Code for the Amazon Online Shopping System

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

We'll cover the following

We've gone over the different aspects of Amazon and observed the attributes attached to the problem using various UML diagrams. Let's explore the more practical side of things, where we'll implement Amazon 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 Amazon:

- Java
- C#
- Python
- C++
- JavaScript

Amazon classes

In this section, we'll 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.

Constants

The following code provides the definition of the various enums and custom data types being used in the Amazon design:

> **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.

```
public class Address {
  private int zipCode;
  private String address;
  private String city;
  private String state;
  private String country;
}

enum OrderStatus {
  UNSHIPPED,
  PENDING,
  SHIPPED,
  CONFIRMED,
  CANCELED,
  REFUNDED
}

enum AccountStatus {
  ACTIVE,
  INACTIVE,
  BLOCKED
}

enum ShipmentStatus {
  PENDING,
  SHIPPED,
  DELIVERED,
  ON_HOLD
}

```

Constant definitions

Account

The `Account` class refers to an account of a customer on Amazon and contains the personal details of a customer, such as their name, shipping address, credit card information, etc. It also provides authenticated users and admins with the functionality to add products, product reviews, and reset passwords. The definition of this class is given below:

```
public class Account {
  private String userName;
  private String password;
  private String name;
  private List<Address> shippingAddress;
  private AccountStatus status;
  private String email;
  private String phone;

  private List<CreditCard> creditCards;
  private List<ElectronicBankTransfer> bankAccounts;

  public Address getShippingAddress();
  public boolean addProduct(Product product);
  public boolean addProductReview(ProductReview review, Product product);
  public boolean deleteProduct(Product product);
  public boolean deleteProductReview(ProductReview review, Product product);
  public boolean resetPassword();
}
```

The Account class

Admin

The `Admin` class refers to a person from the administration of Amazon that can block users, and add, modify, or delete product categories. The definition of this class is provided below:

```
public class Admin {
  private Account account;
  public boolean blockUser(Account account);
  public boolean addNewProductCategory(ProductCategory category);
  public boolean modifyProductCategory(ProductCategory category);
  public boolean deleteProductCategory(ProductCategory category);
}
```

The Admin class

Customer

The `Customer` class is an abstract class that has the `AuthenticatedUser` and `Guest` classes derived from it. The classes are defined below:

```
// Customer is an abstract class
public abstract class Customer {
  private ShoppingCart cart;
  public abstract ShoppingCart getShoppingCart();
}

public class Guest extends Customer {
  public boolean registerAccount();

  public ShoppingCart getShoppingCart() {
    // functionality
  }
}

public class AuthenticatedUser extends Customer {
  private Account account;
  private Order order;
  public OrderStatus placeOrder(Order order);

  public ShoppingCart getShoppingCart() {
    // functionality
  }
}
```

Customer and its child classes

Product, product category, and product review

The following code defines the classes relating to a particular product:

```
public class Product {
  private String productId;
  private String name;
  private String description;
  private byte[] image;
  private double price;
  private ProductCategory category;
  private List<ProductReview> reviews;
  private int availableItemCount;
  private Account account;

  public int getAvailableCount();
  public int updateAvailableCount();
  public boolean updatePrice(double newPrice);
}

public class ProductCategory {
  private String name;
  private String description;
  private List<Product> products;
}

public class ProductReview {
  private int rating;
  private String review;
  private byte[] image;
  private AuthenticatedUser user;
}
```

The Product, ProductCategory, and ProductReview classes

Shopping cart and cart items

The `ShoppingCart` and `CartItem` classes are used to add items to the shopping cart, update the quantities, and send the list of items to checkout. Both classes are defined below:

```
public class CartItem {
  private int quantity;
  private double price;
  public boolean updateQuantity(int quantity);
}

public class ShoppingCart {
  private int totalPrice;
  private List<Items> items;

  public boolean addItem(Item item);
  public boolean removeItem(Item item);
  public List<Item> getItems();
  public boolean checkout();
}
```

The ShoppingCart and CartItem classes

Order and order log

The `Order` class is responsible for making the payment, updating the order status, and sending the particular order to shipment. The `OrderLog` class creates the log of the order and is referenced in the `Order` class to keep track of all the orders. The definition of these two classes is given below:

```
public class Order {
  private String orderNumber;
  private OrderStatus status;
  private Date orderDate;
  private List<OrderLog> orderLog;
  private ShoppingCart shoppingCart;

  public boolean sendForShipment();
  public boolean makePayment(Payment payment);
  public boolean addOrderLog(OrderLog orderLog);
}

public class OrderLog {
  private String orderNumber;
  private Date creationDate;
  private OrderStatus status;
}
```

The Order and OrderLog classes

Shipment and shipment log

The `Shipment` class keeps track of all the major details of the order's dispatch and creates the shipment record using the `ShipmentLog` class. These classes are defined below:

```
public class Shipment {
  private String shipmentNumber;
  private Date shipmentDate;
  private Date estimatedArrival;
  private String shipmentMethod;
  private List<ShipmentLog> shipmentLogs;

  public boolean addShipmentLog(ShipmentLog shipmentLog);
}

public class ShipmentLog {
  private String shipmentNumber;
  private Date creationDate;
  private ShipmentStatus status;
}
```

The Shipment and ShipmentLog classes

Payment

The `Payment` class is another abstract class with the `ElectronicBankTranfer`, `CreditCard`, and `Cash` classes as its child classes. This takes the `PaymentStatus` enum to keep track of the payment status. The definition of this class is provided below:

```
// Payment is an abstract class
public abstract class Payment {
  private double amount;
  private Date timestamp;
  private PaymentStatus status;
  public abstract PaymentStatus makePayment();
}

public class CreditCard extends Payment {
  private String nameOnCard;
  private String cardNumber;
  private String billingAddress;
  private int code;
  public PaymentStatus makePayment() {
     // functionality
  }
}

public class ElectronicBankTransfer extends Payment {
  private String bankName;
  private String routingNumber;
  private String accountNumber;
  private String billingAddress;
  public PaymentStatus makePayment() {
     // functionality
  }
}

public class Cash extends Payment {
  private String billingAddress;
```

Payment and its child classes

Notification

The `Notification` class is responsible for sending notifications to customers about the order and shipment status either via SMS or email. Since you can extend this by adding more options, the `Notification` class will be an abstract class. It is defined below:

```
// Notification is an abstract class
public abstract class Notification {
  private int notificationId;
  private Date createdOn;
  private String content;

  public abstract boolean sendNotification(Account account);
}

public class EmailNotification extends Notification {
  public boolean sendNotification(Account account) {
    // functionality
  }
}

public class PhoneNotification extends Notification {
  public boolean sendNotification(Account account) {
    // functionality
  }
}
```

Notification and its child classes

Search and catalog

The `Catalog` class contains the product information and implements the `Search` interface class to enable the search functionality based on the given criteria (name and category of product). Both classes are defined below:

```
public interface Search {
  public List<Product> searchProductsByName(String name);
  public List<Product> searchProductsByCategory(String category);
}

public class Catalog implements Search {
  private HashMap<String, List<Product>> products;

  public List<Product> searchProductsByName(String name) {
    // functionality
  }

  public List<Product> searchProductsByCategory(String category) {
    // functionality
  }
}

```

The Search interface and Catalog class

Wrapping up

We've explored the complete design of Amazon in this chapter. We've looked at how Amazon is visualized using various UML diagrams and designed using object-oriented principles and design patterns.

← Back

Activity Diagram for the Amazon Online Shopping Sy...

Complete

Next →

Getting Ready: Stack Overflow