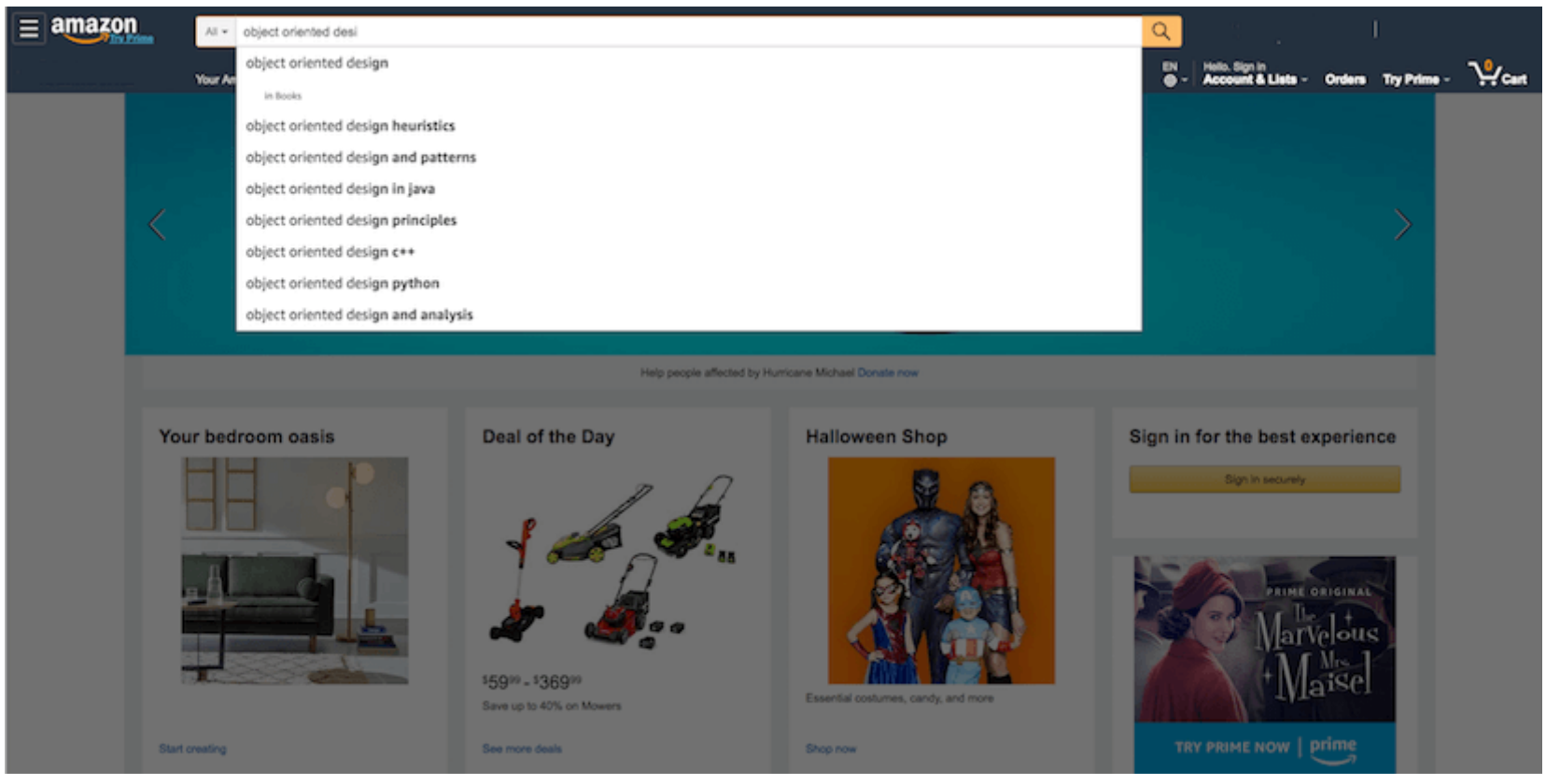
Design Amazon - Online Shopping System

Let's design an online retail store.

We'll cover the following

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.

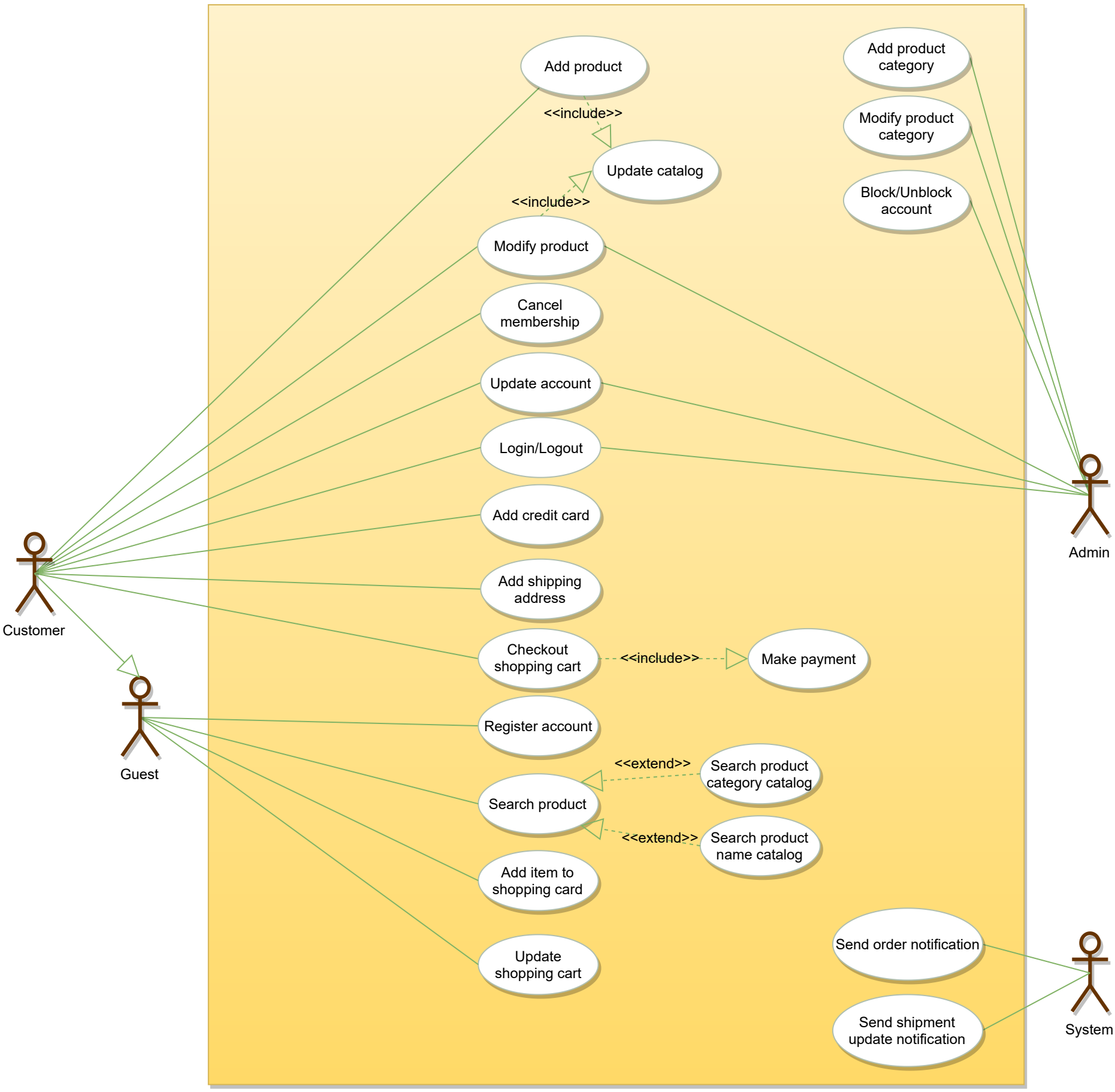
Use case Diagram

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.

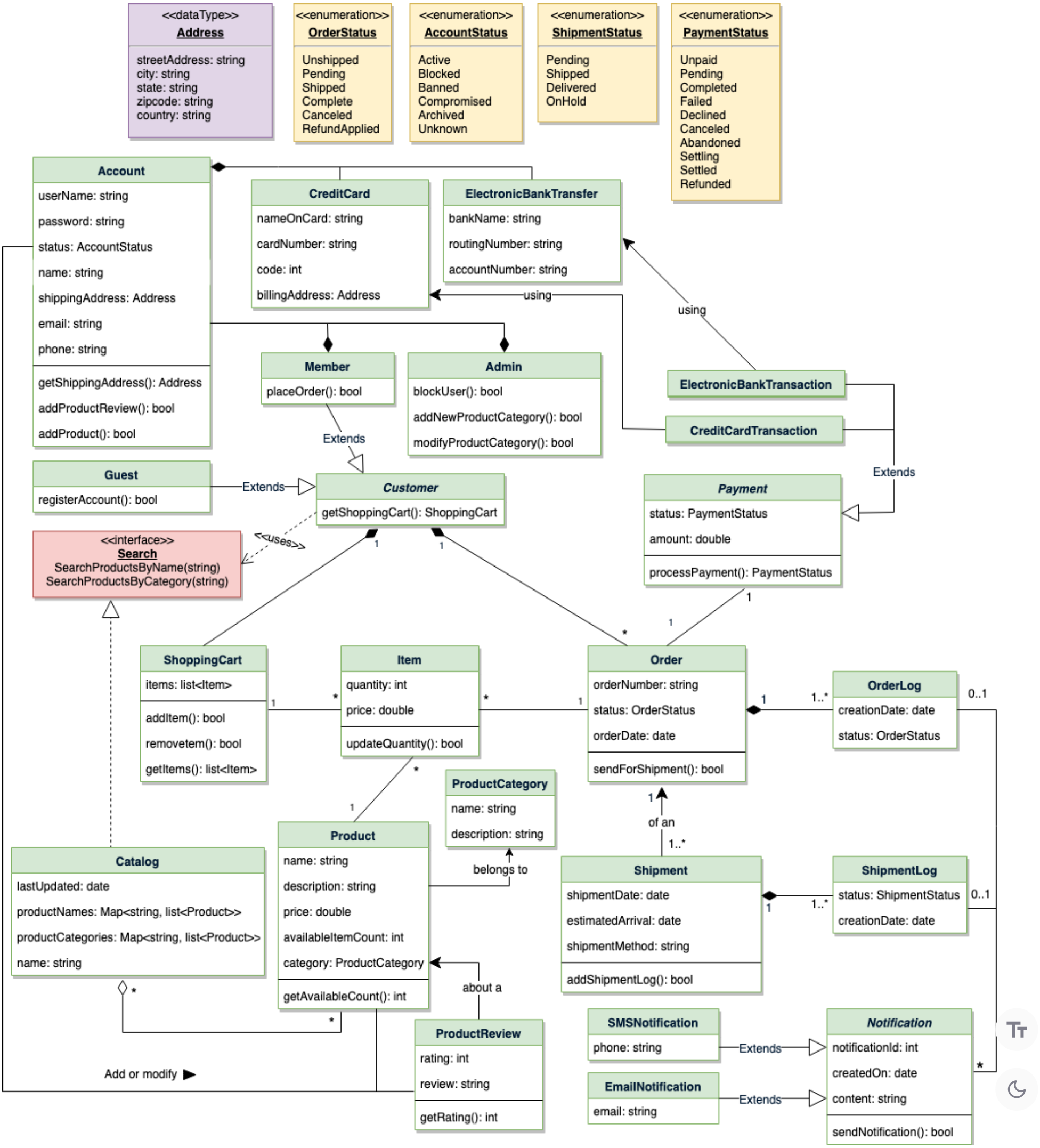


Class diagram

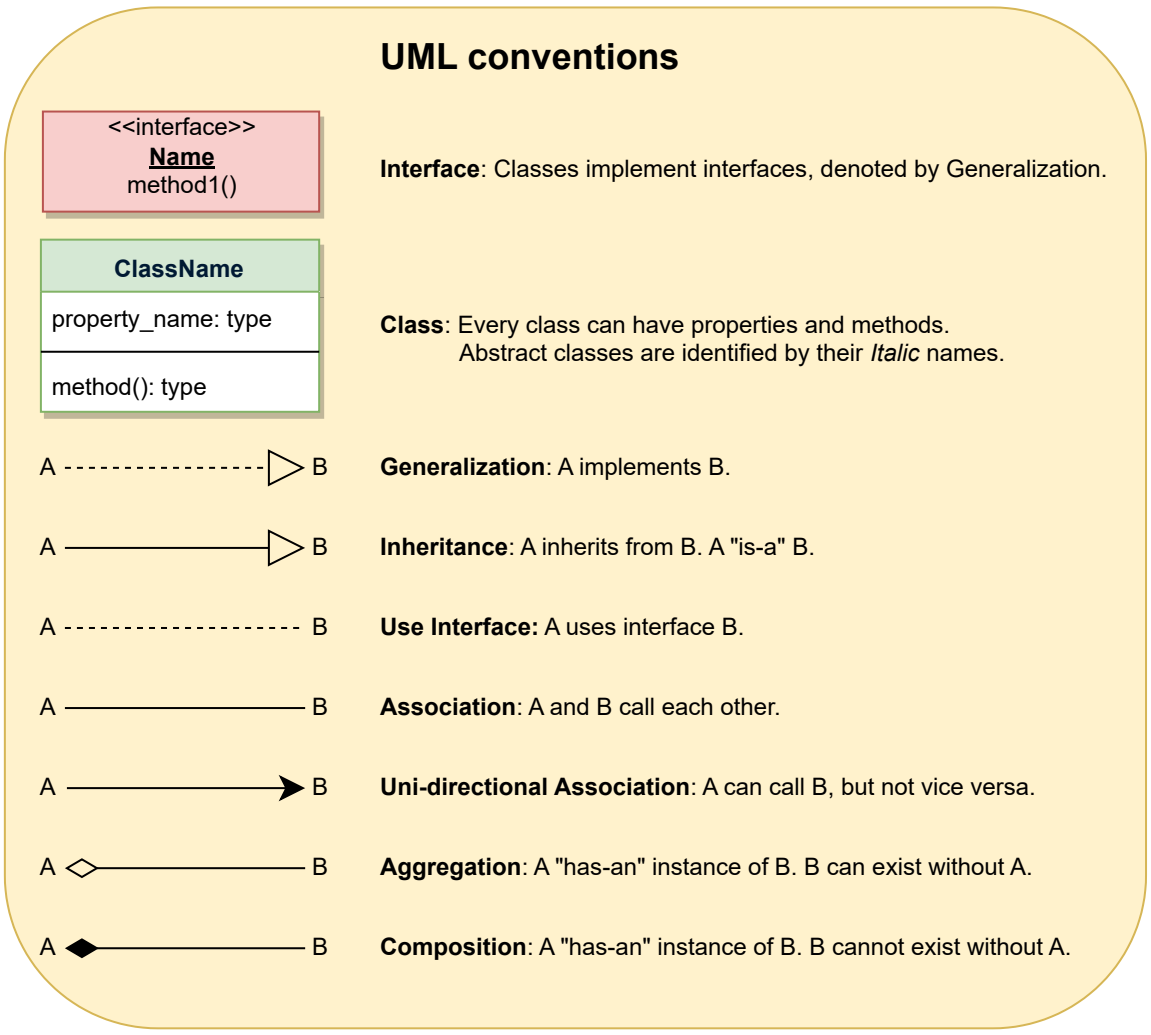
Here are the descriptions of the different classes of our Online Shopping System:

- **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.

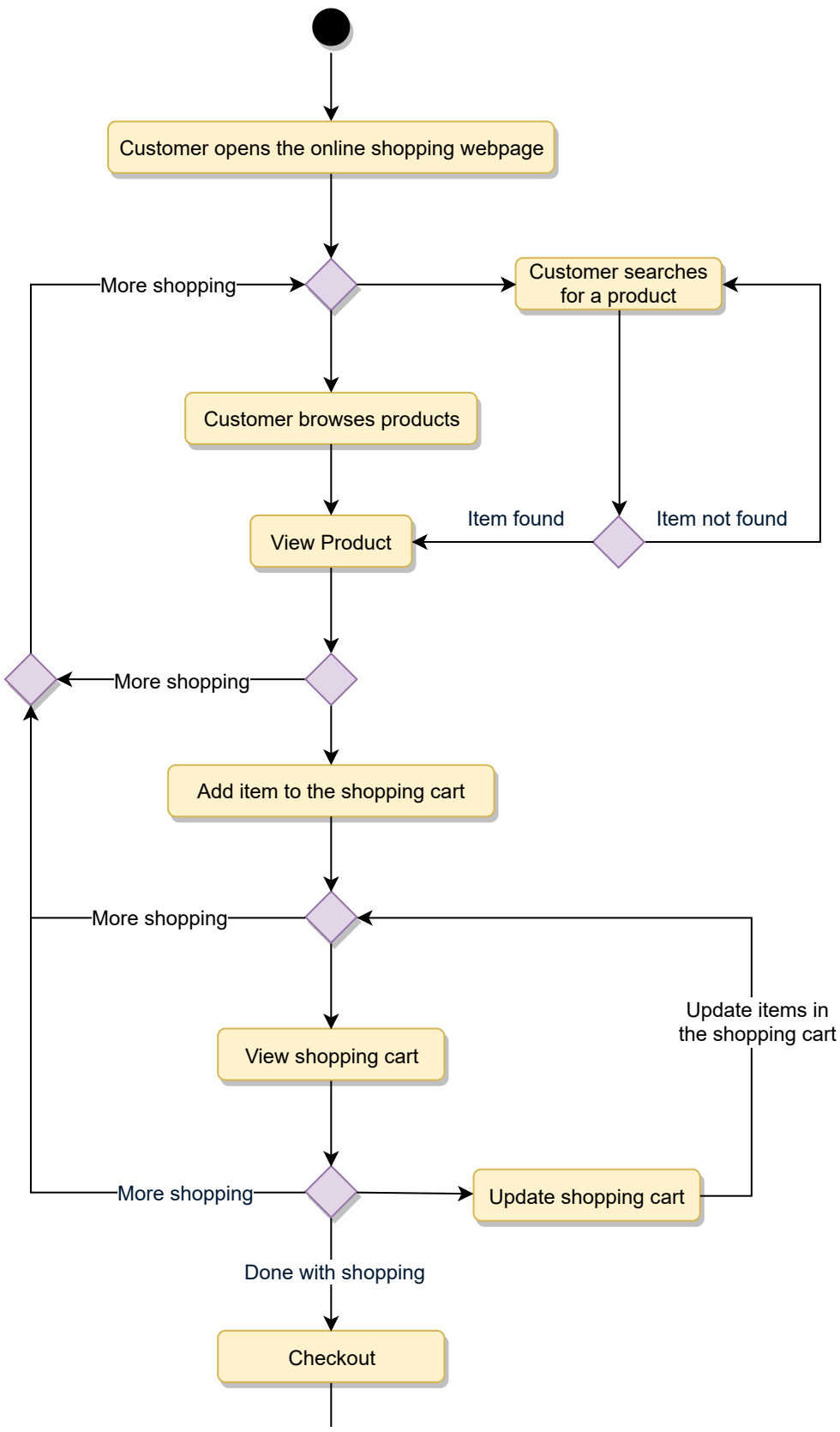


Class diagram for Online Shopping System



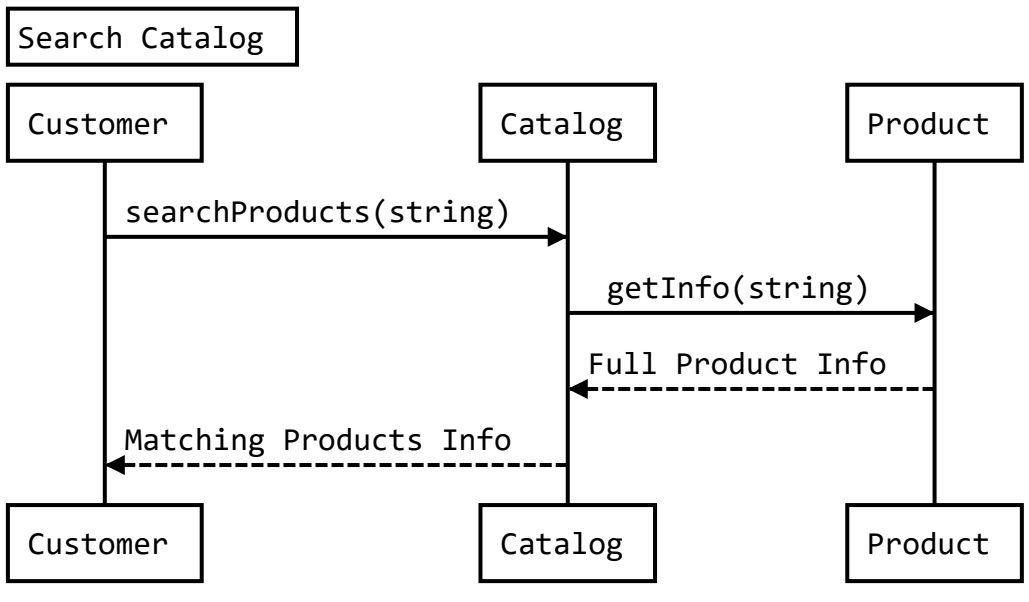
Activity Diagram

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

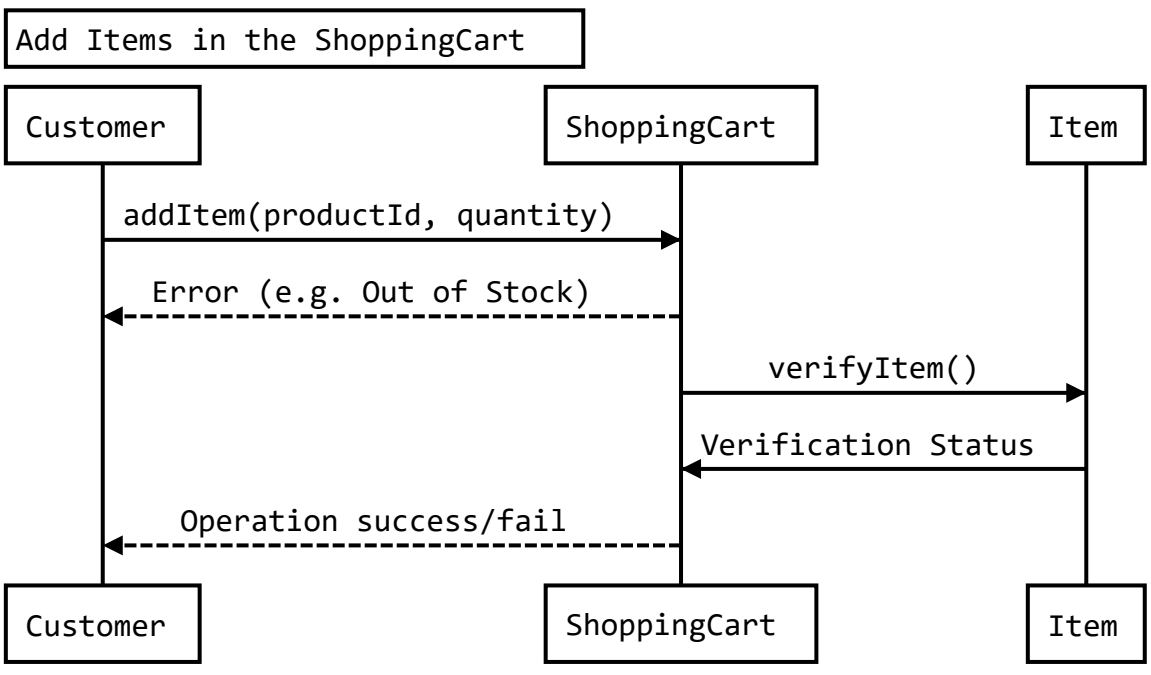


Sequence Diagram

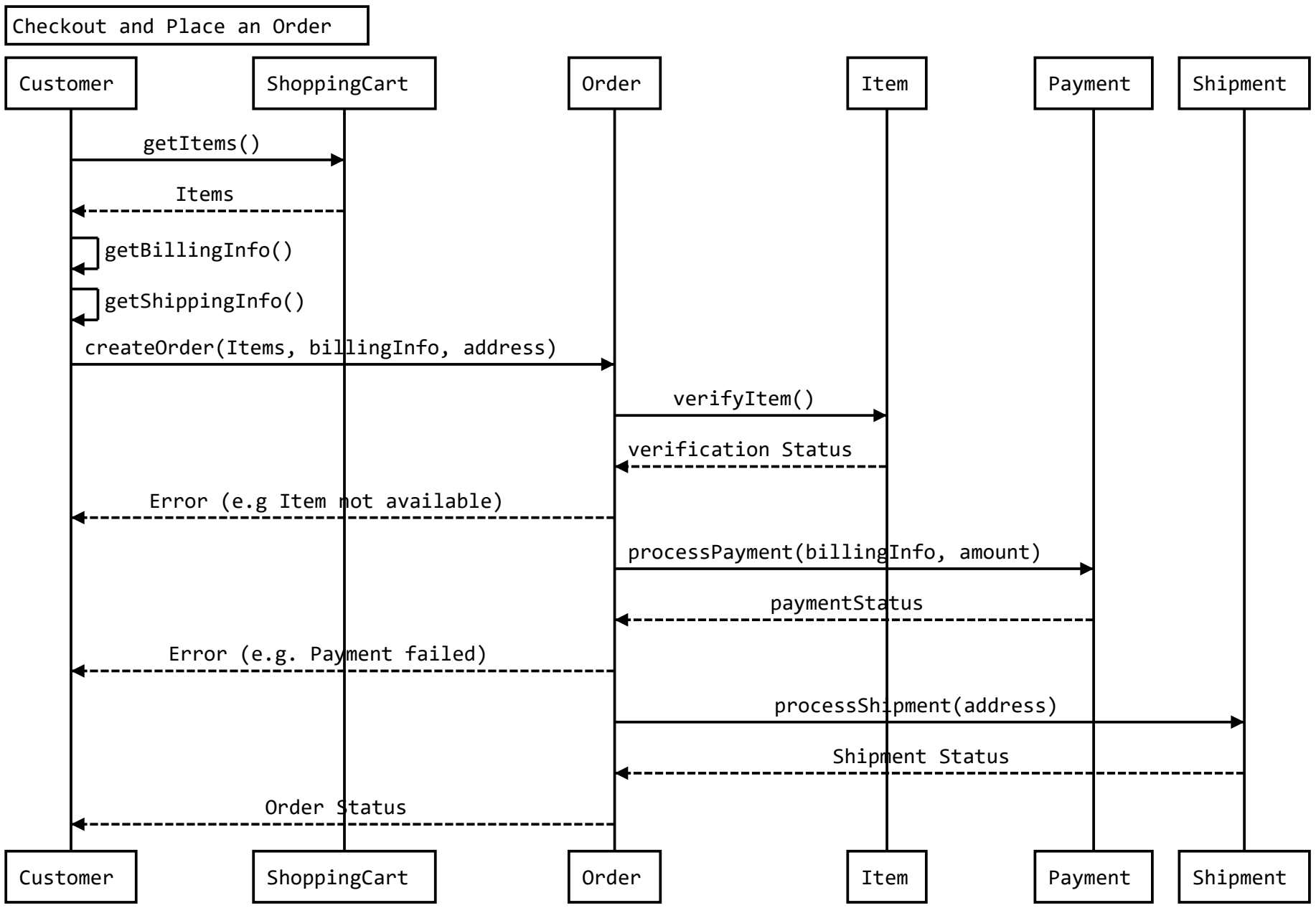
1. Here is the sequence diagram for searching from the catalog:



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 | Python

```
class Address:
  def __init__(self, street, city, state, zip_code, country):
    self.__street_address = street
    self.__city = city
    self.__state = state
    self.__zip_code = zip_code
    self.__country = country


class OrderStatus(Enum):
  UNSHIPPED, PENDING, SHIPPED, COMPLETED, CANCELED, REFUND_APPLIED = 1, 2, 3, 4, 5, 6


class AccountStatus(Enum):
  ACTIVE, BLOCKED, BANNED, COMPROMISED, ARCHIVED, UNKNOWN = 1, 2, 3, 4, 5, 6


class ShipmentStatus(Enum):
  PENDING, SHIPPED, DELIVERED, ON_HOLD = 1, 2, 3, 4


class PaymentStatus(Enum):
  UNPAID, PENDING, COMPLETED, FILLED, DECLINED, CANCELLED, ABANDONED, SETTLING, SETTLED, REFUNDED = 1, 2, 3, 4,
```

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

Java Python

```
# 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.


class Account:
  def __init__(self, user_name, password, name, email, phone, shipping_address, status=AccountStatus):
    self.__user_name = user_name
    self.__password = password
    self.__name = name
    self.__email = email
    self.__phone = phone
    self.__shipping_address = shipping_address
    self.__status = status.ACTIVE
    self.__credit_cards = []
    self.__bank_accounts = []

  def add_product(self, product):
    None

  def add_productReview(self, review):
    None

  def reset_password(self):
    None
```

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

Java Python

```
class ProductCategory:
  def __init__(self, name, description):
    self.__name = name
    self.__description = description


class ProductReview:
  def __init__(self, rating, review, reviewer):
    self.__rating = rating
    self.__review = review
    self.__reviewer = reviewer


class Product:
  def __init__(self, id, name, description, price, category, seller_account):
    self.__product_id = id
    self.__name = name
    self.__description = description
    self.__price = price
    self.__category = category
    self.__available_item_count = 0

    self.__seller = seller_account

  def get_available_count(self):
    return self.__available_item_count
```

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 Python

```
class Item:
```

```
    def __init__(self, id, quantity, price):
      self.__product_id = id
      self.__quantity = quantity
      self.__price = price

    def update_quantity(self, quantity):
      None


  class ShoppingCart:
    def __init__(self):
      self.__items = []

    def add_item(self, item):
      None

    def remove_item(self, item):
      None

    def update_item_quantity(self, item, quantity):
      None

    def get_items(self):
      return self.__items
```

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

Java | Python

```
  class ShipmentLog:
    def __init__(self, shipment_number, status=ShipmentStatus.PENDING):
      self.__shipment_number = shipment_number
      self.__status = status
      self.__creation_date = datetime.date.today()


  class Shipment:
    def __init__(self, shipment_numbe, shipment_methodr):
      self.__shipment_number = shipment_number
      self.__shipment_date = datetime.date.today()
      self.__estimated_arrival = datetime.date.today()
      self.__shipment_method = shipment_method
      self.__shipmentLogs = []

    def add_shipment_log(self, shipment_log):
      None


  # from abc import ABC, abstractmethod
  class Notification(ABC):
    def __init__(self, id, content):
      self.__notification_id = id
      self.__created_on = datetime.date.today()
      self.__content = content

    def send_notification(self, account):
```

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

Java | Python

```
  from abc import ABC, abstractmethod

  class Search(ABC):
    def search_products_by_name(self, name):
      None
```

```
    def search_products_by_category(self, category):
      None


  class Catalog(Search):
    def __init__(self):
      self.__product_names = {}
      self.__product_categories = {}
```