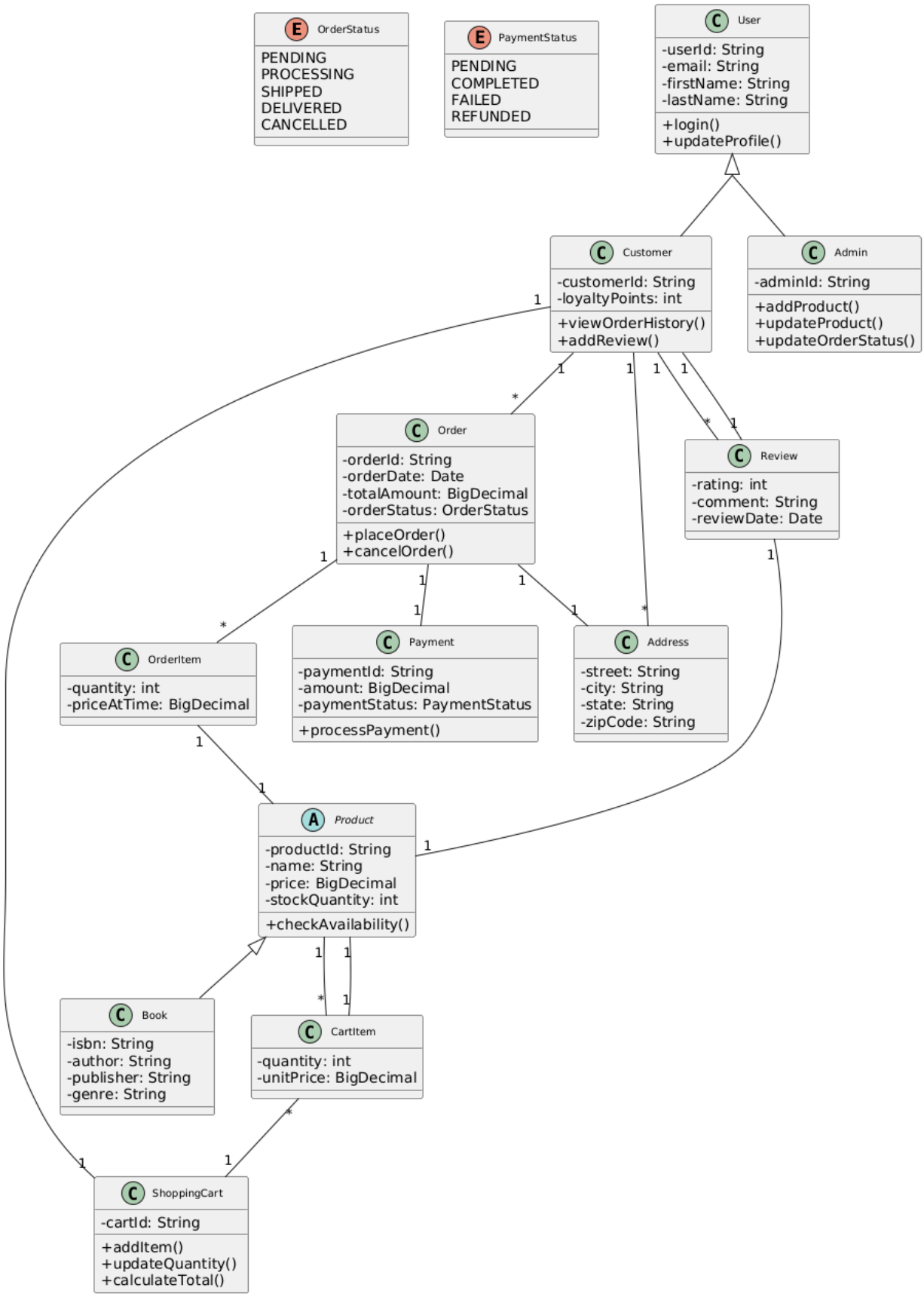
Course: IT643 - Software Design and Testing
Project Name: Chapter 4

Team members

ID	Names
202412088	Hirwa Saraiya
202412049	Monika Gautam
202412065	Dhara Patel
202412047	Manya Singh


OrderStatus
PENDING
PROCESSING
SHIPPED
DELIVERED
CANCELLED
PaymentStatus
PENDING
COMPLETED
FAILED
REFUNDED
User
-userId: String
-email: String
-firstName: String
-lastName: String
+login()
+updateProfile()
Customer
-customerId: String
-loyaltyPoints: int
+viewOrderHistory()
+addReview()
Admin
-adminId: String
+addProduct()
+updateProduct()
+updateOrderStatus()
Order
-orderId: String
-orderDate: Date
-totalAmount: BigDecimal
-orderStatus: OrderStatus
+placeOrder()
+cancelOrder()
Review
-rating: int
-comment: String
-reviewDate: Date
OrderItem
-quantity: int
-priceAtTime: BigDecimal
Payment
-paymentId: String
-amount: BigDecimal
-paymentStatus: PaymentStatus
+processPayment()
Address
-street: String
-city: String
-state: String
-zipCode: String
Product
-productId: String
-name: String
-price: BigDecimal
-stockQuantity: int
+checkAvailability()
Book
-isbn: String
-author: String
-publisher: String
-genre: String
CartItem
-quantity: int
-unitPrice: BigDecimal
ShoppingCart
-cartId: String
+addItem()
+updateQuantity()
+calculateTotal()

1. Entities (Classes) and Their Attributes & Behaviours

Entities are the main objects in the system, each with defining attributes (data) and behaviours (methods).

Entity	**Key Attributes**	**Key Behaviours (Methods)**	**Description**
User	userId, email, firstName, lastName	login(), updateProfile()	An abstract base class representing anyone who uses the system. It holds common user information.
Customer	customerId, loyaltyPoints	viewOrderHistory(), addReview()	A subclass of User. Represents a registered customer who can shop and interact with products.
Admin	adminId	addProduct(), updateProduct(), updateOrderStatus()	A subclass of User. Represents a system administrator with privileges to manage the catalog and orders.
Product	productId, name, price, stockQuantity	checkAvailability()	An abstract class representing any item for sale. Defines common properties for all products.

Book	isbn, author, publisher, genre	-	A concrete subclass of Product. It adds specific attributes that are unique to books.
ShoppingCart	cartId	addItem(), updateQuantity(), calculateTotal()	A temporary container for items a Customer intends to purchase before checkout.
CartItem	quantity, unitPrice	-	A linking class that represents a specific Product and its quantity in a ShoppingCart.
Order	orderId, orderDate, totalAmount, orderStatus	placeOrder(), cancelOrder()	Created at checkout. Represents a confirmed transaction and its current state (e.g., SHIPPED).
OrderItem	quantity, priceAtTime	-	A snapshot of a Product and its price at the moment of purchase, linked to an Order.
Payment	paymentId, amount, paymentStatus	processPayment()	Represents the financial transaction associated with an Order.

Address	street, city, state, zipCode	-	Stores the physical shipping address for an Order. A Customer can have multiple addresses.
Review	rating, comment, reviewDate	-	-

2. Relationships and Their Cardinality/Multiplicity

Relationships describe how entities interact and are connected. Cardinality defines the numerical relationship between instances of these entities (e.g., one-to-one, one-to-many).

Relationship	**Description**	**Cardinality**
Inheritance (Generalization)	Represents an "is-a" relationship.	
User ◁— Customer	A Customer is a type of User.	1:1 (Per Customer instance is one User instance)
User ◁— Admin	An Admin is a type of User.	1:1 (Per Admin instance is one User instance)
Product ◁— Book	A Book is a type of Product.	1:1 (Per Book instance is one Product instance)
Associations (Has-A)	Represents a "has-a" or "uses" relationship.	
Customer ─ ShoppingCart	A Customer has one ShoppingCart.	1 to 1 (One customer has one active cart)
ShoppingCart ─ CartItem	A ShoppingCart contains multiple CartItems.	**1 to *** (One cart contains many items)
CartItem ─ Product	A CartItem references one specific Product.	*** to 1** (Many cart items can reference the same product)

Customer ─ Order	A Customer places multiple Orders.	**1 to *** (One customer can have many orders)
Order ─ OrderItem	An Order contains multiple OrderItems.	**1 to *** (One order contains many items)
OrderItem ─ Product	An OrderItem is an instance of one Product.	*** to 1** (Many order items can be instances of the same product)
Order ─ Payment	An Order is paid by one Payment.	1 to 1 (One order has one payment transaction)
Order ─ Address	An Order is shipped to one Address.	1 to 1 (One order is shipped to one address)
Customer ─ Address	A Customer has multiple Addresses.	**1 to *** (One customer can have many saved addresses)
Customer ─ Review	A Customer writes multiple Reviews.	**1 to *** (One customer can write many reviews)
Review ─ Product	A Review is for one Product.	*** to 1** (Many reviews can be written for one product)

3. Key System Behaviours (Use Cases Mapped to Methods)

The methods in the classes work together to facilitate core use cases:

1. Browsing the Catalog: The Product and Book classes hold the data displayed to the user.
2. Managing a Shopping Cart:
    - customer.addItem(...) triggers shoppingCart.addItem(...), which creates a new CartItem linked to a Product.
    - shoppingCart.calculateTotal() sums the subtotals of all CartItem objects (quantity * unitPrice).
3. Placing an Order:
    - customer.placeOrder() converts the ShoppingCart (with its CartItems) into an Order (with OrderItems).
    - The Order creates a Payment object and calls payment.processPayment().
    - Upon successful payment, the orderStatus is set to PROCESSING and product.stockQuantity is decreased.
4. Product Reviews: customer.addReview(...) creates a new Review object linked to both the Customer and the Product.