**Report**

**1 Singleton Pattern (Database Connection)**

The **Singleton Pattern** ensures that only one instance of the database connection exists throughout the system.

* **Application:**
  + A private constructor prevents direct instantiation.
  + A static instance variable holds the single instance.
  + A thread-safe GetInstance() method ensures only one instance is used.
  + A \_lock ensure only one thread is running.
* **Benefits:**
  + Prevents multiple connections to the database.
  + Reduces resource consumption and improves efficiency.

**Implementation Example:**

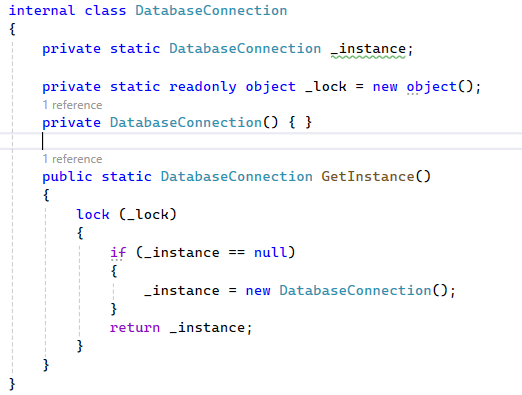


Figure .DatabaseConnection Class

**2 Factory Method Pattern (Document Creation)**

The **Factory Method Pattern** is used to create document objects dynamically, ensuring flexibility when adding new document types.

* **Application:**
  + A factory method takes a document type string and returns the appropriate document instance.
  + Subclasses represent different document types, ensuring encapsulation.
* **Benefits:**
  + Enables easy extension by adding new document types without modifying existing code.
  + Improves code maintainability and scalability.

**Implementation Example:**



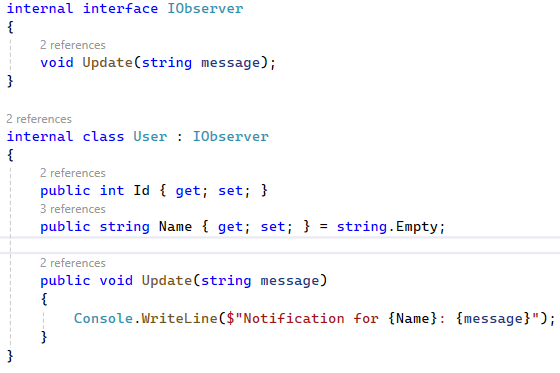
Figure .Applied of Refactory pattern for Document

**3 Observer Pattern (Notification System)**

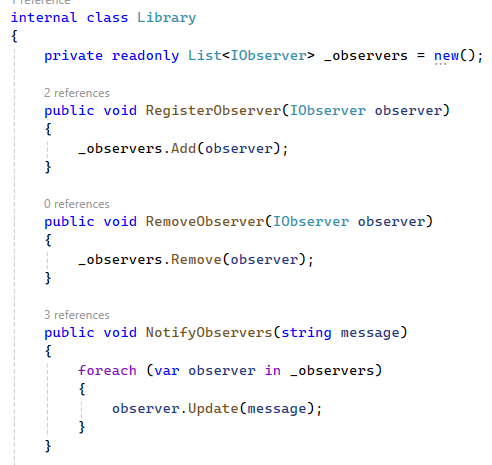
The **Observer Pattern** enables users to receive notifications when documents are added, borrowed, or returned.

* **Application:**
  + A list of registered observers (users) subscribes to the library system.
  + When an event occurs, all observers receive an update.
* **Benefits:**
  + Decouples notification logic from the core system.
  + Allows multiple users to subscribe/unsubscribe dynamically.

**Implementation Example:**



The above image is the implementation in the user class, then all users have the Update() method.



Library which including all users must be passed the observer to use all the method in the Observer.

**4 Strategy Pattern (Loan Fee Calculation)**

The **Strategy Pattern** allows dynamic selection of loan fee calculation logic based on document type.

* **Application:**
  + Different fee calculation strategies are implemented as separate classes.
  + The appropriate strategy is selected at runtime based on the document type.
* **Benefits:**
  + Supports multiple pricing strategies without modifying existing code.
  + Encourages code reuse and flexibility.

**Implementation Example:**

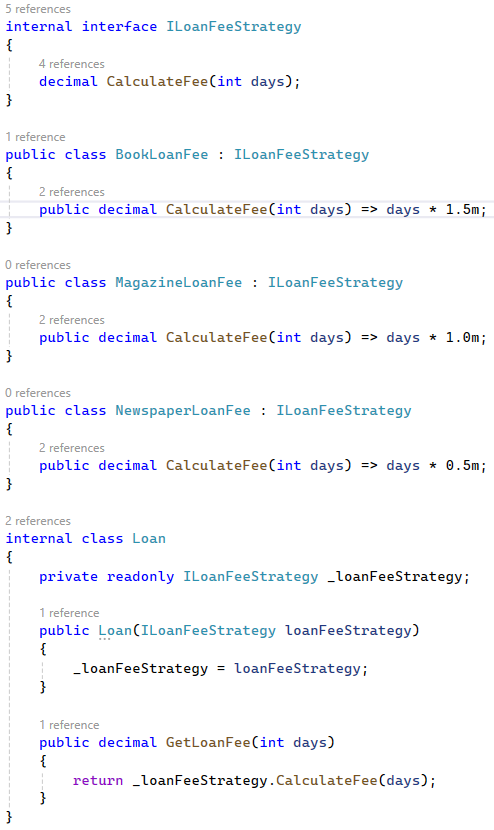


Figure . Apllied of Strategy pattern