Amazon Online Shopping System

→ Amazon is online platform that allows users to buy & sell products.

→ Numerous product categories, that customer can search & choose

→ Product can have review & rating

→ Customers have added their favorite products to the cart.

→ Payment to their orders.

→ Customer notified about their order & Shipment Status until the order is delivered.

Expectations from Interviewer

→ How will the buyer discover a product?

→ How will the search surface result.

→ How will the design adhere to such known best practices while also introducting innovative checkout Semantics like "one-click purchase".

→ How will the users pay? Will they use credit cards, gift cards. & so on

→ How will the payment method work with the checkout process?

→ When can user post a review & a rating

→ How are useful reviews tracked & less useful reviews deprioritized.

Design pattern to be used

→ Factory design pattern

→ Strategy design pattern

Requirement Collection for Amazon Online Shopping

R1: A customer can either be an authenticated user or a guest. The authenticated user is one with a registered account on the Amzon online shopping, whereas a guest does not have a registered account.

R2: A authenticated user should be able to buy, sell & search the products via the product name or category. guest is only able to search for products.

R3: A product can have multiple reviews & rating from multiple customers.

R4: An authenticated user should be able to add, remove, or modify product items in their shopping cart. the authenticated user can further checkout & buy the items.

R5: The authenticated user should be able to specify a shipping address at the time of order

R6: Payment can be made through credit cards, electronic banks transfer, or through cash on delivery

R7: An order can be canceled, only in the case that it hasn't been shipped

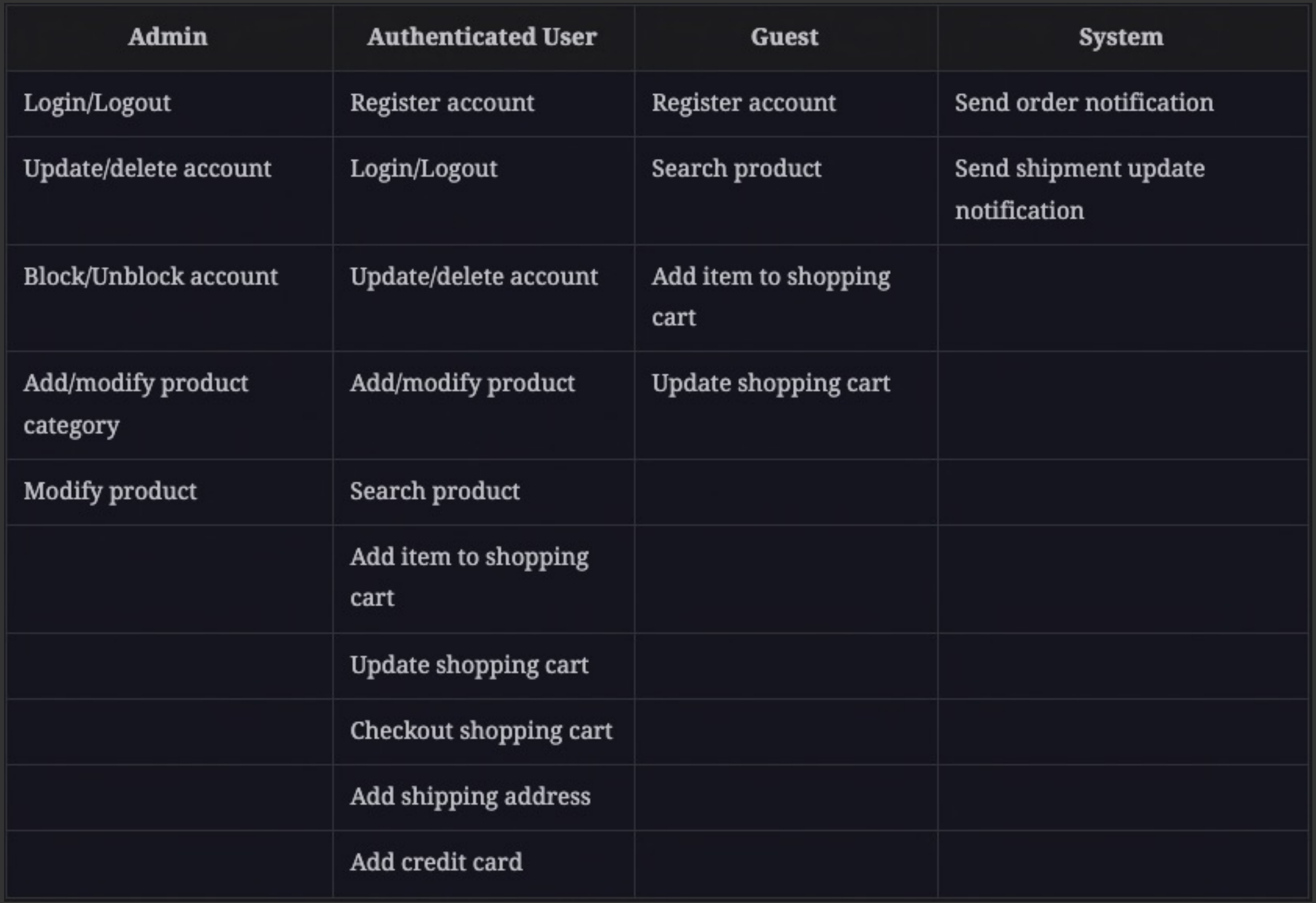
R8: Notifications are sent whenever there is a change in the order or shipping status.

R9: Shipping can be tracked to see the current status & the estimated time of arrival for the order

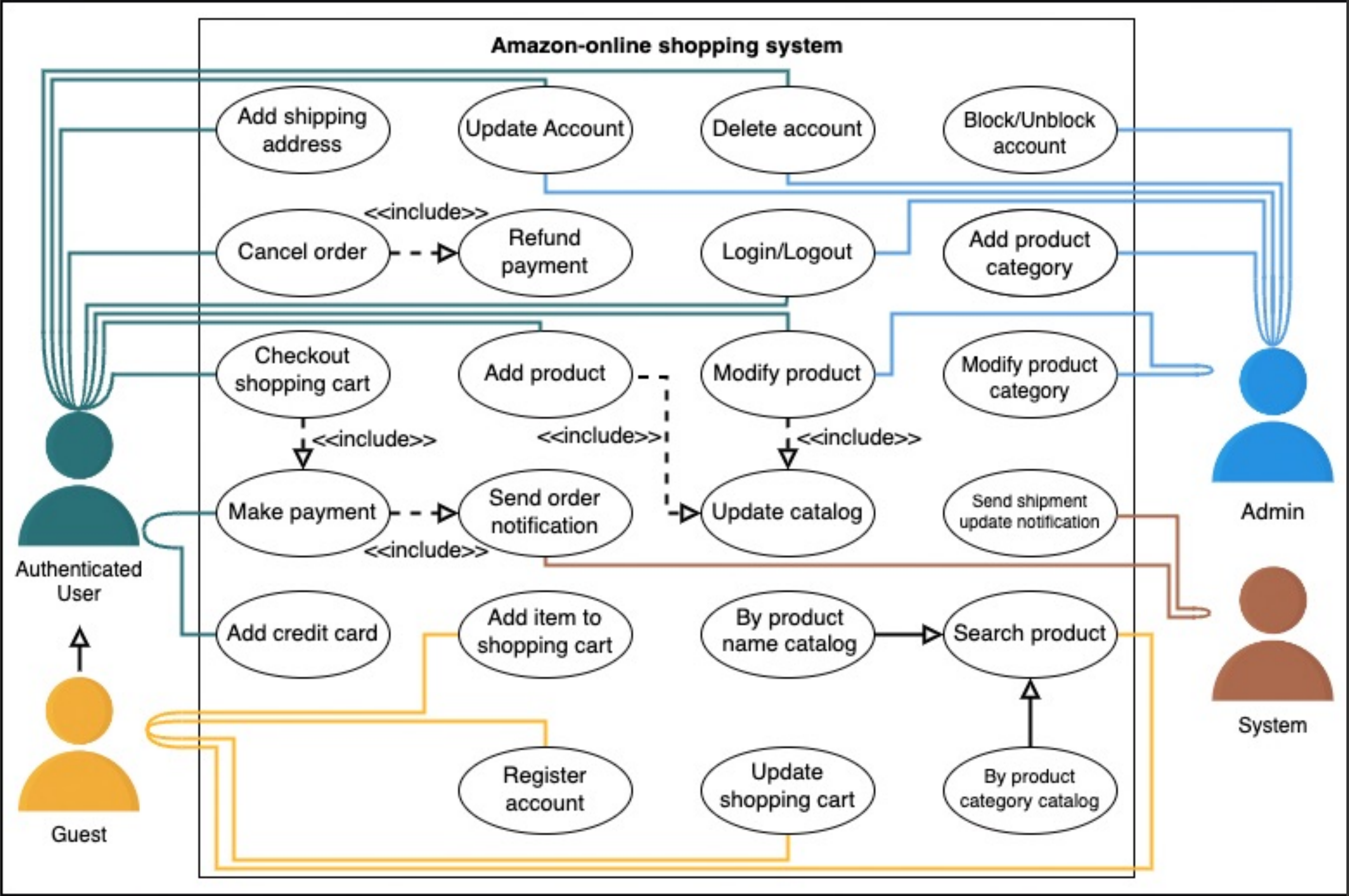
R10: An admin should exist that can add, modify or remove product categories & block users.

Primary Actors → Authenticated user
→ Guest

Secondary Actors → Admin
→ System

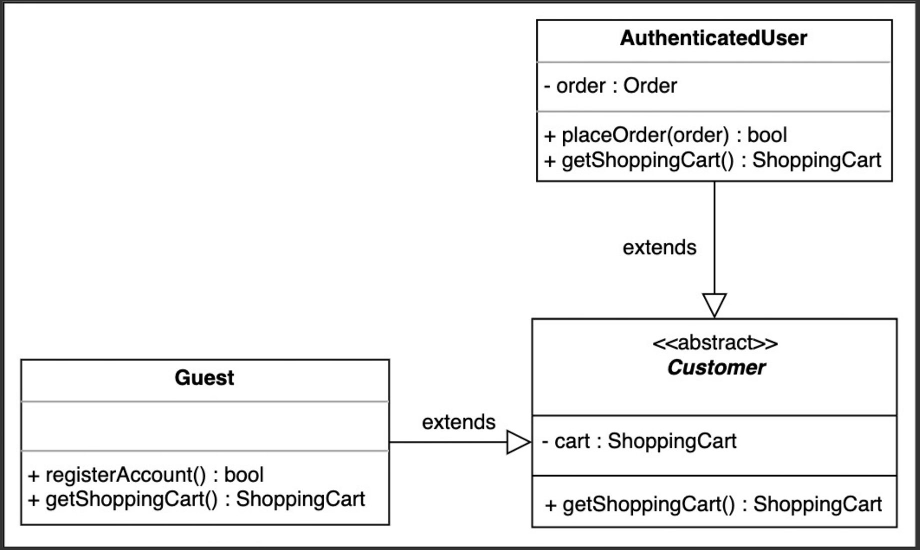


Admin	Authenticated User	Guest	System
Login/Logout	Register account	Register account	Send order notification
Update/delete account	Login/Logout	Search product	Send shipment update notification
Block/Unblock account	Update/delete account	Add item to shopping cart	
Add/modify product category	Add/modify product	Update shopping cart	
Modify product	Search product		
	Add item to shopping cart		
	Update shopping cart		
	Checkout shopping cart		
	Add shipping address		
	Add credit card		


Amazon-online shopping system
Add shipping address
Update Account
Delete account
Block/Unblock account
Cancel order
<<include>>
Refund payment
Login/Logout
Add product category
Checkout shopping cart
Add product
Modify product
Modify product category
<<include>>
<<include>>
<<include>>
Make payment
Send order notification
Update catalog
Send shipment update notification
<<include>>
Add credit card
Add item to shopping cart
By product name catalog
Search product
Register account
Update shopping cart
By product category catalog
Authenticated User
Guest
Admin
System

Class Diagram for Amazon Online Shopping

1. Customer



AuthenticatedUser

- order : Order

+ placeOrder(order) : bool
+ getShoppingCart() : ShoppingCart

extends

<<abstract>>
Customer

- cart : ShoppingCart

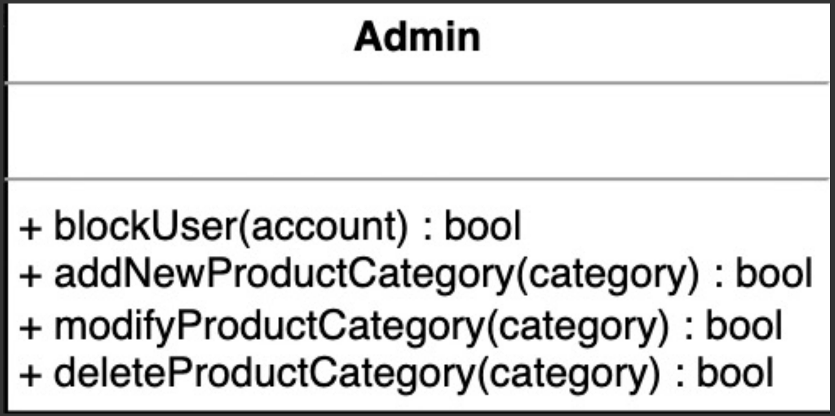
+ getShoppingCart() : ShoppingCart

extends

Guest

+ registerAccount() : bool
+ getShoppingCart() : ShoppingCart

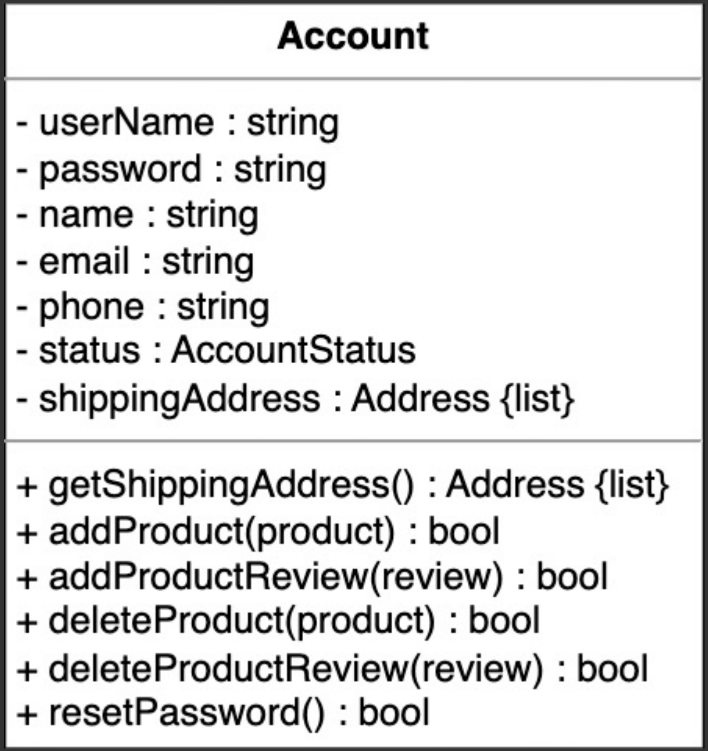
2. Admin



Admin

+ blockUser(account) : bool
+ addNewProductCategory(category) : bool
+ modifyProductCategory(category) : bool
+ deleteProductCategory(category) : bool

3. Account

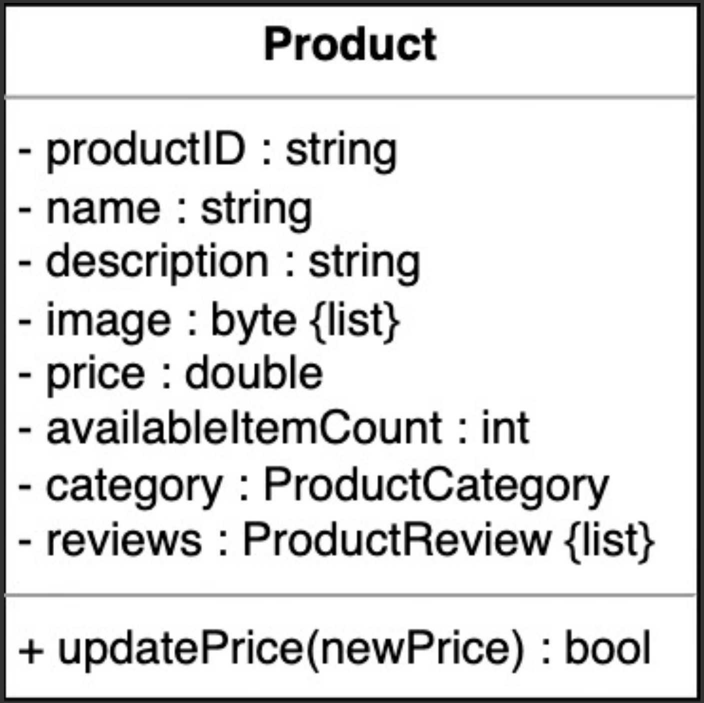


Account

- userName : string
- password : string
- name : string
- email : string
- phone : string
- status : AccountStatus
- shippingAddress : Address {list}

+ getShippingAddress() : Address {list}
+ addProduct(product) : bool
+ addProductReview(review) : bool
+ deleteProduct(product) : bool
+ deleteProductReview(review) : bool
+ resetPassword() : bool

4. Product

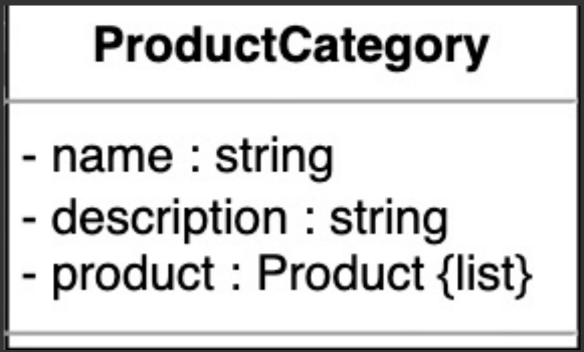


Product

- productID : string
- name : string
- description : string
- image : byte {list}
- price : double
- availableItemCount : int
- category : ProductCategory
- reviews : ProductReview {list}

+ updatePrice(newPrice) : bool

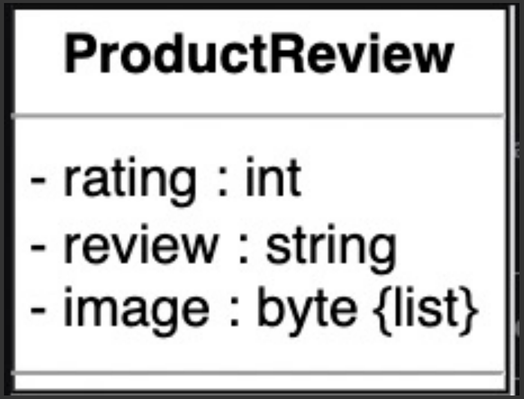
5. Product Category



ProductCategory

- name : string
- description : string
- product : Product {list}

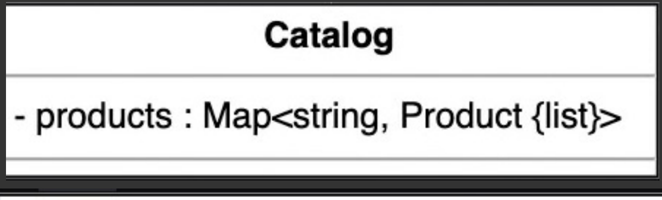
6. Product Review



ProductReview

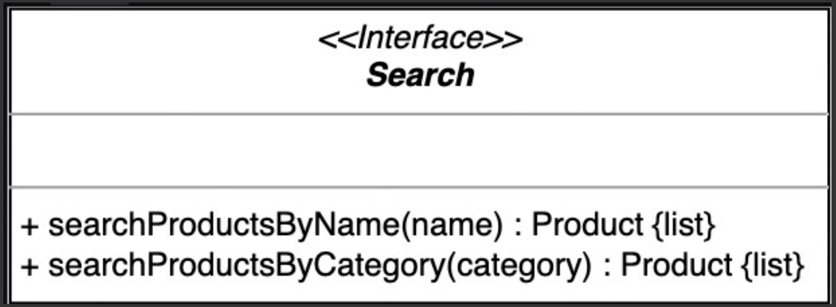
- rating : int
- review : string
- image : byte {list}

7. Catalog & Search



Catalog

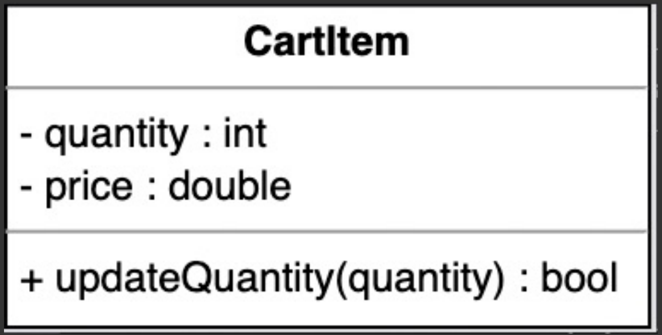
- products : Map<string, Product {list}>



<<Interface>>
Search

+ searchProductsByName(name) : Product {list}
+ searchProductsByCategory(category) : Product {list}

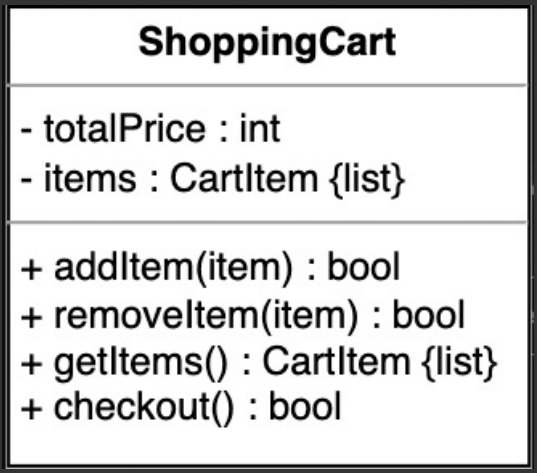
8. Cart Item & Shopping Cart



CartItem

- quantity : int
- price : double

+ updateQuantity(quantity) : bool

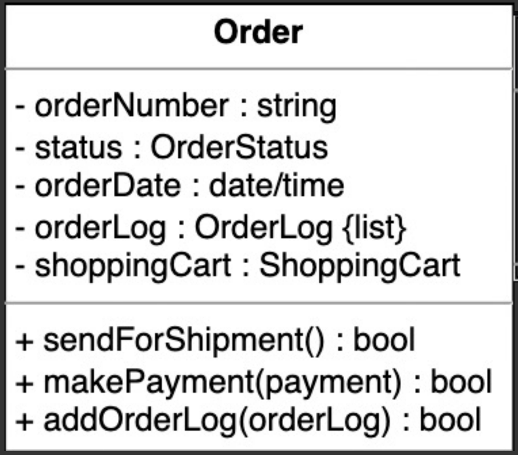


ShoppingCart

- totalPrice : int
- items : CartItem {list}

+ addItem(item) : bool
+ removeItem(item) : bool
+ getItems() : CartItem {list}
+ checkout() : bool

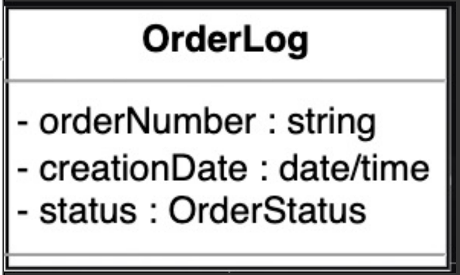
9. Order & Order log



Order

- orderNumber : string
- status : OrderStatus
- orderDate : date/time
- orderLog : OrderLog {list}
- shoppingCart : ShoppingCart

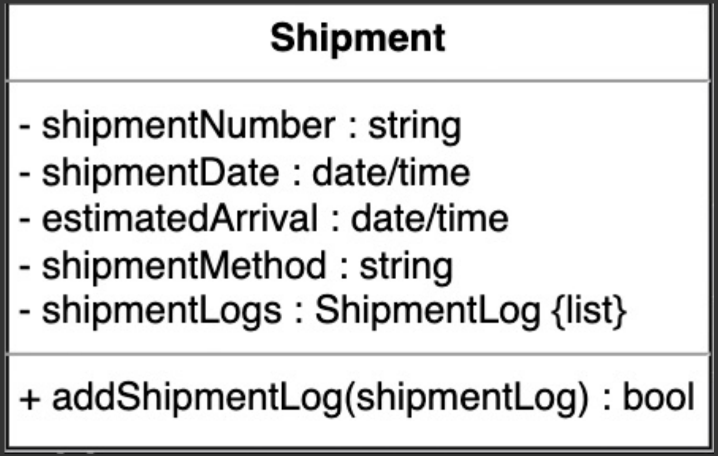
+ sendForShipment() : bool
+ makePayment(payment) : bool
+ addOrderLog(orderLog) : bool



OrderLog

- orderNumber : string
- creationDate : date/time
- status : OrderStatus

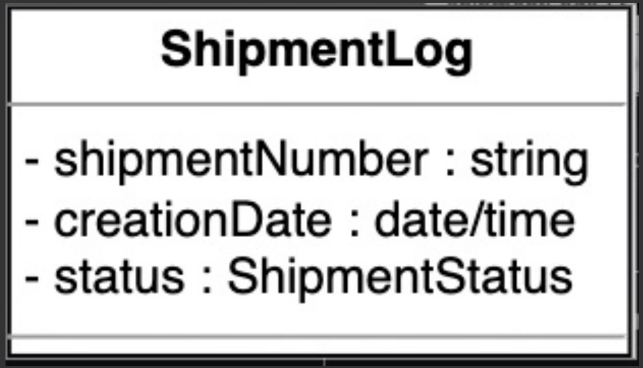
10. Shipment & Shipment log



Shipment

- shipmentNumber : string
- shipmentDate : date/time
- estimatedArrival : date/time
- shipmentMethod : string
- shipmentLogs : ShipmentLog {list}

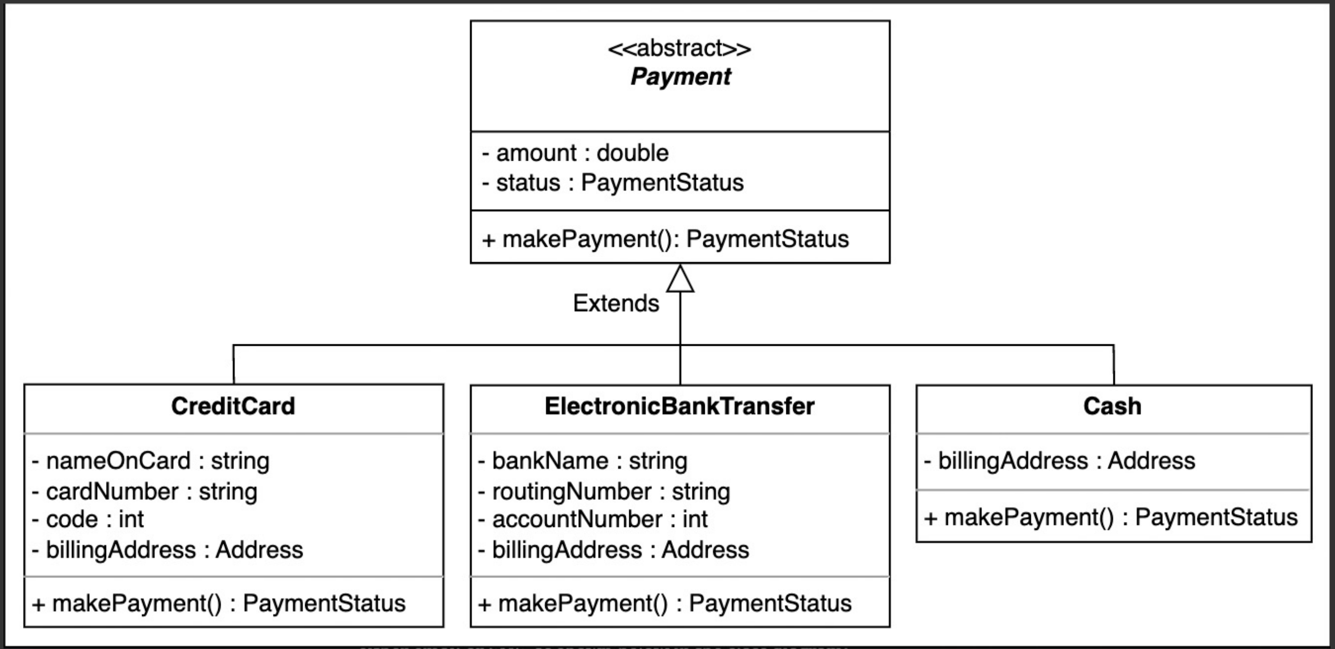
+ addShipmentLog(shipmentLog) : bool



ShipmentLog

- shipmentNumber : string
- creationDate : date/time
- status : ShipmentStatus

11 Payment



<<abstract>>
Payment

- amount : double
- status : PaymentStatus

+ makePayment(): PaymentStatus

Extends

CreditCard

- nameOnCard : string
- cardNumber : string
- code : int
- billingAddress : Address

+ makePayment() : PaymentStatus

ElectronicBankTransfer

- bankName : string
- routingNumber : string
- accountNumber : int
- billingAddress : Address

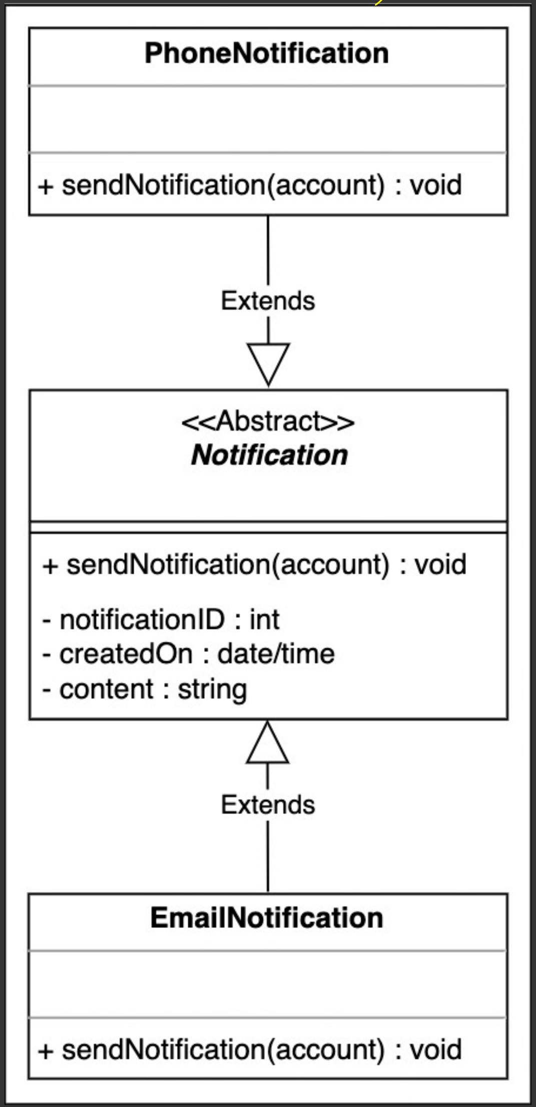
+ makePayment() : PaymentStatus

Cash

- billingAddress : Address

+ makePayment() : PaymentStatus

12. Notification



PhoneNotification

+ sendNotification(account) : void

Extends

<<Abstract>>
Notification

+ sendNotification(account) : void

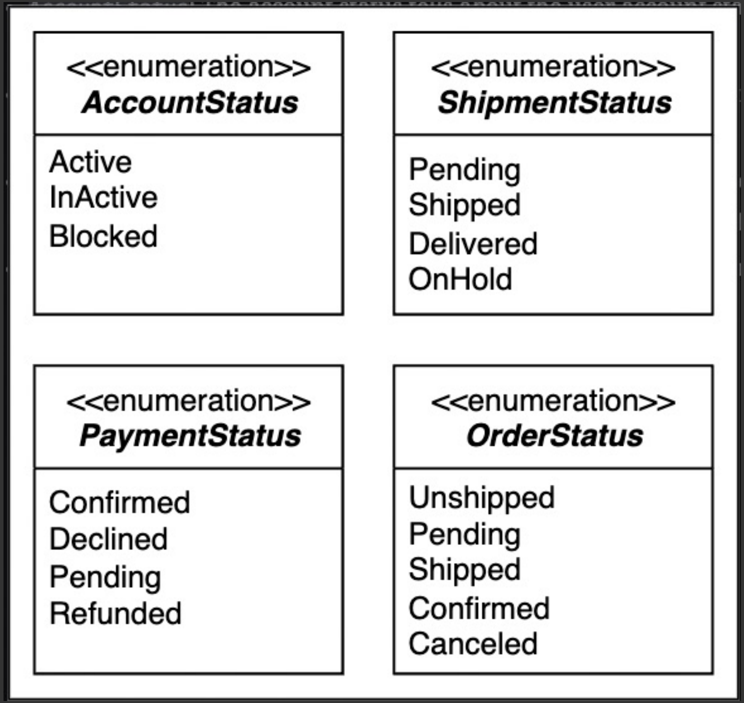
- notificationID : int
- createdOn : date/time
- content : string

Extends

EmailNotification

+ sendNotification(account) : void

13. Enumerations



<<enumeration>>
AccountStatus

Active
InActive
Blocked

<<enumeration>>
ShipmentStatus

Pending
Shipped
Delivered
OnHold

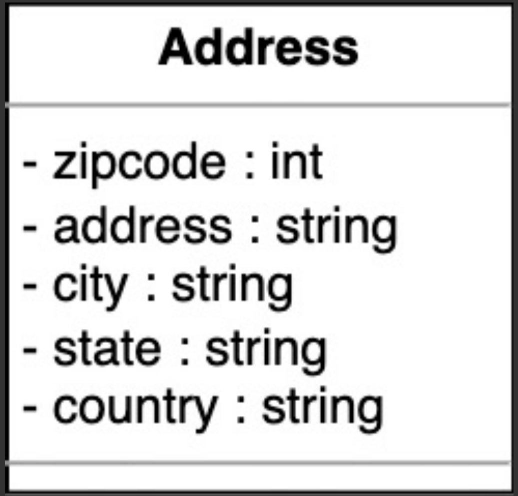
<<enumeration>>
PaymentStatus

Confirmed
Declined
Pending
Refunded

<<enumeration>>
OrderStatus

Unshipped
Pending
Shipped
Confirmed
Canceled

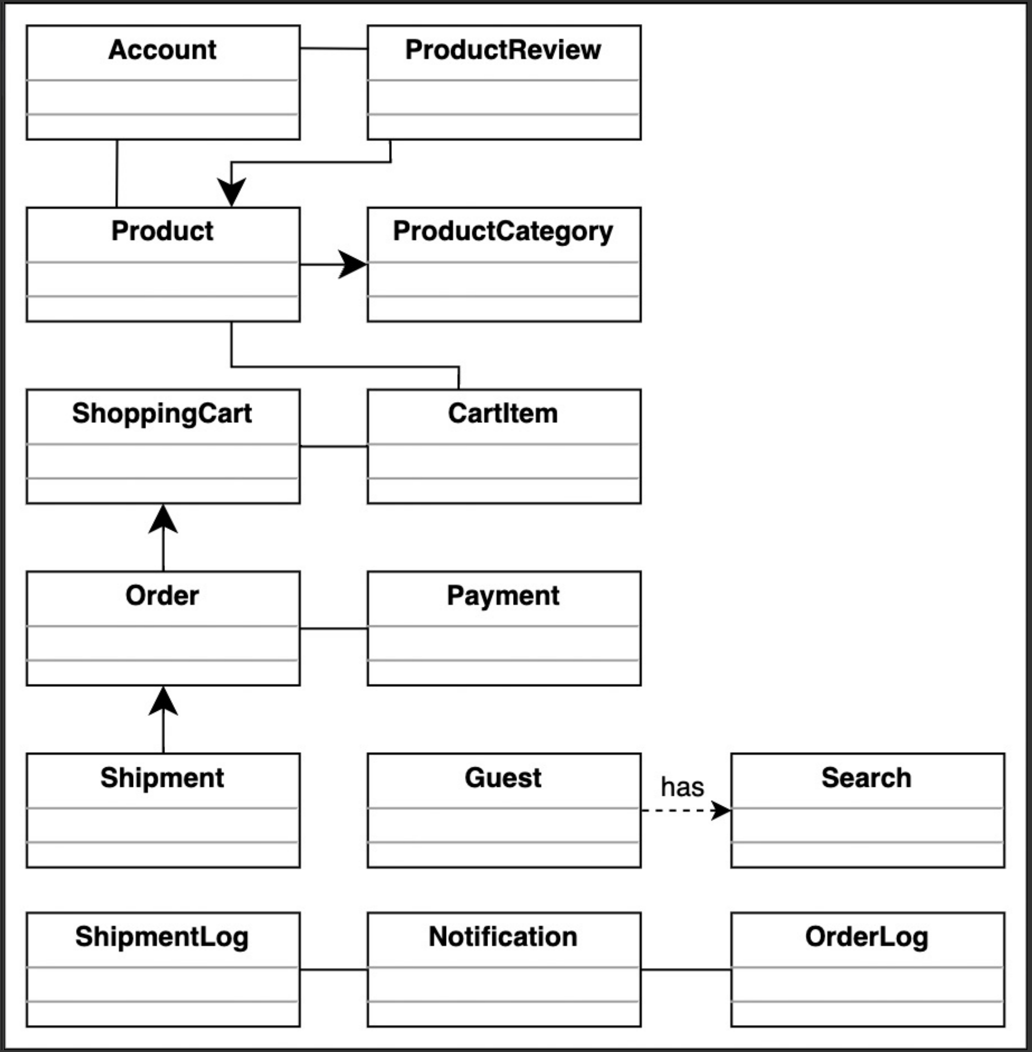
14. Address



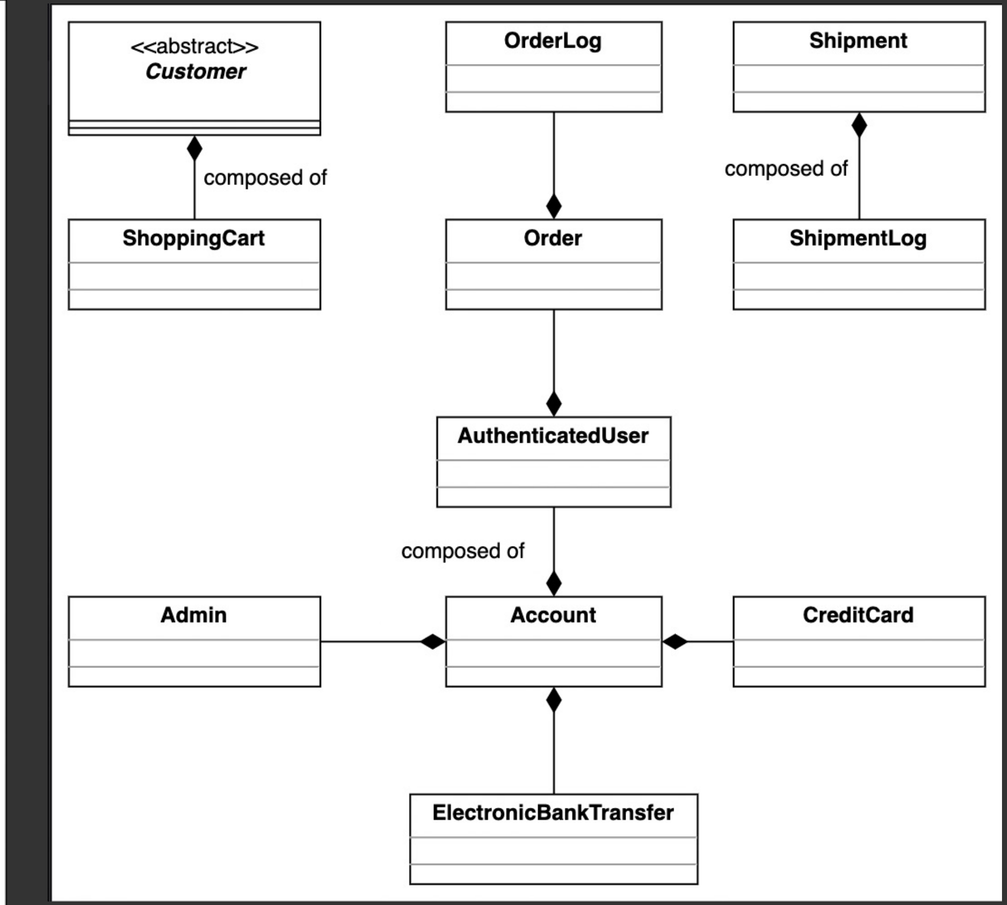
Address

- zipcode : int
- address : string
- city : string
- state : string
- country : string

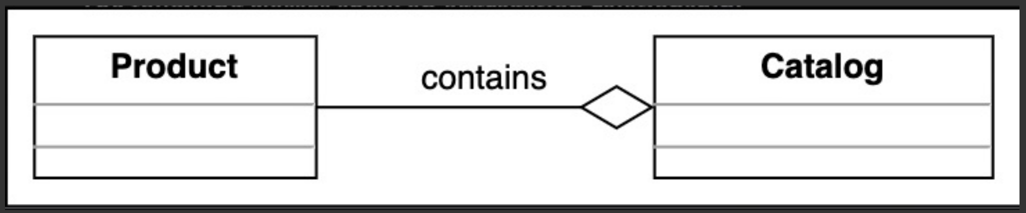
Associations



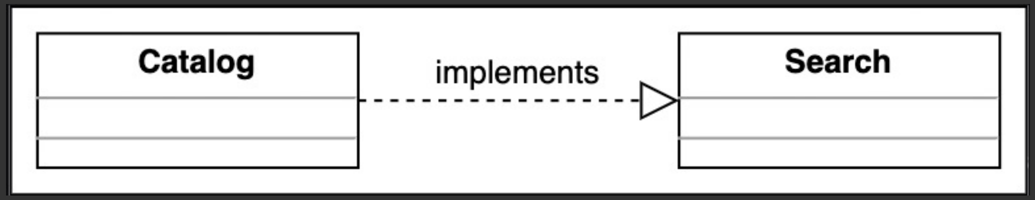
Composition



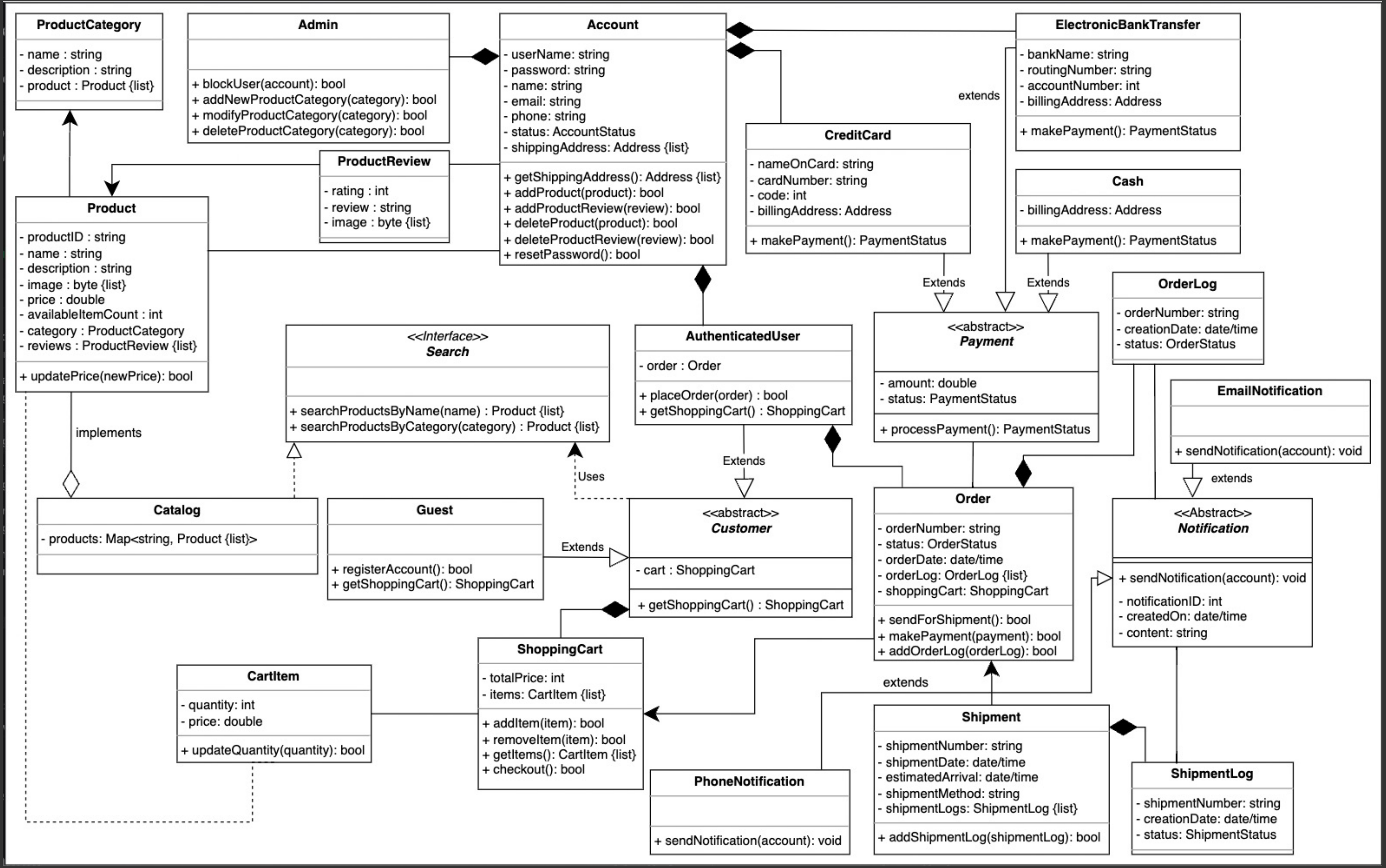
Aggregation



Generalization



Class Diagram



ProductCategory
- name : string
- description : string
- product : Product {list}

Admin
+ blockUser(account): bool
+ addNewProductCategory(category): bool
+ modifyProductCategory(category): bool
+ deleteProductCategory(category): bool

Account
- userName: string
- password: string
- name: string
- email: string
- phone: string
- status: AccountStatus
- shippingAddress: Address {list}

+ getShippingAddress(): Address {list}
+ addProduct(product): bool
+ addProductReview(review): bool
+ deleteProduct(product): bool
+ deleteProductReview(review): bool
+ resetPassword(): bool

ElectronicBankTransfer
- bankName: string
- routingNumber: string
- accountNumber: int
- billingAddress: Address

+ makePayment(): PaymentStatus

CreditCard
- nameOnCard: string
- cardNumber: string
- code: int
- billingAddress: Address

+ makePayment(): PaymentStatus

Cash
- billingAddress: Address

+ makePayment(): PaymentStatus

ProductReview
- rating : int
- review : string
- image : byte {list}

Product
- productID : string
- name : string
- description : string
- image : byte {list}
- price : double
- availableItemCount : int
- category : ProductCategory
- reviews : ProductReview {list}

+ updatePrice(newPrice): bool

<<Interface>>
Search

+ searchProductsByName(name) : Product {list}
+ searchProductsByCategory(category) : Product {list}

AuthenticatedUser
- order : Order

+ placeOrder(order) : bool
+ getShoppingCart() : ShoppingCart

<<abstract>>
Payment
- amount: double
- status: PaymentStatus

+ processPayment(): PaymentStatus

OrderLog
- orderNumber: string
- creationDate: date/time
- status: OrderStatus

EmailNotification

+ sendNotification(account): void

Catalog
- products: Map<string, Product {list}>

Guest

+ registerAccount(): bool
+ getShoppingCart(): ShoppingCart

<<abstract>>
Customer
- cart : ShoppingCart

+ getShoppingCart() : ShoppingCart

Order
- orderNumber: string
- status: OrderStatus
- orderDate: date/time
- orderLog: OrderLog {list}
- shoppingCart: ShoppingCart

+ sendForShipment(): bool
+ makePayment(payment): bool
+ addOrderLog(orderLog): bool

<<Abstract>>
Notification

+ sendNotification(account): void
- notificationID: int
- createdOn: date/time
- content: string

CartItem
- quantity: int
- price: double

+ updateQuantity(quantity): bool

ShoppingCart
- totalPrice: int
- items: CartItem {list}

+ addItem(item): bool
+ removeItem(item): bool
+ getItems(): CartItem {list}
+ checkout(): bool

PhoneNotification

+ sendNotification(account): void

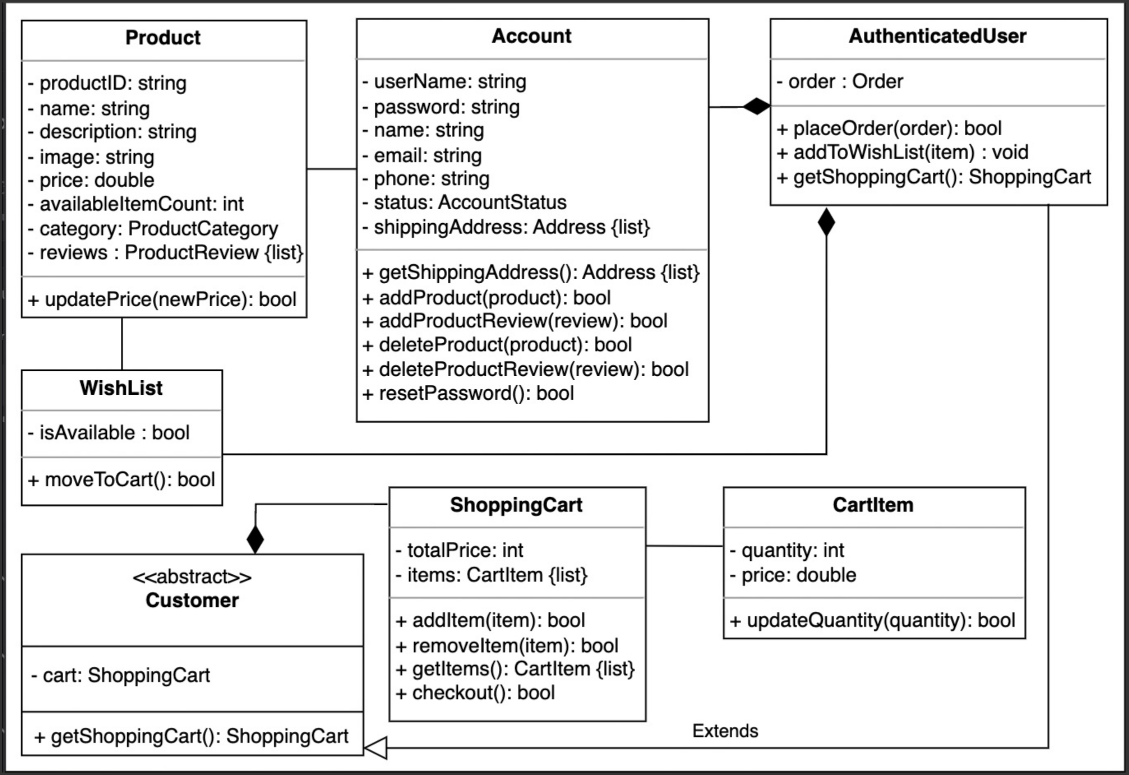
Shipment
- shipmentNumber: string
- shipmentDate: date/time
- estimatedArrival: date/time
- shipmentMethod: string
- shipmentLogs: ShipmentLog {list}

+ addShipmentLog(shipmentLog): bool

ShipmentLog
- shipmentNumber: string
- creationDate: date/time
- status: ShipmentStatus

Addional Requirement

1. Wish list



Product
- productID: string
- name: string
- description: string
- image: string
- price: double
- availableItemCount: int
- category: ProductCategory
- reviews : ProductReview {list}

+ updatePrice(newPrice): bool

Account
- userName: string
- password: string
- name: string
- email: string
- phone: string
- status: AccountStatus
- shippingAddress: Address {list}

+ getShippingAddress(): Address {list}
+ addProduct(product): bool
+ addProductReview(review): bool
+ deleteProduct(product): bool
+ deleteProductReview(review): bool
+ resetPassword(): bool

AuthenticatedUser
- order : Order

+ placeOrder(order): bool
+ addToWishList(item) : void
+ getShoppingCart(): ShoppingCart

WishList
- isAvailable : bool

+ moveToCart(): bool

ShoppingCart
- totalPrice: int
- items: CartItem {list}

+ addItem(item): bool
+ removeItem(item): bool
+ getItems(): CartItem {list}
+ checkout(): bool

CartItem
- quantity: int
- price: double

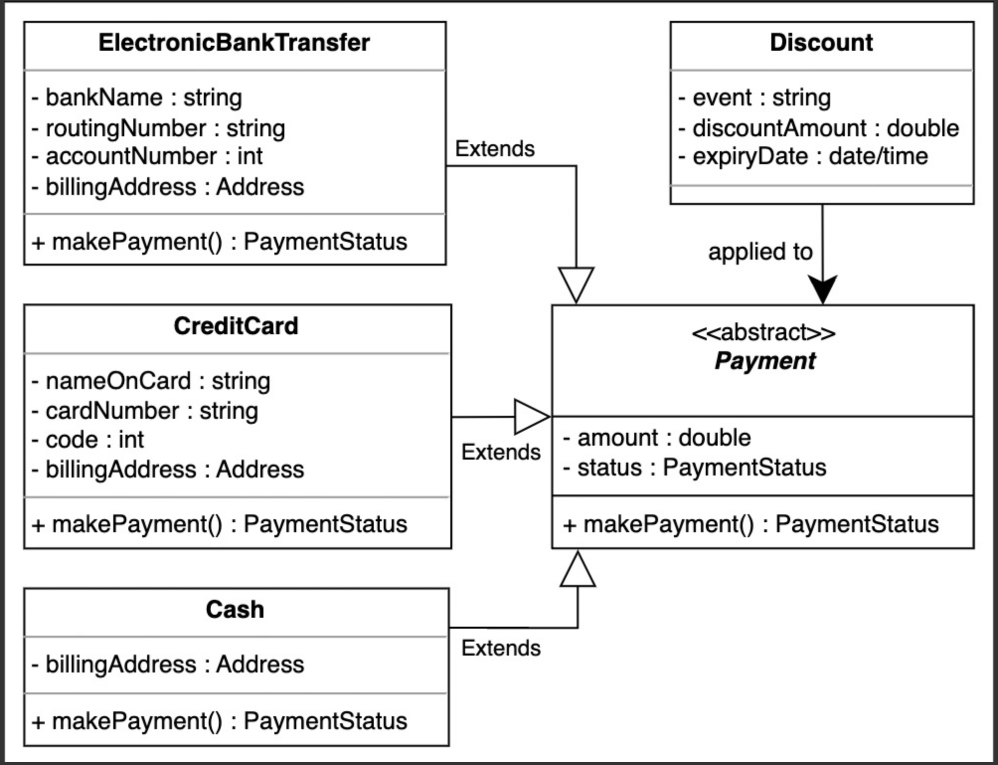
+ updateQuantity(quantity): bool

<<abstract>>
Customer
- cart: ShoppingCart

+ getShoppingCart(): ShoppingCart

Extends

2. Discount



ElectronicBankTransfer
- bankName : string
- routingNumber : string
- accountNumber : int
- billingAddress : Address

+ makePayment() : PaymentStatus

Discount
- event : string
- discountAmount : double
- expiryDate : date/time

applied to

Extends

CreditCard
- nameOnCard : string
- cardNumber : string
- code : int
- billingAddress : Address

+ makePayment() : PaymentStatus

Extends

<<abstract>>
Payment
- amount : double
- status : PaymentStatus

+ makePayment() : PaymentStatus

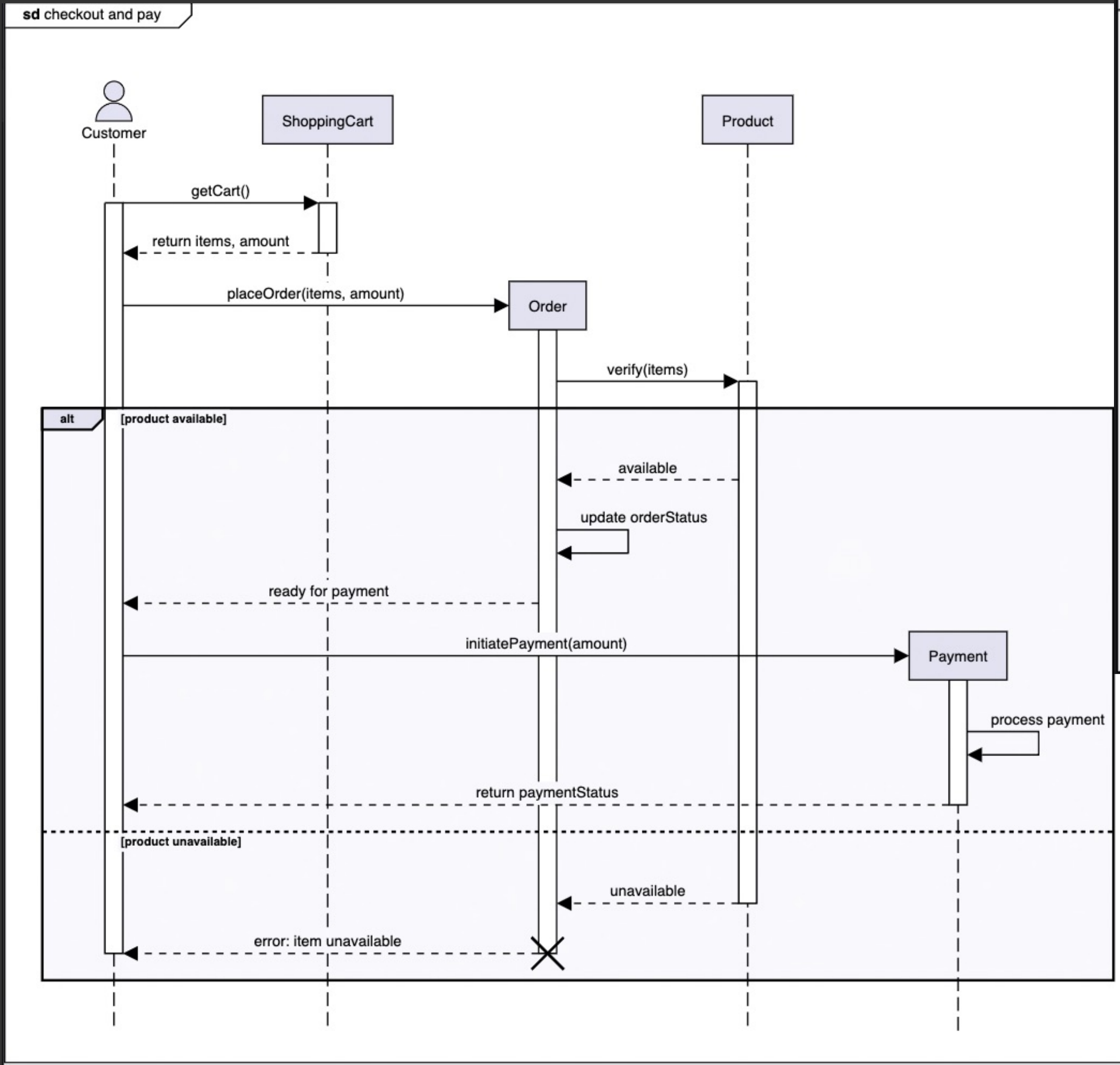
Cash
- billingAddress : Address

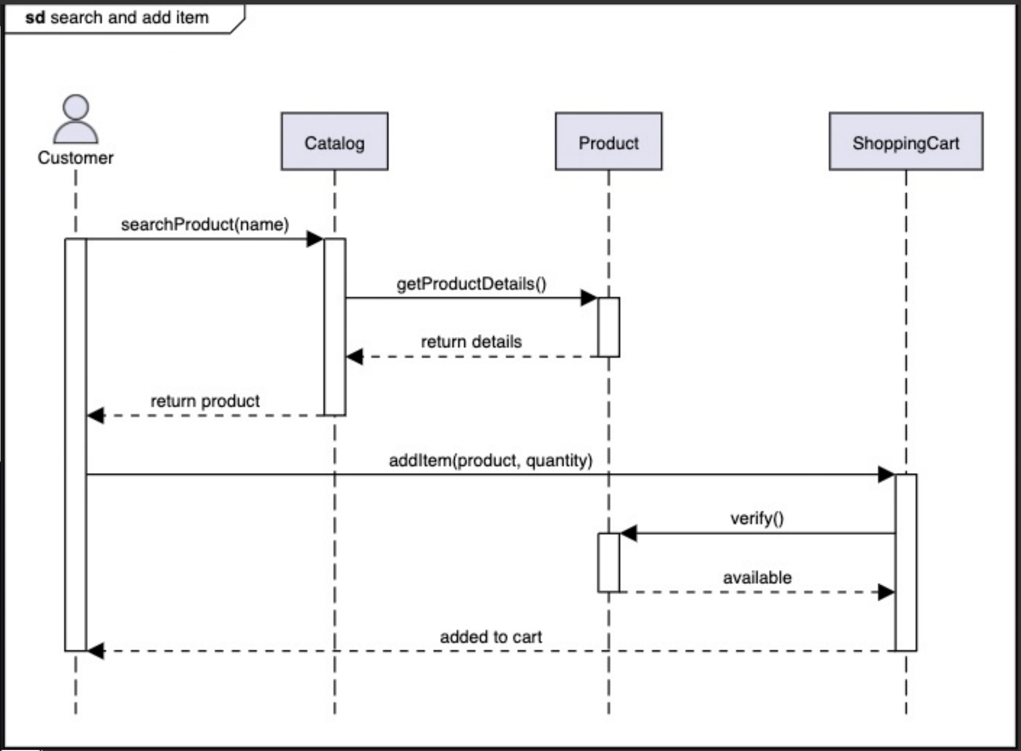
+ makePayment() : PaymentStatus

Extends

Sequence Diagram for Amazon Online Shopping

→ sequence diagram → checkout & pay

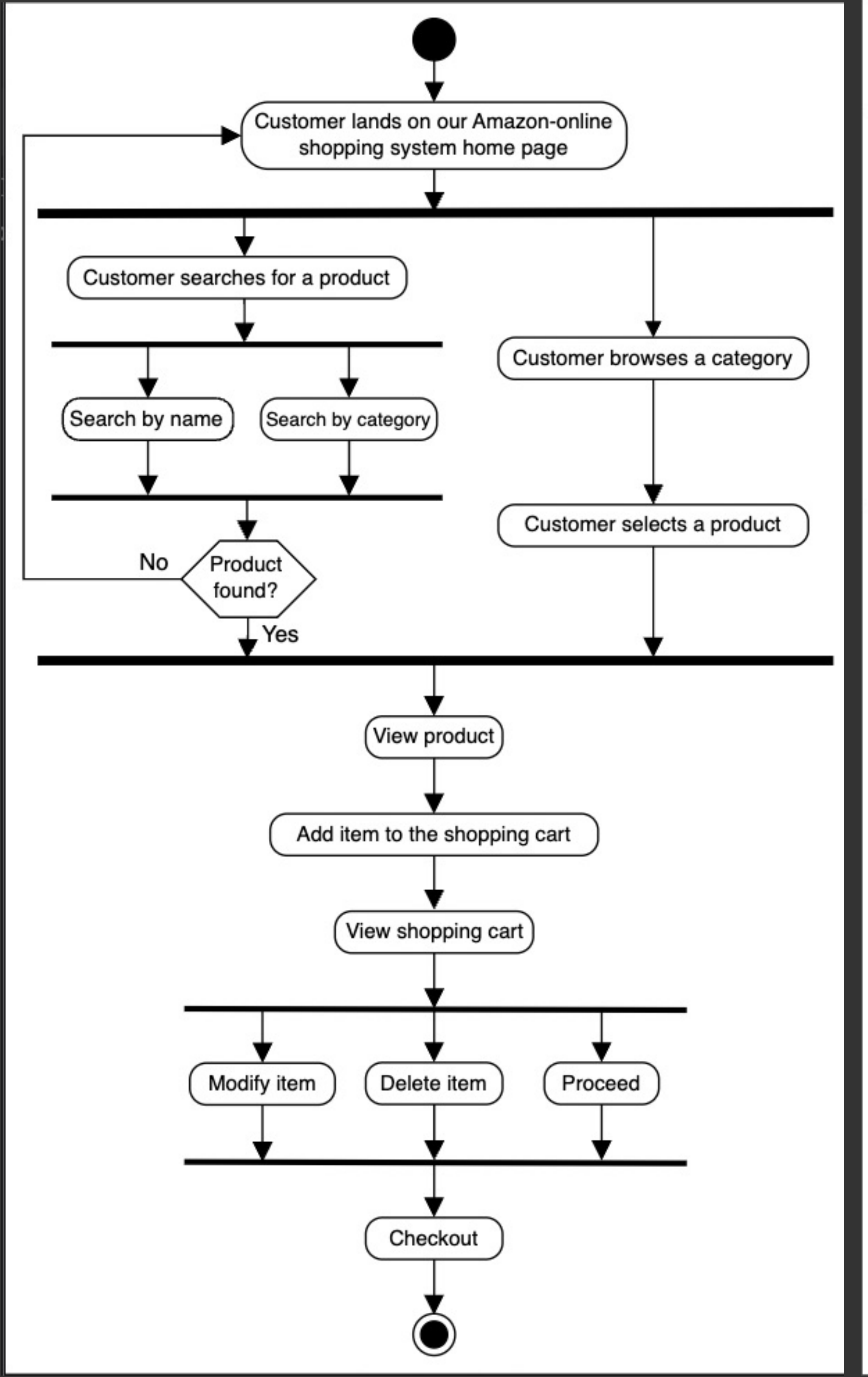


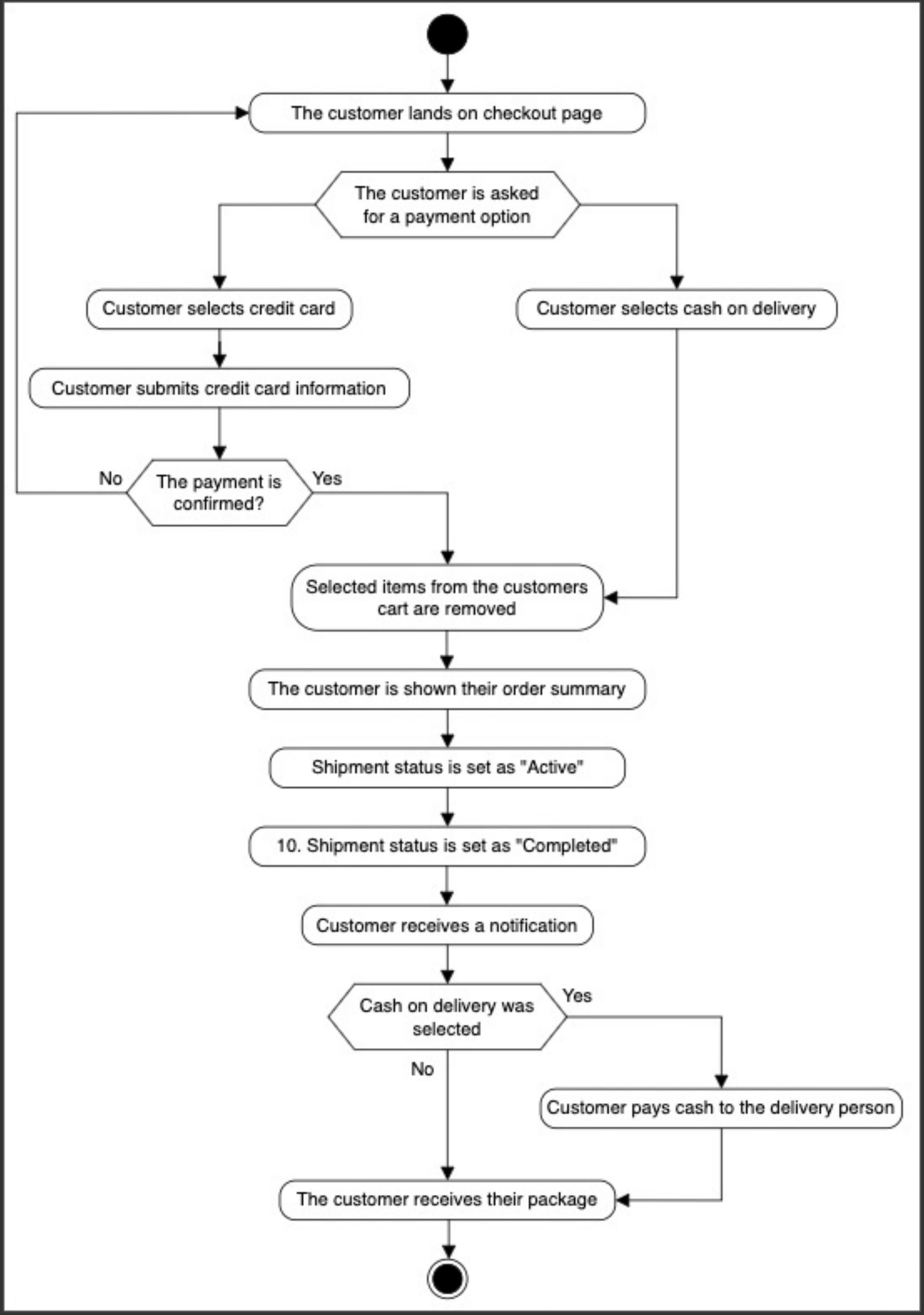


↓

search & add items

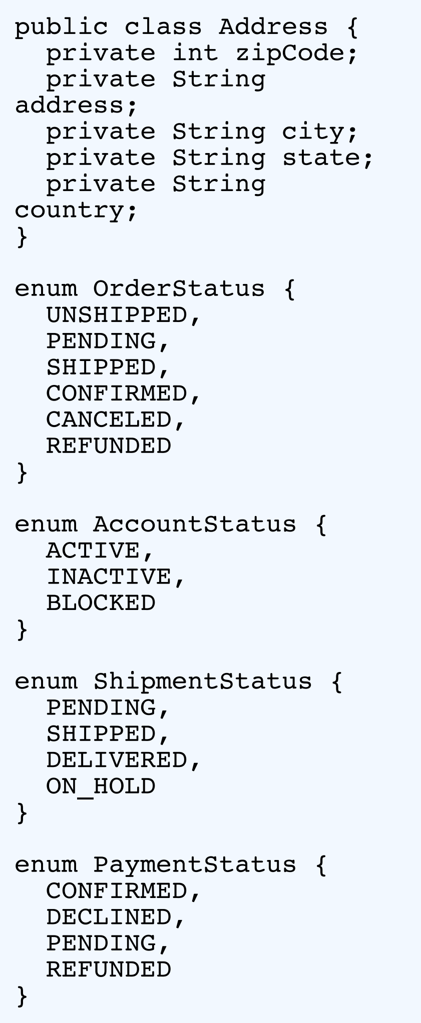
Activity Diagram for the Amazon Online Shopping System


Customer lands on our Amazon-online shopping system home page
Customer searches for a product
Customer browses a category
Search by name
Search by category
Customer selects a product
No
Product found?
Yes
View product
Add item to the shopping cart
View shopping cart
Modify item
Delete item
Proceed
Checkout


The customer lands on checkout page
The customer is asked for a payment option
Customer selects credit card
Customer selects cash on delivery
Customer submits credit card information
No
The payment is confirmed?
Yes
Selected items from the customers cart are removed
The customer is shown their order summary
Shipment status is set as "Active"
10. Shipment status is set as "Completed"
Customer receives a notification
Cash on delivery was selected
Yes
No
Customer pays cash to the delivery person
The customer receives their package

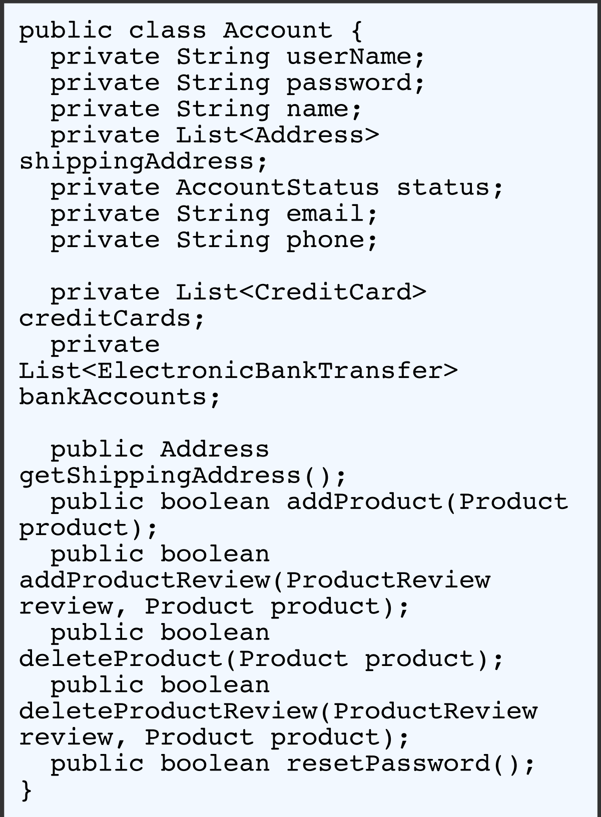
Code for Amazon Shopping System

1. Address & Enumeration



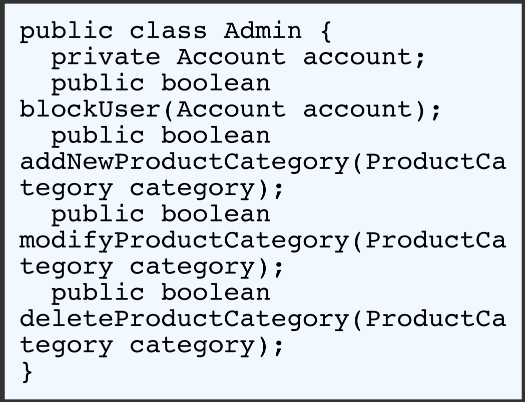
```
public class Address {
  private int zipCode;
  private String
address;
  private String city;
  private String state;
  private String
country;
}

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

enum AccountStatus {
  ACTIVE,
  INACTIVE,
  BLOCKED
}

enum ShipmentStatus {
  PENDING,
  SHIPPED,
  DELIVERED,
  ON_HOLD
}

enum PaymentStatus {
  CONFIRMED,
  DECLINED,
  PENDING,
  REFUNDED
}
```

2. Account



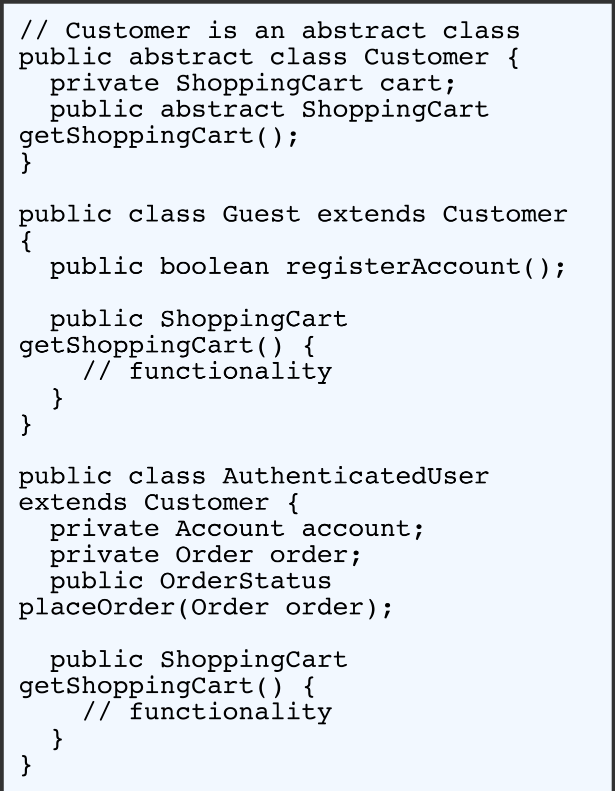
```
public class Account {
  private String userName;
  private String password;
  private String name;
  private List<Address>
shippingAddress;
  private AccountStatus status;
  private String email;
  private String phone;

  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();
}
```

3. Admin



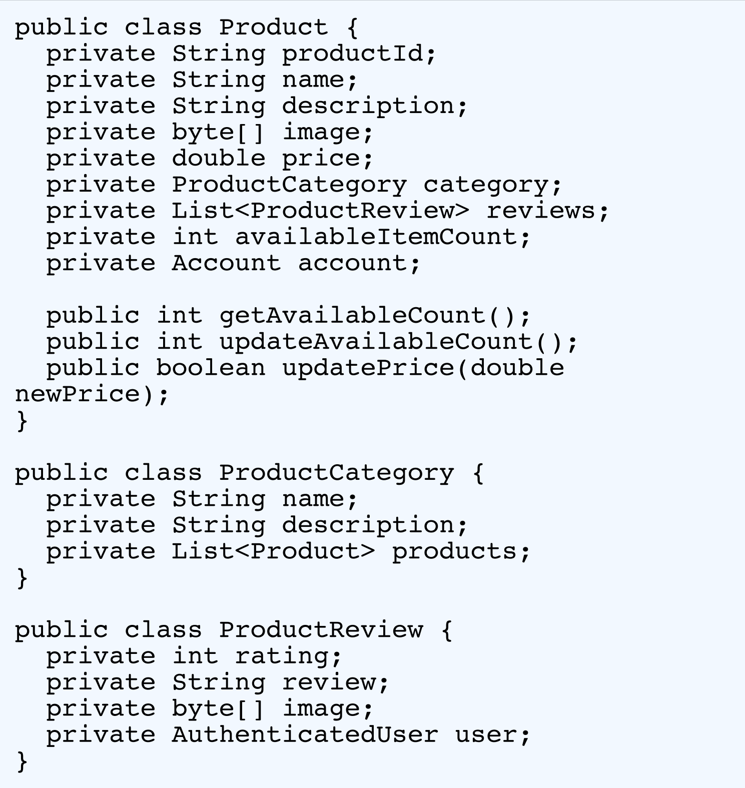
```
public class Admin {
  private Account account;
  public boolean
blockUser(Account account);
  public boolean
addNewProductCategory(ProductCa
tegory category);
  public boolean
modifyProductCategory(ProductCa
tegory category);
  public boolean
deleteProductCategory(ProductCa
tegory category);
}
```

4. Customer



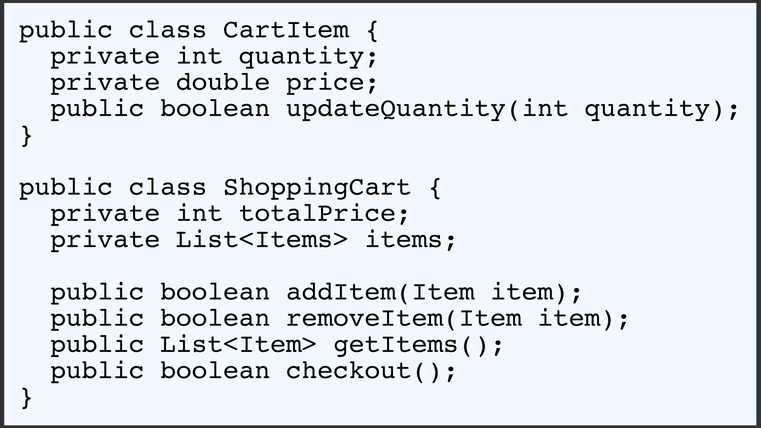
```
// Customer is an abstract class
public abstract class Customer {
  private ShoppingCart cart;
  public abstract ShoppingCart
getShoppingCart();
}

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
  }
}
```

5. Product, Product Category & Product Review



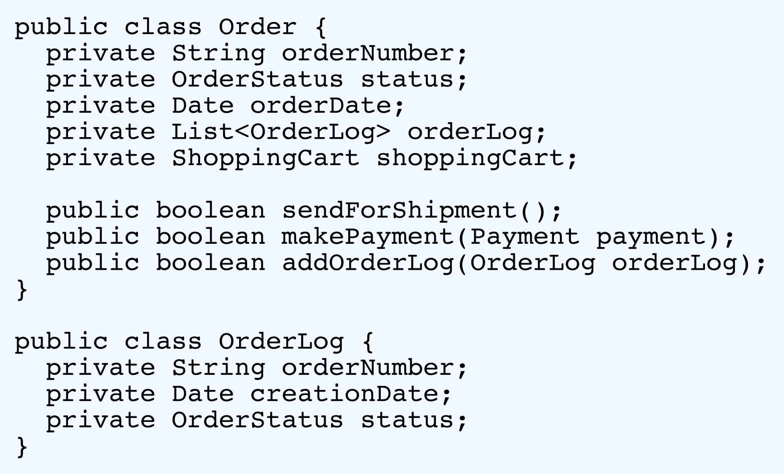
```
public class Product {
  private String productId;
  private String name;
  private String description;
  private byte[] image;
  private double price;
  private ProductCategory category;
  private List<ProductReview> reviews;
  private int availableItemCount;
  private Account account;

  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;
}
```

6. ShoppingCart & Cart Item



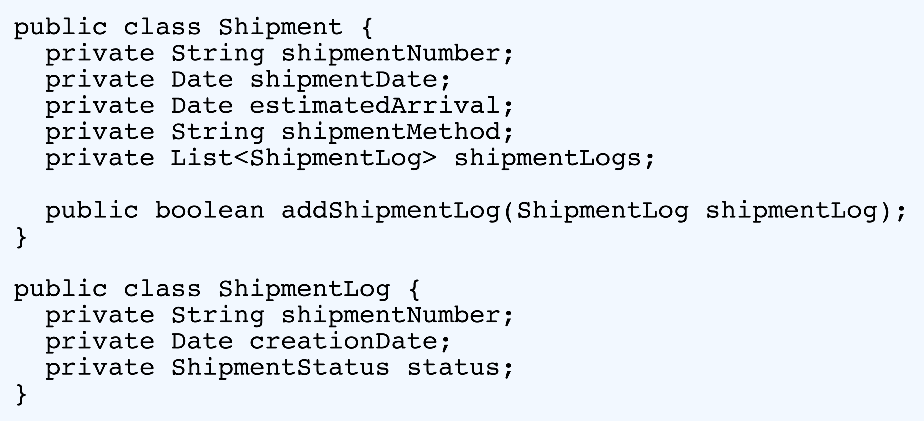
```
public class CartItem {
  private int quantity;
  private double price;
  public boolean updateQuantity(int quantity);
}

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();
}
```

7. Order & Order Log



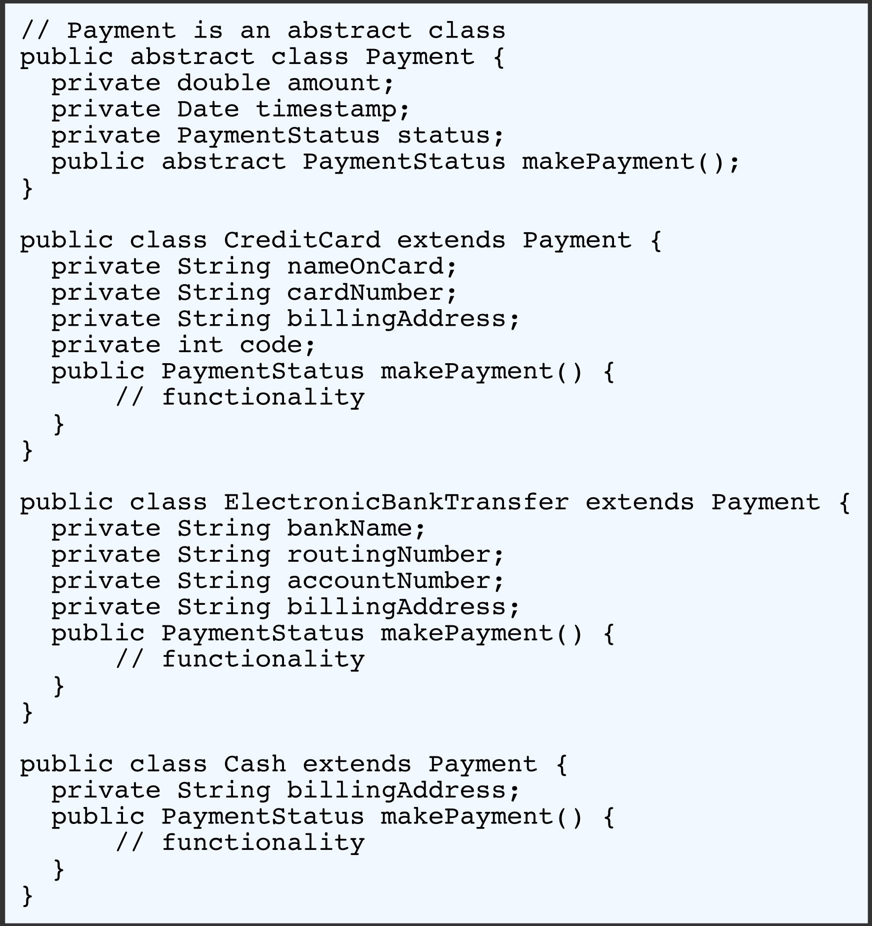
```
public class Order {
  private String orderNumber;
  private OrderStatus status;
  private Date orderDate;
  private List<OrderLog> orderLog;
  private ShoppingCart shoppingCart;

  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;
}
```

8. Shipment & Shipment Log



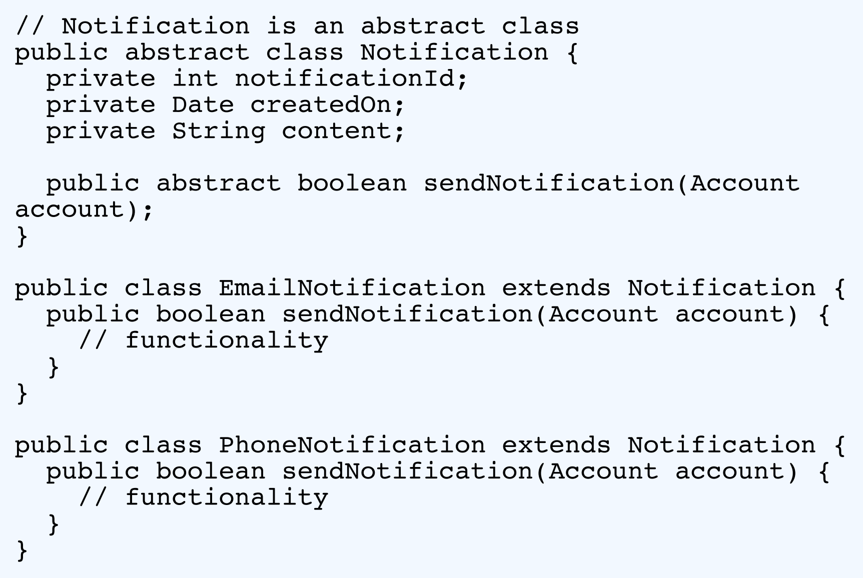
```
public class Shipment {
  private String shipmentNumber;
  private Date shipmentDate;
  private Date estimatedArrival;
  private String shipmentMethod;
  private List<ShipmentLog> shipmentLogs;

  public boolean addShipmentLog(ShipmentLog shipmentLog);
}

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

9. Payment



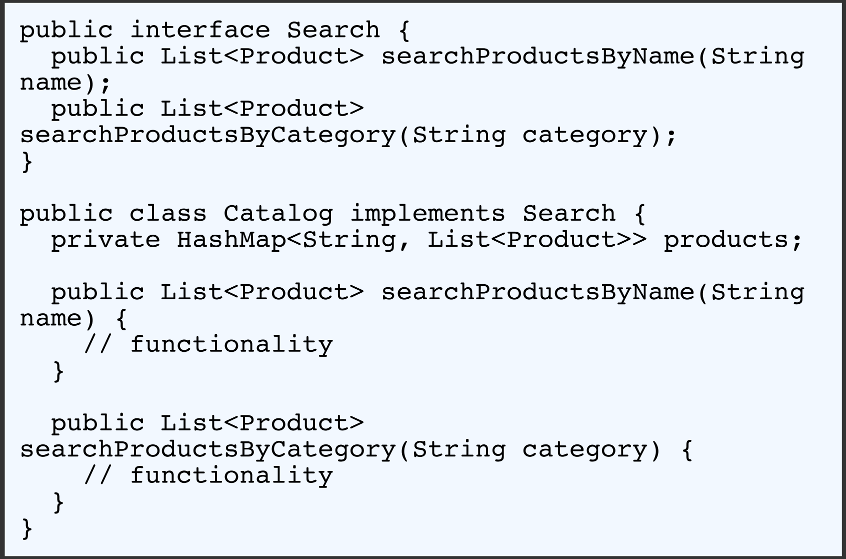
```
// Payment is an abstract class
public abstract class Payment {
  private double amount;
  private Date timestamp;
  private PaymentStatus status;
  public abstract PaymentStatus makePayment();
}

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;
  public PaymentStatus makePayment() {
      // functionality
  }
}
```

10. Notification



```
// 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
  }
}
```

11. Search & Catlog



```
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
  }
}
```