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

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
}
```

Constant definitions

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;
}
```

The Item and Order classes

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:

```
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);
}
```

The Package and LockerPackage classes

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:

```
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;
}
```

The Locker and LockerLocation classes

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();
}
```

The LockerService and Notification classes

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.

Back

Activity Diagram for the Amazon Locker Service

Complete

Next

Getting Ready: Vending Machine