**Class Diagram**

Diagram

Description automatically generated

**Class Responsibility**

|  |  |
| --- | --- |
| **Books** | |
| Book ID  Book Name  Book Type  Author  Book Description | Customers  Sellers  Carts  Wishlists  Payments  Orders  Orderline |

|  |  |
| --- | --- |
| **Customers** | |
| Customers ID  Customers Name  Gender  Phone Number  Email Address  Delivery Address | Books  Sellers  Carts  Wishlists  Payments  Orders |

|  |  |
| --- | --- |
| **Sellers** | |
| Sellers ID  Sellers Name  Book ID  Shop ID  Shop Name  Email Address  Phone Number  Shop Rating | Customers  Books  Payments  Orders  Orderline |

|  |  |
| --- | --- |
| **Carts** | |
| Book ID  Book Name  Book Amount  Book Price  Shop ID | Customers  Books |

|  |  |
| --- | --- |
| **Wishlists** | |
| Book ID  Book Name  Book Amount  Book Price  Shop ID | Customers  Books |

|  |  |
| --- | --- |
| **Payments** | |
| Payment ID  Customer ID  Seller ID  Order ID  Email Address  Phone Number  Total Amounts  Payments Method | Customers  Sellers  Books  Orders |

|  |  |
| --- | --- |
| **Order** | |
| Order ID  Customer ID  Seller ID  Book ID  Book Amount  Book Price  Total Order Amounts  Email Address  Phone Number  Recipient Name  Delivery Address  Delivery Status | Customers  Books  Sellers  Payments  Orderline |

|  |  |
| --- | --- |
| **Orderlines** | |
| Order ID  Book Amount  Book Price  Delivery Status  Back Order | Orders  Books |

**Design Pattern: Factory Pattern**

Diagram

Description automatically generated

Factory pattern was chosen to implement the Book Trading System class diagram. According to *Refactoring Guru* (2021), factory pattern is a type of creational design pattern that provides an interface or abstract class for creating objects in a parent class and allows subclasses to give the decision for the classes to instantiate. The reason of choosing factory pattern is to provide choices to create object types for subclasses. Besides that, it can be used when a parent class does not know what subclasses is needed to be create. Moreover, the class of objects that being returned by the factory pattern can be altered by subclasses without changing to the entire codebase. For instance, the payments method that only allowed for the customers to make payments was Touch ‘n Go eWallet. Subsequently, after using the factory pattern, we can add more payments method such as debit cards, credit cards, online banking and Boost without altering the whole codebase. This will significantly decrease the workload due to only small part of codes are needed to modify for each additional changes needed. Hence, factory pattern has been chosen as a design pattern for developing the Book Trading System.

**Reference:**

*Java T Point* (2021) Factory Method Pattern. Available from <https://www.javatpoint.com/factory-method-design-pattern> [accessed I July 2021].

*Refactoring Guru* (2021) Factory Method. Available from <https://refactoring.guru/design-patterns/factory-method> [accessed 1 July 2021].

*Source Making* (2021) Factory Method Design Pattern. Available from <https://sourcemaking.com/design_patterns/factory_method> [accessed 1 July 2021].