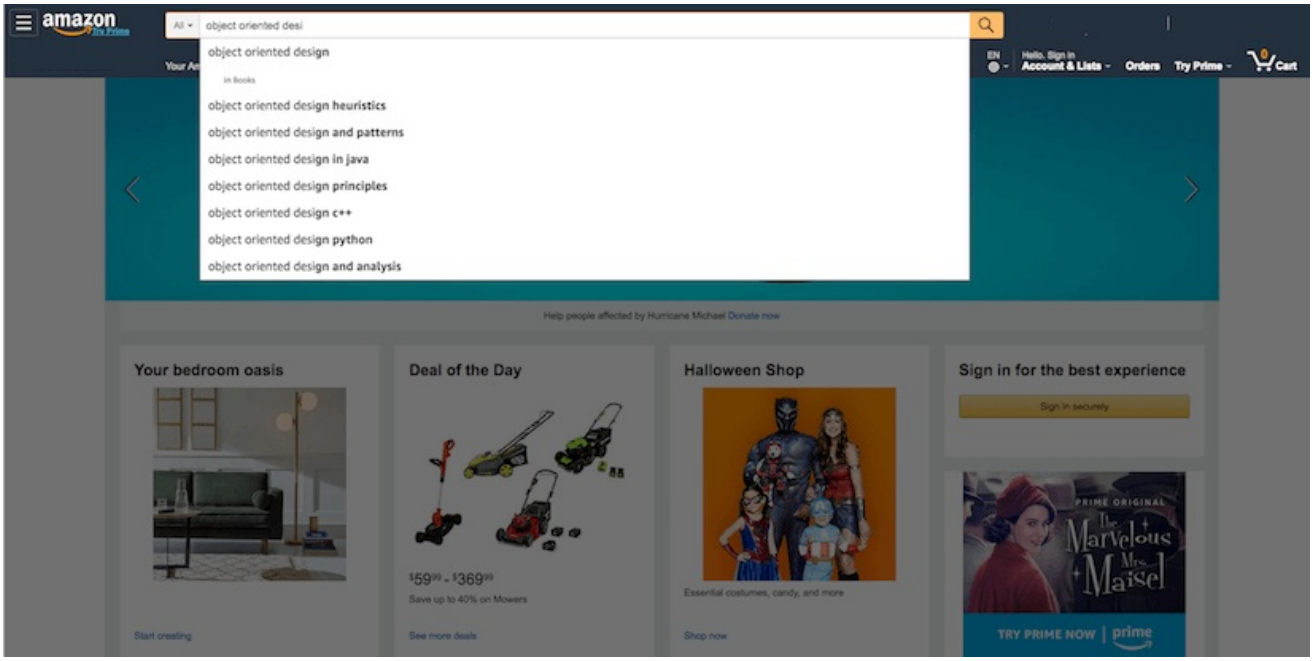
Design Amazon - Online Shopping System

Let's design an online retail store.

Amazon (amazon.com) is the world's largest online retailer. The company was originally a bookseller but has expanded to sell a wide variety of consumer goods and digital media. For the sake of this problem, we will focus on their online retail business where users can sell/buy their products.



Requirements and Goals of the System

We will be designing a system with the following requirements:

1. Users should be able to add new products to sell.
2. Users should be able to search for products by their name or category.
3. Users can search and view all the products, but they will have to become a registered member to buy a product.
4. Users should be able to add/remove/modify product items in their shopping cart.
5. Users can check out and buy items in the shopping cart.
6. Users can rate and add a review for a product.
7. The user should be able to specify a shipping address where their order will be delivered.
8. Users can cancel an order if it has not shipped.
9. Users should get notifications whenever there is a change in the order or shipping status.
10. Users should be able to pay through credit cards or electronic bank transfer.
11. Users should be able to track their shipment to see the current state of their order.

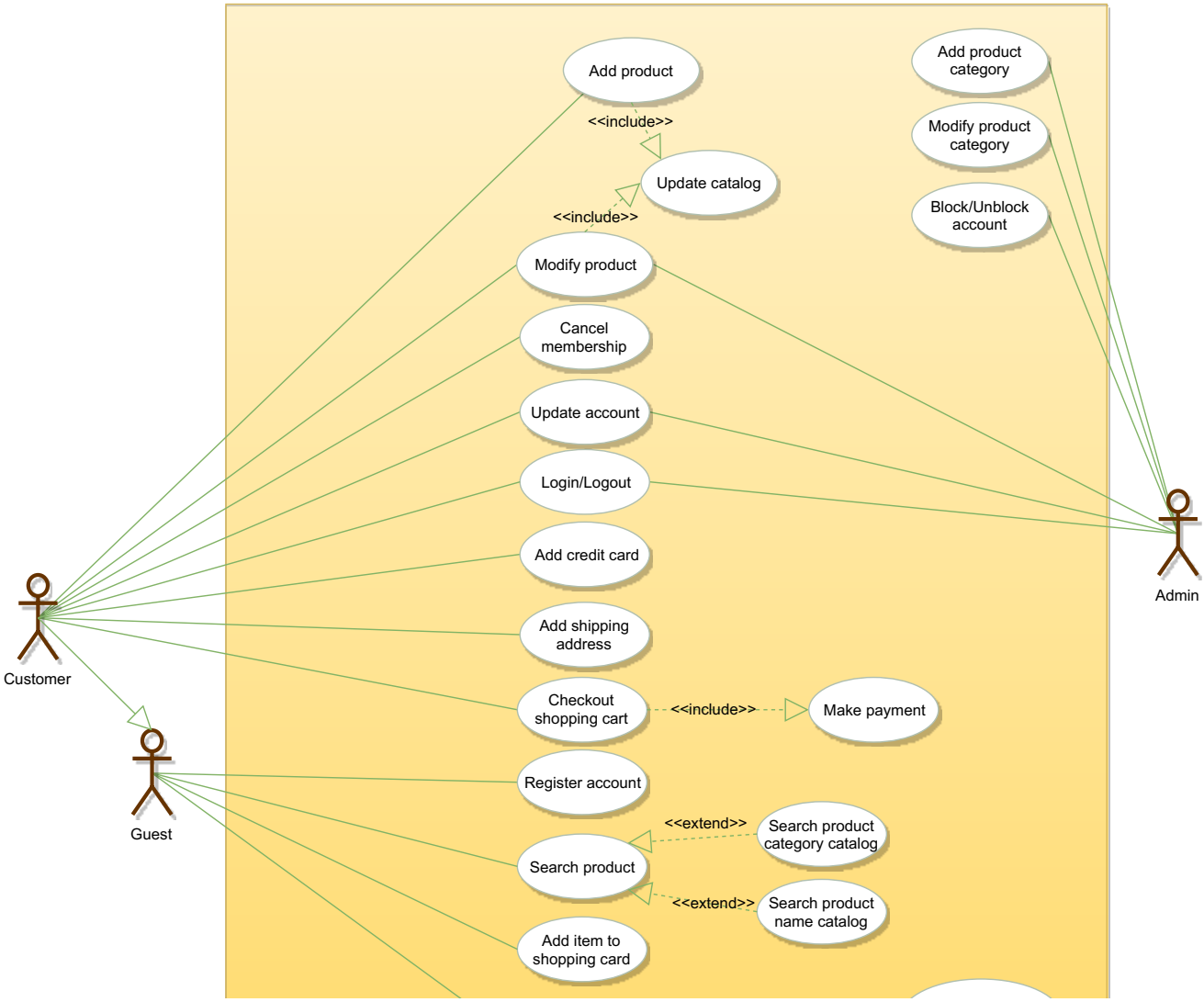
Use case Diagram

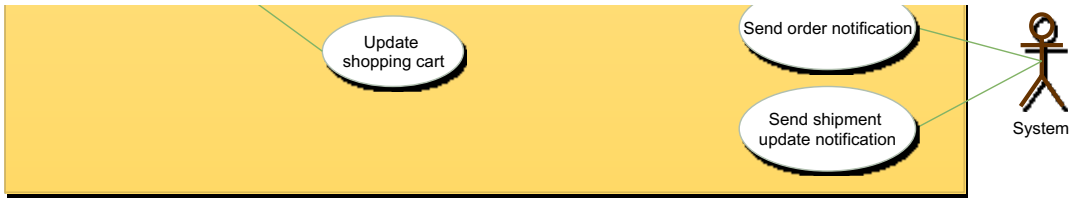
We have four main Actors in our system:

- **Admin:** Mainly responsible for account management and adding or modifying new product categories.
- **Guest:** All guests can search the catalog, add/remove items to the shopping cart, as well as become registered members.
- **Member:** Members can perform all the activities that guests can, in addition to which, they can place orders and add new products to sell.
- **System:** Mainly responsible for sending notifications for orders and shipping updates.

Here are the top use cases of the Online Shopping System:

1. Add/update products; whenever a product is added or modified, we will update the catalog.
2. Search for products by their name or category.
3. Add/remove product items in the shopping cart.
4. Check-out to buy product items in the shopping cart.
5. Make a payment to place an order.
6. Add a new product category.
7. Send notifications to members with shipment updates.



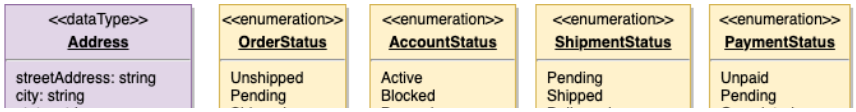


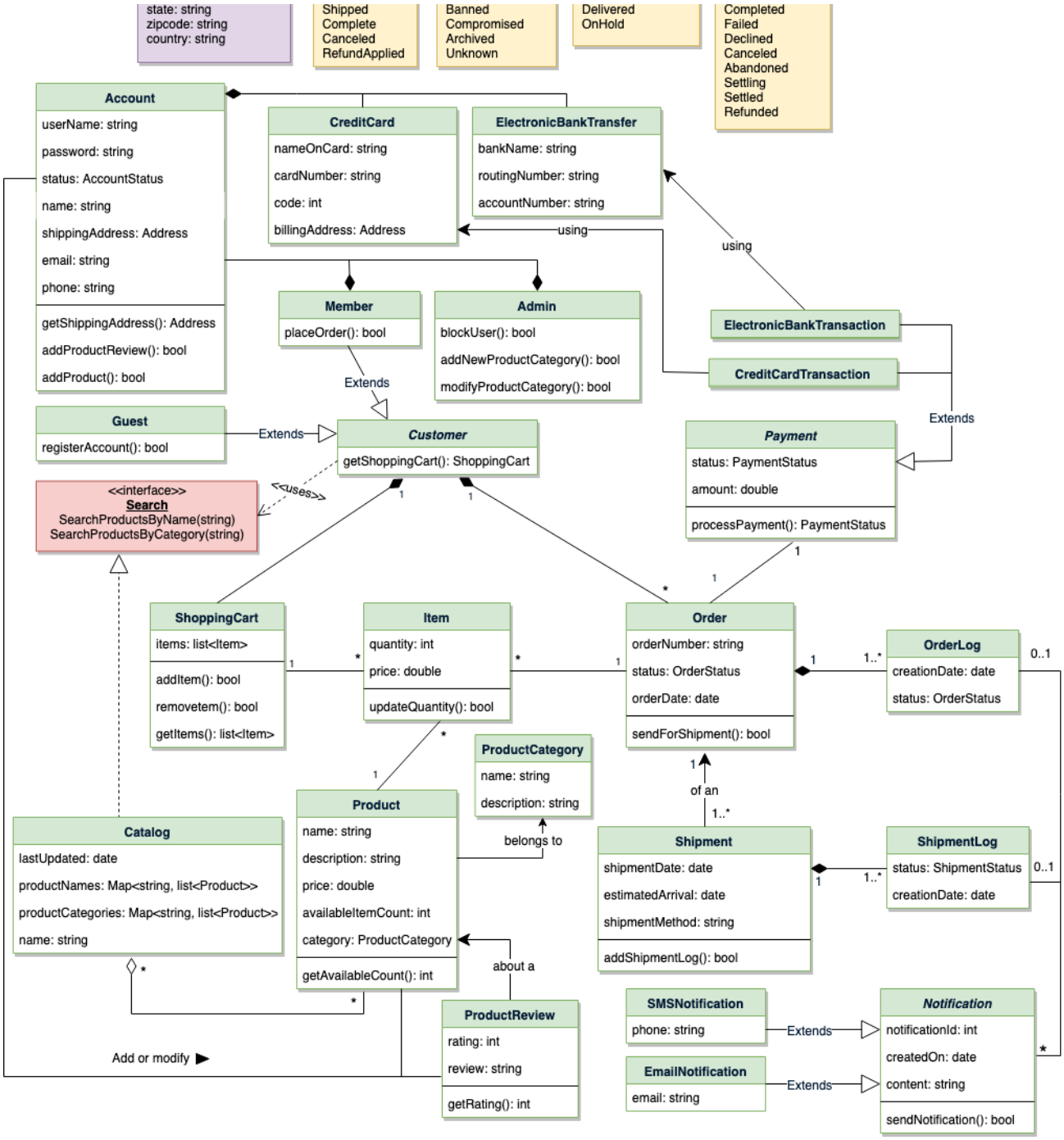
Class diagram

#

Here are the descriptions of the different classes of our Online Shopping System:

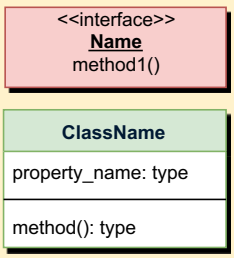
- **Account:** There are two types of registered accounts in the system: one will be an Admin, who is responsible for adding new product categories and blocking/unblocking members; the other, a Member, who can buy/sell products.
- **Guest:** Guests can search for and view products, and add them in the shopping cart. To place an order they have to become a registered member.
- **Catalog:** Users of our system can search for products by their name or category. This class will keep an index of all products for faster search.
- **ProductCategory:** This will encapsulate the different categories of products, such as books, electronics, etc.
- **Product:** This class will encapsulate the entity that the users of our system will be buying and selling. Each Product will belong to a ProductCategory.
- **ProductReview:** Any registered member can add a review about a product.
- **ShoppingCart:** Users will add product items that they intend to buy to the shopping cart.
- **Item:** This class will encapsulate a product item that the users will be buying or placing in the shopping cart. For example, a pen could be a product and if there are 10 pens in the inventory, each of these 10 pens will be considered a product item.
- **Order:** This will encapsulate a buying order to buy everything in the shopping cart.
- **OrderLog:** Will keep a track of the status of orders, such as unshipped, pending, complete, canceled, etc.
- **ShipmentLog:** Will keep a track of the status of shipments, such as pending, shipped, delivered, etc.
- **Notification:** This class will take care of sending notifications to customers.
- **Payment:** This class will encapsulate the payment for an order. Members can pay through credit card or electronic bank transfer.





Class diagram for Online Shopping System

UML conventions



Interface: Classes implement interfaces, denoted by Generalization.

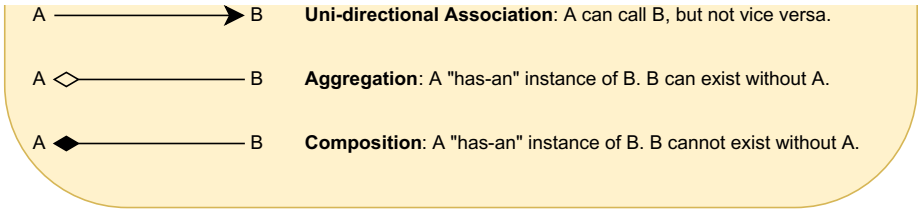
Class: Every class can have properties and methods.
Abstract classes are identified by their *Italic* names.

A - - - - - - - - - - - - - - - ▷ B **Generalization**: A implements B.

A ——————————▷ B **Inheritance**: A inherits from B. A "is-a" B.

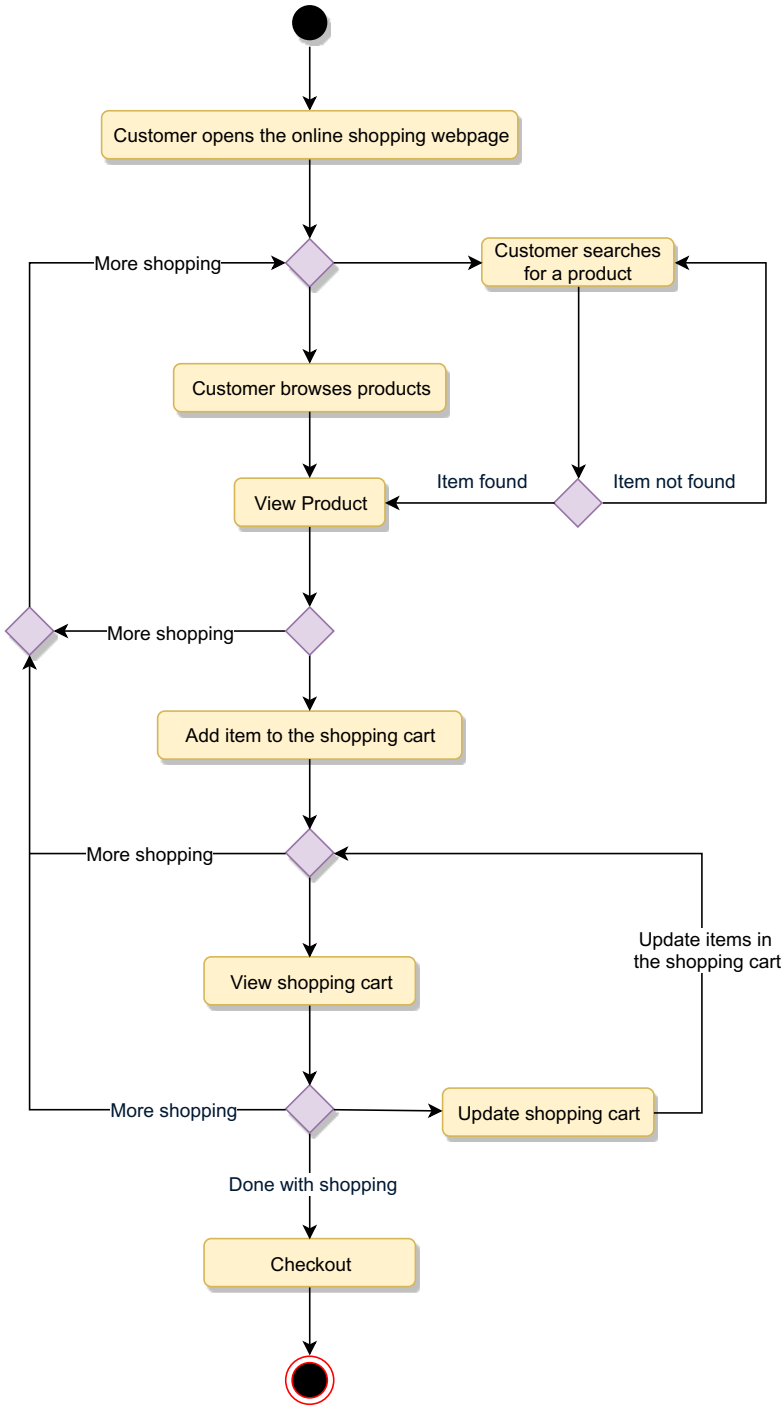
A - - - - - - - - - - - - - - - - - - - B **Use Interface:** A uses interface B.

A ——————————— B **Association**: A and B call each other.



Activity Diagram

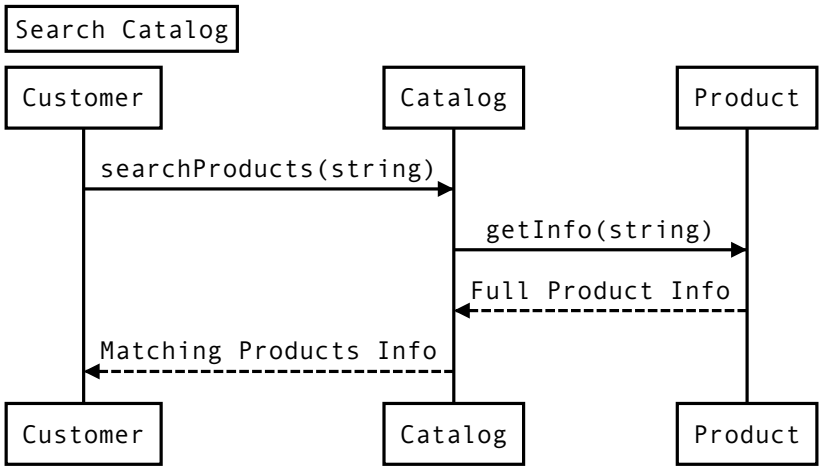
Following is the activity diagram for a user performing online shopping:



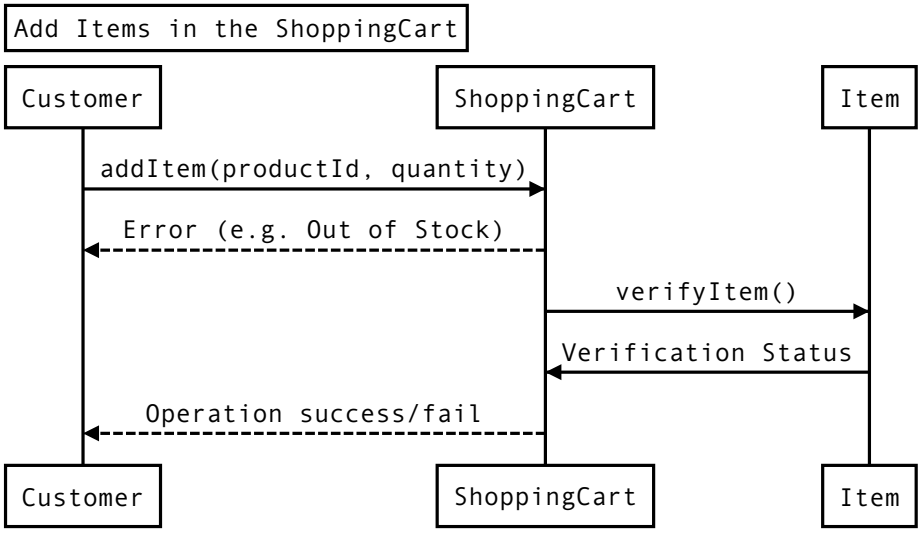
Sequence Diagram

#

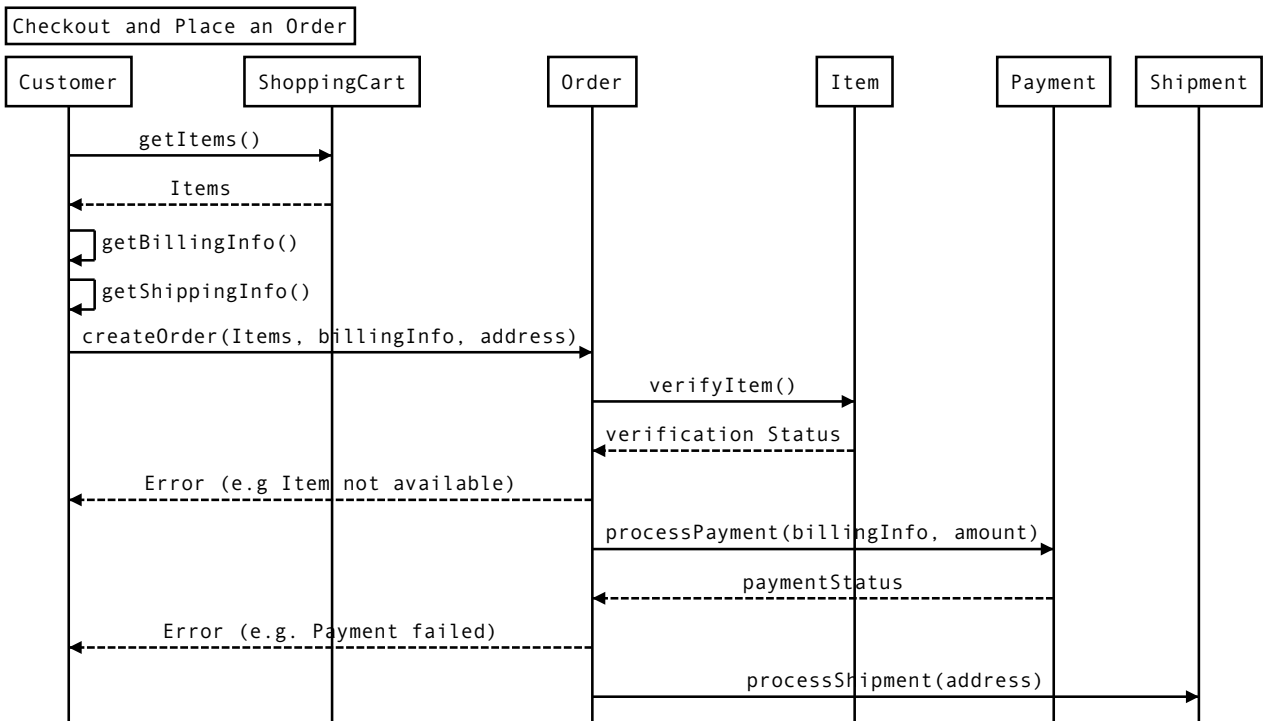
1. Here is the sequence diagram for searching from the catalog:

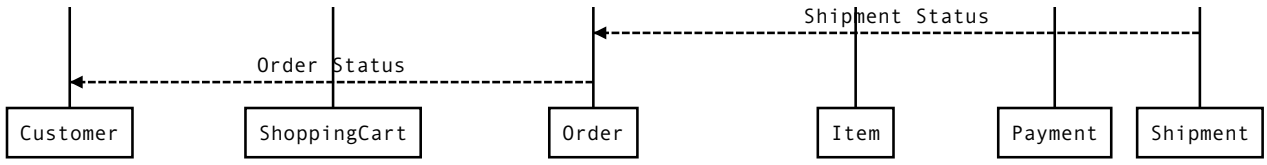


2. Here is the sequence diagram for adding an item to the shopping cart:



3. Here is the sequence diagram for checking out to place an order:





Code

#

Here is the high-level definition for the classes described above.

Enums, data types, and constants: Here are the required enums, data types, and constants:

Java

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

public enum OrderStatus {
  UNSHIPPED, PENDING, SHIPPED, COMPLETED, CANCELED, REFUND_APPLIED
}

public enum AccountStatus {
  ACTIVE, BLOCKED, BANNED, COMPROMISED, ARCHIVED, UNKNOWN
}

public enum ShipmentStatus {
  PENDING, SHIPPED, DELIVERED, ON_HOLD,
}

public enum PaymentStatus {
  UNPAID, PENDING, COMPLETED, FILLED, DECLINED, CANCELLED, ABANDONED, SETTLING, SETTLED, REFUNDED
}
```

Account, Customer, Admin, and Guest: These classes represent different people that interact with our system:

Java

```
// 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 methods function.

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

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

  public boolean addProduct(Product product);
  public boolean addProductReview(ProductReview review);
```

```
  public boolean resetPassword();
}

public abstract class Customer {
  private ShoppingCart cart;
  private Order order;

  public ShoppingCart getShoppingCart();
  public bool addItemToCart(Item item);
  public bool removeItemFromCart(Item item);
}

public class Guest extends Customer {
  public bool registerAccount();
}

public class Member extends Customer {
  private Account account;
  public OrderStatus placeOrder(Order order);
}
```

ProductCategory, Product, and ProductReview: Here are the classes related to a product:

Java

```
public class ProductCategory {
  private String name;
  private String description;
}

public class ProductReview {
  private int rating;
  private String review;

  private Member reviewer;
}

public class Product {
  private String productID;
  private String name;
  private String description;
  private double price;
  private ProductCategory category;
  private int availableItemCount;

  private Account seller;

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

ShoppingCart, Item, Order, and OrderLog: Users will add items to the shopping cart and place an order to buy all the items in the cart.

Java

```
public class Item {
  private String productID;
  private int quantity;
  private double price;

  public boolean updateQuantity(int quantity);
}

public class ShoppingCart {
```

```
public class ShoppingCart {
  private List<Items> items;

  public boolean addItem(Item item);
  public boolean removeItem(Item item);
  public boolean updateItemQuantity(Item item, int quantity);
  public List<Item> getItems();
  public boolean checkout();
}

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

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

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

Shipment, ShipmentLog, and Notification: After successfully placing an order, a shipment record will be created:

Java

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

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 abstract class Notification {
  private int notificationId;
  private Date createdOn;
  private String content;

  public boolean sendNotification(Account account);
}
```

Search interface and Catalog: Catalog will implement Search to facilitate searching of products.

Java

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

```
public class Catalog implements Search {
   HashMap<String, List<Product>> productNames;
   HashMap<String, List<Product>> productCategories;

  public List<Product> searchProductsByName(String name) {
    return productNames.get(name);
  }

  public List<Product> searchProductsByCategory(String category) {
    return productCategories.get(category);
  }
}
```

Mark as Completed

← Back

Next →
Design Stack Over…