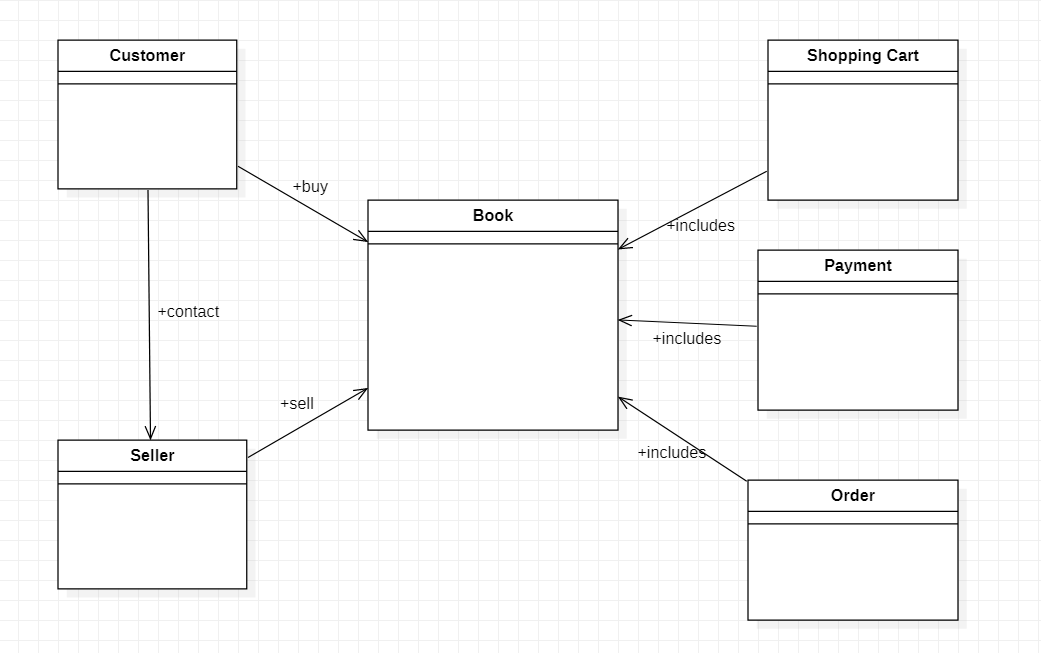
**Class Diagram**



**Class Responsibility**

|  |  |
| --- | --- |
| Customers | |
| Customer ID  Customer Name  Customer Phone Number  Delivery Address  Payment ID  Order ID  Order records | Sellers  Books  Shopping Cart  Payments  Orders |

|  |  |
| --- | --- |
| Sellers | |
| Seller ID  Seller Name  Seller Phone Number  Display Web Page  Display Sell Details  Check Inventory  Sell Books  Send Invoice | Customers  Books  Payments  Orders |

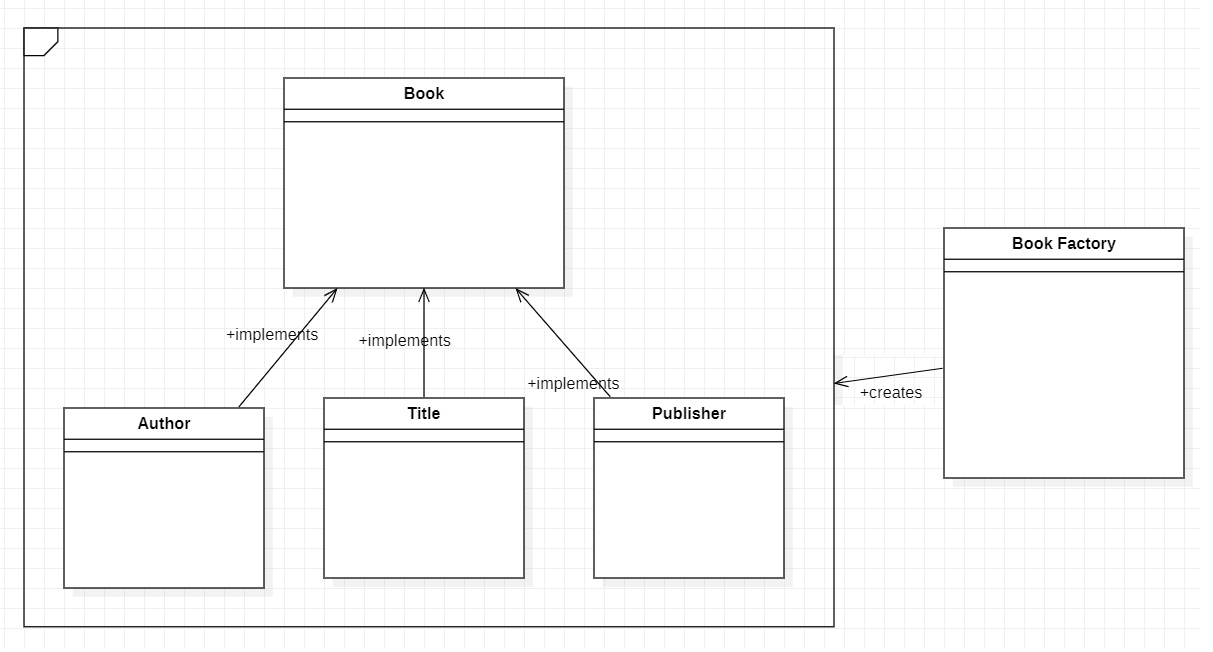
|  |  |
| --- | --- |
| Books | |
| Book ID  Book Name  Author  Book Price  Year of Publication  Book Category  Book Details  Search book | Customers  Sellers  Shopping Cart  Orders |

|  |  |
| --- | --- |
| Shopping Cart | |
| Display Customers ID  Display Order Totals  Add to Cart  Remove from Cart  Quantity of Items  Book Details  Seller Details | Customers  Books  Payments  Orders |

|  |  |
| --- | --- |
| Payment | |
| Payment ID  Display Payment of Book Store Name  Display Payment Details  Display Payment Amount  Select Payment Method  Issue Receipt  Keep Records of Customer Payment | Customers  Orders |

|  |  |
| --- | --- |
| Order | |
| Order ID  Process Order  Order Confirmation  Display Amount of Order  Display Quantity of Order  Display Delivery Date  Display Shipping Fee  Shipping Method | Customers  Sellers  Books  Orders  Payments |

**Factory Design Pattern**



The factory pattern was chosen because of its modular scalability. When I determine to add a new class to the factory, I can add the class directly without passing other tests or codes. Besides, factory pattern obtains good testability. For instance, if a creator achieves four classes, he can test the function independently of the calling class to ensure the factory returns the correct types on the correct conditions. It doesn't need to know the factory itself or the implementation of a specific class. Apart from that, when using factory pattern, I can reuse the class. If I want to instantiate in many places, I do not have to repeat my conditions, so when I add a new class, I do not risk losing one. To sum up, the benefit I mentioned above is the reason why I choose the factory pattern.