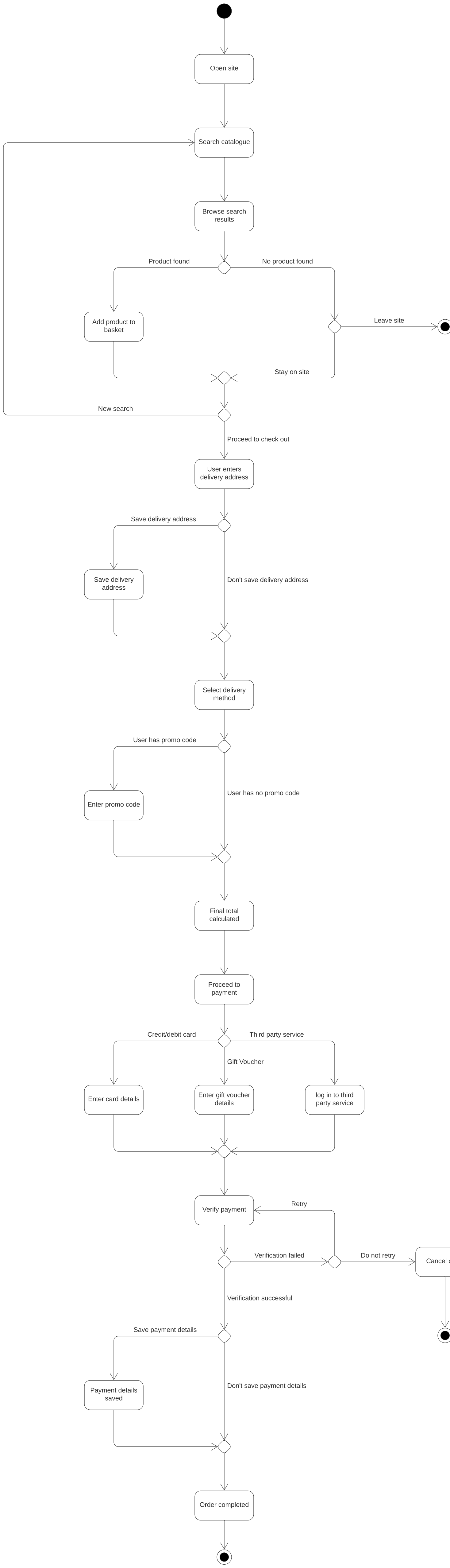
Object-Oriented Information Systems Mid-Module Assignment: System Design

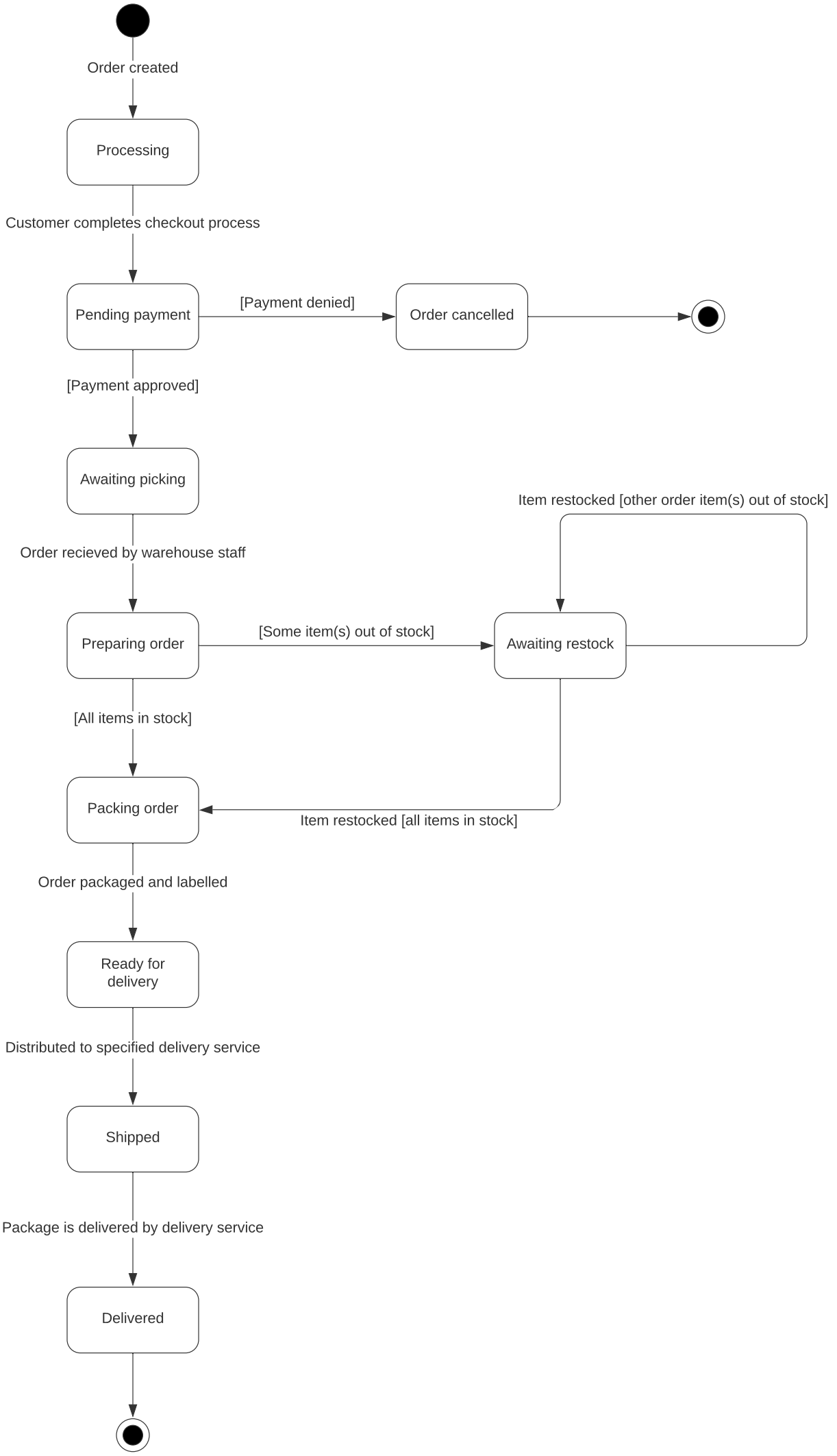
Activity Diagram

The activity diagram below shows the proposed process which a customer would go through to complete an order on the site. Upon completion by the customer, the order would then be passed to the appropriate warehouse staff, as specified in the question.


Open site
Search catalogue
Browse search results
Product found
No product found
Add product to basket
Leave site
Stay on site
New search
Proceed to check out
User enters delivery address
Save delivery address
Save delivery address
Don't save delivery address
Select delivery method
User has promo code
Enter promo code
User has no promo code
Final total calculated
Proceed to payment
Credit/debit card
Third party service
Gift Voucher
Enter card details
Enter gift voucher details
log in to third party service
Verify payment
Retry
Verification failed
Do not retry
Cancel order
Verification successful
Save payment details
Payment details saved
Don't save payment details
Order completed

State Diagram

The state diagram below shows the various states of an order as it is passed through the system, from the point when it is created by the customer at checkout.


Order created
Processing
Customer completes checkout process
Pending payment
[Payment denied]
Order cancelled
[Payment approved]
Awaiting picking
Order recieved by warehouse staff
Item restocked [other order item(s) out of stock]
Preparing order
[Some item(s) out of stock]
Awaiting restock
[All items in stock]
Packing order
Item restocked [all items in stock]
Order packaged and labelled
Ready for delivery
Distributed to specified delivery service
Shipped
Package is delivered by delivery service
Delivered

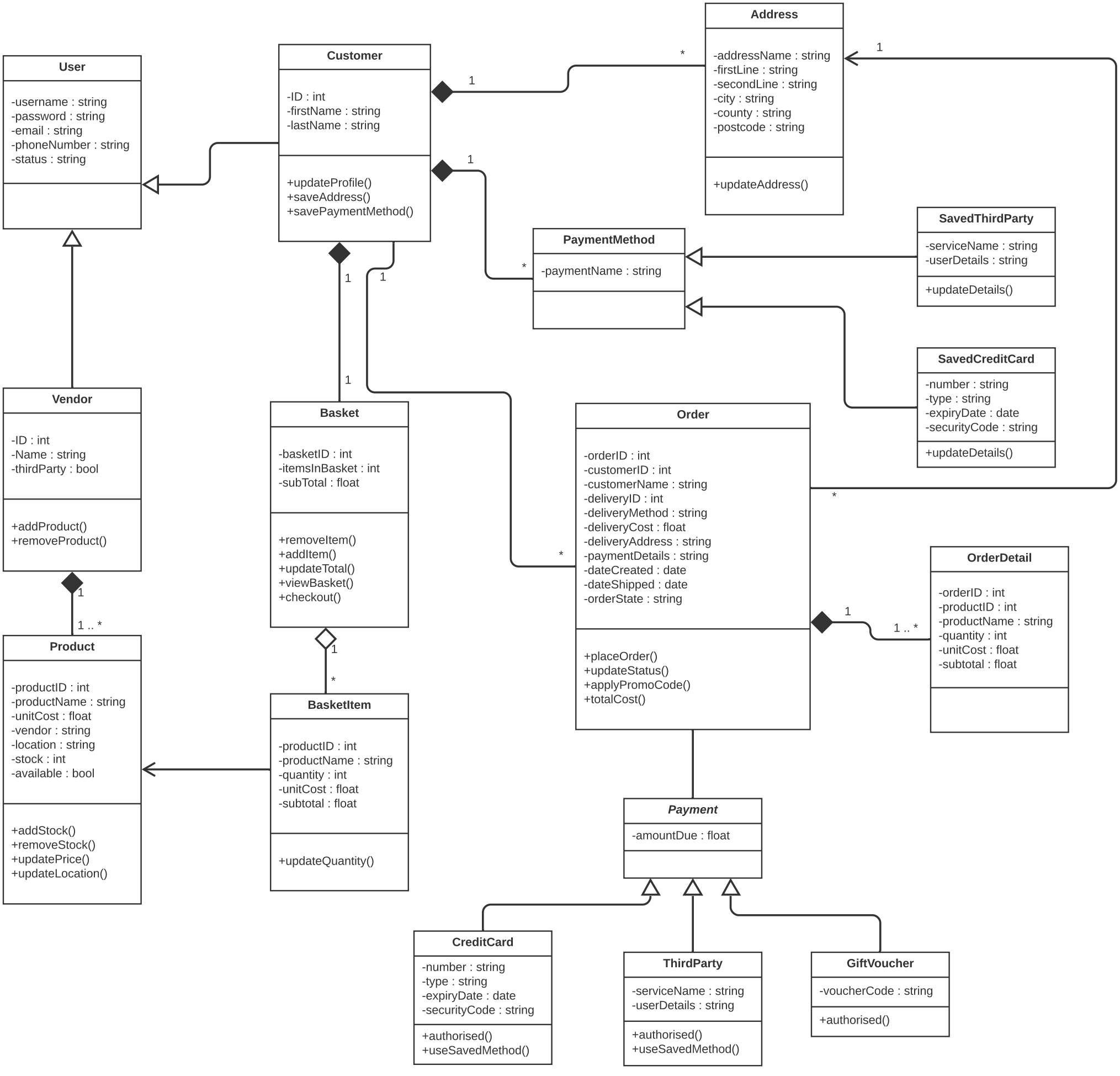
Class Diagram

The class diagram below lays out the class structure for the online store. With this structure, each Vendor would have a unique ID, and an attribute denoting them as first or third party vendors. The methods on the Vendor class would allow each vendor to generate new instances of the Product class, as well as remove them. Each instance of the Product class is part of the vendor's catalogue, and therefore if the vendor were to remove themselves from the store, all Product instances associated with them would be removed, hence the use of the composition relationship between the two classes.

Each Customer object can store multiple Address and PaymentMethod objects. This allows for ease of use for the customer. This is a composition relationship, as if the Customer object is removed from the system, so too must its associated addresses and payment methods. The PaymentMethod class is the parent of the SavedCreditCard and SavedThirdParty classes, both of which store payment information in their attributes, and have an update method.

Each customer in this system would have a unique instance of the Basket class associated with their account. A basket can be empty, or contain any number of BAsketItem objects. Each BasketItem object is associated with a Product object, and contains the product's name, ID, and unit cost. The quantity attribute on a BasketItem can be updated by the updateQuantity() method with will then also update the subtotal for the object. Once the customer has completed their shopping, each BasketItem object will be converted to an OrderDetail object as part of the checkout() method.

The Order class contains many attributes for things such as a unique order ID, the date the order was made, delivery method, and the order State. The methods on this class allow for the order to be placed, the order status to be updated, promo codes to be applied, and the total cost to be calculated. Any Order object is also composed of 1 or more OrderDetail objects. The Order class is associated with the Customer class in a many-to-one relationship, as any one customer can make multiple orders, but each order can only be made by a single customer. This also applies to the relationship between an order and an address.


User
-username : string
-password : string
-email : string
-phoneNumber : string
-status : string
Customer
-ID : int
-firstName : string
-lastName : string
+updateProfile()
+saveAddress()
+savePaymentMethod()
Address
-addressName : string
-firstLine : string
-secondLine : string
-city : string
-county : string
-postcode : string
+updateAddress()
PaymentMethod
-paymentName : string
SavedThirdParty
-serviceName : string
-userDetails : string
+updateDetails()
SavedCreditCard
-number : string
-type : string
-expiryDate : date
-securityCode : string
+updateDetails()
Vendor
-ID : int
-Name : string
-thirdParty : bool
+addProduct()
+removeProduct()
Basket
-basketID : int
-itemsInBasket : int
-subTotal : float
+removeItem()
+addItem()
+updateTotal()
+viewBasket()
+checkout()
Order
-orderID : int
-customerID : int
-customerName : string
-deliveryID : int
-deliveryMethod : string
-deliveryCost : float
-deliveryAddress : string
-paymentDetails : string
-dateCreated : date
-dateShipped : date
-orderState : string
+placeOrder()
+updateStatus()
+applyPromoCode()
+totalCost()
OrderDetail
-orderID : int
-productID : int
-productName : string
-quantity : int
-unitCost : float
-subtotal : float
Product
-productID : int
-productName : string
-unitCost : float
-vendor : string
-location : string
-stock : int
-available : bool
+addStock()
+removeStock()
+updatePrice()
+updateLocation()
BasketItem
-productID : int
-productName : string
-quantity : int
-unitCost : float
-subtotal : float
+updateQuantity()
Payment
-amountDue : float
CreditCard
-number : string
-type : string
-expiryDate : date
-securityCode : string
+authorised()
+useSavedMethod()
ThirdParty
-serviceName : string
-userDetails : string
+authorised()
+useSavedMethod()
GiftVoucher
-voucherCode : string
+authorised()
1
*
1 .. *