learn the questions to further 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)
  + [Locker size](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#Locker-size)
  + [Locker selection](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#Locker-selection)
  + [Locker status](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#Locker-status)
  + [Returning an item](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#Returning-an-item)
* [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

**Amazon** is an online retail platform that allows its customers to place orders and buy products online. There are times when the customer is not available in the particular location to pick up the order. In such a case, Amazon Locker can be one of the most secure way of delivery.

**Amazon Locker** is also known as Amazon Hub or Amazon Hub Locker. It is a fully automated package delivery service provided by Amazon. Customers can choose any locker location as their delivery address and pick up the package from that location at no additional cost. In particular, when a customer places an order and chooses to get their item delivered to locker service, the system suggests the nearest available locker based on preferences. The order is packaged and placed in the locker. The customer gets the notification containing the code to open the locker, and they can pick up the package using that code within a valid amount of time. This is how the Amazon Locker service functions.

This problem is applicable to any retailer who wants to deliver goods safely to their customers.

The Amazon Locker service

## Expectations from the interviewee

The Amazon Locker service consists of multiple components. Each component has its own functionality and constraints. The following section provides an overview of some of the main expectations that the interviewer will want to hear you discuss in more detail during the interview.

### Locker size

Every locker is of a specific size in the Amazon Locker system. The interviewer expects you to ask questions listed below:

* Will every locker be of the same size?
* Is there any size restriction on an item that can be kept in the locker?

### Locker selection

The most significant part of the Amazon Locker service is the selection of the locker. The system has to make sure that more than one customer should not be able to access a locker at a single time. The interviewer expects you to ask the questions listed below to identify how the system will work in such situations:

* How will the system make sure that multiple customers do not get the same locker?
* Will the customer choose the locker of his own choice, or will the system assign him a locker based on availability?
* Can a customer get two lockers for different orders at the same time?
* Will the system keep in mind the locker and package sizes while assigning the locker to the customer?

### Locker status

Since this problem revolves around the locker, you may ask the questions listed below about the locker status:

* Is there any time constraint on the package that can be kept in the locker?
* What will happen if the customer does not come to pick up his package within the valid time period?

### Returning an item

Similar to the order delivery process, the item can also be returned through the Amazon Locker service. Therefore, you may ask the questions listed below:

* Can the customer return an item through the Amazon Locker service?
* If yes, will they get the same locker from which they picked up the item?
* How will the locker be assigned to the customer while returning an item?

## Design approach

We will design this Amazon Locker service using the bottom-up design approach. For this purpose, we will follow the steps below:

* Identify and design the simple components first, like the locker and item.
* Use these small components to design bigger components, such as the locker location and order that can be composed of multiple lockers and items, respectively.
* Repeat the steps above until we design the whole system.

## Design pattern

It is always a good practice to discuss the design patterns that the Amazon Locker service 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 can be used to design the Amazon Locker service:

* Strategy design pattern
* Repository design pattern

Let's explore the requirements of the Amazon Locker service in the next lesson.

Back

**Requirements for the Amazon Locker Service**

Learn about all requirements for the Amazon locker service.

**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 Amazon Locker service. 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**

The requirements for the Amazon Locker service are defined below:

**R1:** While ordering the item(s), the customer can choose the nearest location to pick up the order package from the locker.

**R2:** One or more items can be contained in one order. An order will be placed in a package before the delivery.

**R3:** There can be different sizes of lockers including extra small, small, medium, large, extra large, and double extra large.

**R4:** The locker is assigned to the customer based on the size of the order package.

**R5:** When the order package is delivered to the locker location specified by the customer, a 6-digit code will be sent to the customer to open the locker.

**R6:** The package will be kept or placed inside the locker for three days only.

**R7:** If the customer does not pick up the package from their locker within three days, the refund process will be initiated and the customer won’t be allowed to pick up the package any longer.

**R8:** Only eligible packages can be placed in the locker such that the size of the package must be less than the size of the locker.

**R9:** There can be multiple lockers at every locker location.

**R10:** The Amazon Locker is accessed within a specific time. Every location has its opening and closing time. Therefore, the customer should pick up the package during this time period.

**R11:** The item can be returned to the Amazon Locker if it doesn’t match the expectation of the customer or is faulty, and there is a refund policy available for that product.

**R12:** To return an item, the customer needs to choose the nearest locker location. An available locker will be assigned to them based on the size and location.

**R13:** When the customer picks up the order package from the locker, the locker’s state is changed to closed, and the customer will no longer be able to open the locker with the given code.

We've identified our requirements for the problem, and in the next lesson, we will define different use cases for the amazon locker system.

# Use Case Diagram for the Amazon Locker Service

Learn how to define use cases and create the corresponding use case diagram for the Amazon Locker 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)
  + [Customer](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#Customer)
  + [Delivery guy](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#Delivery-guy)
  + [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)
  + [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 Amazon Locker system and understand the relationship between its different components.

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

## System

Our system is “Amazon Locker."

## Actors

Now, we’ll define the main actors of our Amazon Locker system.

### Primary actors

* **Customer:**This is Amazon's customer who ordered a package delivered to the Amazon Locker. It can enter the code at the locker and get its product. This actor can also request a return and put the package back at the locker.
* **Delivery guy:**This can also enter the code and add the product to the locker so the "Customer" can pick it up. This actor can pick up a returned package from the locker.

### Secondary actors

* **System:** This can send the delivery and due date notifications, generate code, validate code, and choose a locker. It can also find the locker and open or close the locker door.

## Use Cases

This section will define the use cases for lockers. 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.

### Customer

* **Enter code:**To enter the code to open a locker
* **Add product:**To add a product to the locker
* **Remove product:**To pick up the product from the locker
* **Delivery notification:**To notify about the product location status
* **Submit return request:**To submit a return request to return a product
* **Overdue notification:**To notify if the date and time for the product pickup are passed

### Delivery guy

* **Enter code:**To enter the code to open a locker
* **Add product:**To add a product to the locker
* **Remove product:**To pick up the product from the locker
* **Return notification:**To notify about the product return status

### System

* **Validate code:**To validate the locker code entered
* **Find locker:**To find the locker as per the code entered
* **Lock/unlock door:**To lock or unlock the door lock
* **Return notification:**To notify about the product return status
* **Generate code:**To generate the locker code
* **Issue locker:**To issue the appropriate locker depending upon product size
* **Overdue notification:**To notify if the date and time for the product pickup are passed
* **Delivery notification:**To notify about the product location status

## Relationships

We describe the relationships between and among actors and their use cases in this section.

### Associations

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

|  |  |  |
| --- | --- | --- |
| **Customer** | **Delivery Guy** | **System** |
| Enter code | Enter code | Validate code |
| Add product | Add product | Find locker |
| Remove product | Remove product | Lock/unlock door |
| Delivery notification | Return notification | Return notification |
| Submit return request |  | Generate code |
| Overdue notification | Issue locker |
|  | Overdue notification |
| Delivery notification |

### Include

* When a “Customer” enters a code, the system then checks if the code is correct or not and finds the locker according to the code entered. Then, the system unlocks the door. This means that:
  + The "Enter code" use case has an include relationship with the "Validate code" use case.
  + The "Validate code" use case has an include relationship with the "Find locker" use case.
  + The " Find locker" use case has an include relationship with the "Lock/unlock door" use case.
* To return a product, the customer must go to the Amazon website and submit a return request. After the approval of the request, Amazon Locker will generate a code that will be used to access the locker.
  + The "Submit return request" use case has an include relationship with the "Request notification" use case.
  + The "Request notification" use case has an include relationship with the "Generate code" use case.

## Use case diagram

A diagram of a diagram

Description automatically generated

# Class Diagram for the Amazon Locker Service

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

**We'll cover the following**

* [Components of an Amazon Locker service](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-Amazon-Locker-service)
  + [Item](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#Item)
  + [Order](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#Order)
  + [Notification](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#Notification)
  + [Package and locker package](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#Package-and-locker-package)
  + [Locker](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#Locker)
  + [Locker location](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#Locker-location)
  + [Locker service](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#Locker-service)
  + [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)
  + [Association](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#Association)
  + [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 Amazon Locker service](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-Amazon-Locker-service)
* [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)

We’ll create the class diagram for the Amazon Locker service. In the class diagram, we will first design the classes for the system, and then we will identify the relationship between classes according to the requirements gathered for the Amazon Locker service problem.

## Components of an Amazon Locker service

In this section, we’ll define the classes for an Amazon Locker service. As mentioned earlier, we are following the bottom-up approach to design a class diagram for the Amazon Locker service.

### Item

The Item class represents each item of the order. Every item contains an Id and the order’s quantity. The class representation is shown below:

A screenshot of a computer

Description automatically generated

A screenshot of a box

Description automatically generated

A screenshot of a box

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

Description automatically generated

A diagram of a package

Description automatically generated

A diagram of a package

Description automatically generated

locker to the customer by considering the customer’s location, and locker size. Therefore, the Strategy design pattern can be applied here. Other than that, our system can also have the following strategies:

* OTP generation
* Random number generation
* Locker assignment
* Locker filtration

We can also use the Repository design pattern for the Amazon Locker system, where we can make the following repositories:

* Locker repository
* Package repository

We have completed the class diagram of the Amazon Locker service according to the requirements. Now, let's design its sequence diagram in the next lesson.

**Sequence Diagram for the Amazon Locker Service**

Create a sequence diagram for product return in the Amazon Locker system.

**We'll cover the following**

* [Return package](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#Return-package)

Sequence diagrams are 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 Amazon Locker system. In this lesson, we will create sequence diagrams for the following interaction.

**Return package**

The sequence diagram for the package return should have the following actors and objects that will interact with each other:

* **Actor:** Customer
* **Object:**Locker
* System

Here are the steps in the return package interaction:

1. The customer requests the system to return a package.
2. If the package return request is approved:
   1. The system requests an available locker.
   2. The locker is returned to the system.
   3. The system sends an OTP code to the customer.
   4. The customer verifies their code.
   5. If verification is successful:
      1. The system assigns a locker to the customer.
      2. The customer places their package in the locker.
   6. Else:
      1. The customer receives an error message.
3. Else, if the package return request is not approved:
   1. The customer is informed that the return request is not approved.

Based on the order above, the sequence diagram of package return in the Amazon Locker system is given below:

A screenshot of a computer screen

Description automatically generated

Next, let's look at the activity diagrams for the Amazon Locker system to understand the control flow of the system.

# Activity Diagram for the Amazon Locker Service

Create an activity diagram for the Amazon Locker service problem.

**We'll cover the following**

* [Product pickup](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#Product-pickup)
  + [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 diagrams are 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 Amazon Locker system. In this lesson, we will create an activity diagram for the following activity.

## Product pickup

The states and actions that will be involved in this activity diagram are provided below.

### States

**Initial state:**A customer who has ordered a product from Amazon comes to the Amazon Locker to pick up the product.

**Final state:**The customer either successfully gets the product or the system shows an incorrect code error.

### Actions

The customer arrives at the Amazon Locker and enters the code. The system validates the code and opens the locker.

Based

A screenshot of a diagram

Description automatically generated

# Code for the Amazon Locker Service

Write the object-oriented code to implement the design of the Amazon Locker service problem.

**We'll cover the following**

* [Amazon Locker service 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#Amazon-Locker-service-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)
  + [Item and Order](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#Item-and-Order)
  + [Package and LockerPackage](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#Package-and-LockerPackage)
  + [Locker and LockerLocation](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#Locker-and-LockerLocation)
  + [LockerService and Notification](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#LockerService-and-Notification)
* [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 gone over the different aspects of the Amazon Locker service 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 Amazon Locker service 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 Amazon Locker service:

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

## Amazon Locker service classes

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

**Note:** For simplicity, we are not 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 method functions.

### Enumerations

First, we will define all the enumerations used in the Amazon Locker service. According to the class diagram, there are two enumerations in the system, i.e., LockerSize and LockerState 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.

// definition of enumerations used in the Amazon Locker service

public enum LockerSize {

EXTRA\_SMALL,

SMALL,

MEDIUM,

LARGE,

EXTRA\_LARGE,

DOUBLE\_EXTRA\_LARGE

}

public enum LockerState {

CLOSED,

BOOKED,

AVAILABLE

}

### Item and Order

The Item class represents the single item while the Order represents the order placed by the customer and can contain the list of items. The definition of these two classes is given below:

public class Item {

private String itemId;

private int quantity;

}

public class Order {

private String orderId;

private List<Item> items;

private String deliveryLocation;

}

### Package and LockerPackage

When an order is packed, it is represented by the Package , and the package which is contained in the locker is represented by the LockerPackage class. The code to implement these classes is shown below:

Java

C#

Python

C++

JavaScript

public class Package {

private String packageId;

private double packageSize;

private Order order;

public void pack();

}

public class LockerPackage extends Package {

private int codeValidDays;

private String lockerId;

private String packageId;

private String code;

private Date packageDeliveryTime;

public boolean isValidCode();

public boolean verifyCode(String code);

}

### Locker and LockerLocation

The Locker is the most important class of the system and a LockerLocation can contain multiple Locker instances. The implementation of these classes is given below:

Java

public class Locker {

private String lockerId;

private LockerSize lockerSize;

private String locationId;

private LockerState lockerState;

public boolean addPackage();

public boolean removePackage();

}

public class LockerLocation {

private String name;

private List<Locker> lockers;

private double longitude;

private double latitude;

private Date openTime;

private Date closeTime;

}

### LockerService and Notification

The final class of an Amazon Locker service is the LockerService class which will be singleton class, which means that the entire system will have only one instance of this class. The following code provides the definition of the LockerService and Notification classes used in the Amazon Locker service:

public class LockerService {

private List<LockerLocation> locations;

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

private static LockerService lockerService = null;

// Created a static method to access the Singleton instance of LockerService class

public static LockerService getInstance() {

if (lockerService == null) {

lockerService = new LockerService();

}

return lockerService;

}

}

public class Notification {

private String customerId;

private String orderId;

private String lockerId;

private String code;

public void send();

}

**Wrapping up**

We've explored the complete design of an Amazon Locker service in this chapter. We've looked at how an Amazon Locker service design can be visualized using various UML diagrams.