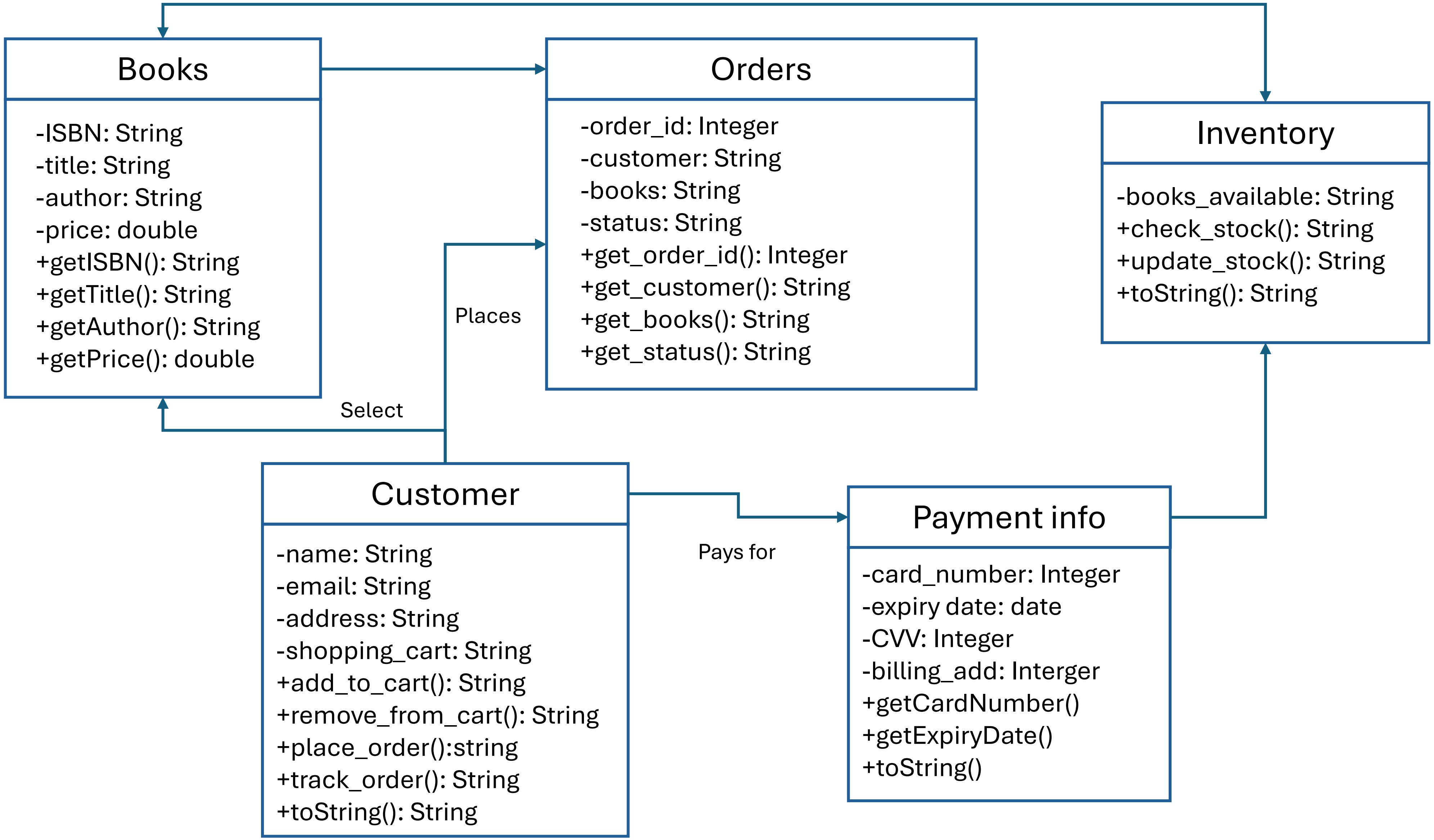
Class Diagram for Data Optimization in Books Library



Books

-ISBN: String
-title: String
-author: String
-price: double
+getISBN(): String
+getTitle(): String
+getAuthor(): String
+getPrice(): double

Orders

-order_id: Integer
-customer: String
-books: String
-status: String
+get_order_id(): Integer
+get_customer(): String
+get_books(): String
+get_status(): String

Inventory

-books_available: String
+check_stock(): String
+update_stock(): String
+toString(): String

Customer

-name: String
-email: String
-address: String
-shopping_cart: String
+add_to_cart(): String
+remove_from_cart(): String
+place_order():string
+track_order(): String
+toString(): String

Payment info

-card_number: Integer
-expiry date: date
-CVV: Integer
-billing_add: Interger
+getCardNumber()
+getExpiryDate()
+toString()

Benefits of Class Diagram

Visualization tool: The diagram provides a clear visual of the books available in the library and how customers can order and track their orders. They show classes, attributes relationships, and access modifiers

- **Communication tool:** It serves as an effective tool between the designers, developers, and stakeholders, conveying system design and behavior in a standardized format that can be easily understood.
- **Identifying relationships:** Inheritance and association relationships can be identified easily in the diagram, which assists in understanding the interconnection of different systems.
- **Framework implementation:** It serves as a tool for implementing systems code, which assists the developers to understand the structure, hence providing a solid foundation for the development process.
- **Design and planning:** The diagram above aids in the early stages of book library design and planning as it identifies the key classes and their relationships
- **Maintenance and Refactoring**: When any change is required in the Book library, these diagrams serve as a guide in implementing any change required. Developers can identify impacted classes and update them accordingly. This makes refactoring easier as dependencies and associations can be identified easily.
- **Documentation tool:** It is an essential tool used to understand how the system is interconnected, which is a referral and a training tool for the new developers.

Limitations of Class Diagram

Complexity management: As the system grows and expands, it becomes challenging to update and maintain the class diagrams. Large systems make class diagrams complex to understand and, hence can be confusing.

- **Inflexibility:** Class diagrams above may be rigid and may not accommodate changes or any iterations in the system design. Making changes to class diagrams is time-consuming and hard to arrive at a substantial decision when implementing a change.
- **Lack of implementation details:** Class diagrams do not specify implementation details such as algorithms and database schema. Developers must rely on other documentation or comments within the code to understand implementation specifics.
- **Static View only:** They only represent the static view and no dynamic behavior is depicted. They do not show how objects interact at runtime or handle events.
- **Maintenance challenge:** It is challenging to maintain the class diagrams above as the system and technology evolve. Keeping the diagrams synchronized with the changes in the codes and structure is challenging and requires effort and dedication
- **Abstraction level:** The class diagrams above operate in high levels of abstraction with a focus on static structure systems only. They may not capture dynamic aspects such as behavior, interactions, or state of changes