**Project Overview**

This Java project is a simple product management system that employs several design patterns to enhance its structure and functionality. The primary design patterns used in this project include Singleton, Observer, Adapter, Decorator, Factory, and Strategy.

**Project Components**

**1. Product Class:**

* Represents a product with attributes such as name, barcode, and price.
* Implements the toString method for formatted product display.

**2. ProductAdapter:**

* Implements the Adapter pattern to adapt LegacyProduct instances to the new Product format.
* Utilizes the ProductFactory for creating new Product instances based on LegacyProduct data.

**3. ProductFactory:**

* Utilizes the Factory pattern to create instances of Product.
* Provides a centralized point for creating Product objects.

**4. ProductSortingStrategy Interface:**

* Represents the Strategy pattern for sorting products.
* Defines the sort method to be implemented by concrete strategy classes.

**5. DefaultProductSortingStrategy:**

* Concrete implementation of the sorting strategy interface.
* Default strategy that returns the list unchanged.

**6. NameProductSortingStrategy:**

* Concrete sorting strategy by product name in alphabetical order.

**7. PriceProductSortingStrategy:**

* Concrete sorting strategy by product price in ascending order.

**8. Observer Interface:**

* Represents the Observer pattern for observing changes in the product warehouse.
* Defines the update method to be implemented by concrete observer classes.

**9. WarehouseObserver:**

* Concrete implementation of the observer interface.
* Observes changes in the product warehouse and updates accordingly.

**10. ProductDecorator:**

* Implements the Decorator pattern to add functionality to the Product class.
* Decorates Product instances with additional features.

**11. ProductWarehouse:**

* Represents the product storage system as a Singleton.
* Utilizes the Observer pattern to notify observers (WarehouseObserver) of changes.
* Supports adding, removing, and sorting products.

**12. Main Class:**

* Contains the main application logic.
* Provides a menu-driven interface for users to interact with the product warehouse.
* Demonstrates the use of various design patterns throughout the application.

**Project Workflow**

1. **Adding Products:**
   * Allows users to add products in both legacy and new formats.
   * Utilizes the Adapter pattern to adapt legacy products to the new format.
2. **Removing Products:**
   * Enables users to remove products from the warehouse.
3. **Viewing Products:**
   * Utilizes the Decorator pattern to add additional features to products.
   * Supports sorting products based on different criteria using the Strategy pattern.
4. **Sorting Products:**
   * Users can choose between sorting by price or by name.
5. **Finding Products:**
   * Enables users to find products by barcode.
6. **Exiting the Application:**
   * Allows users to exit the application.

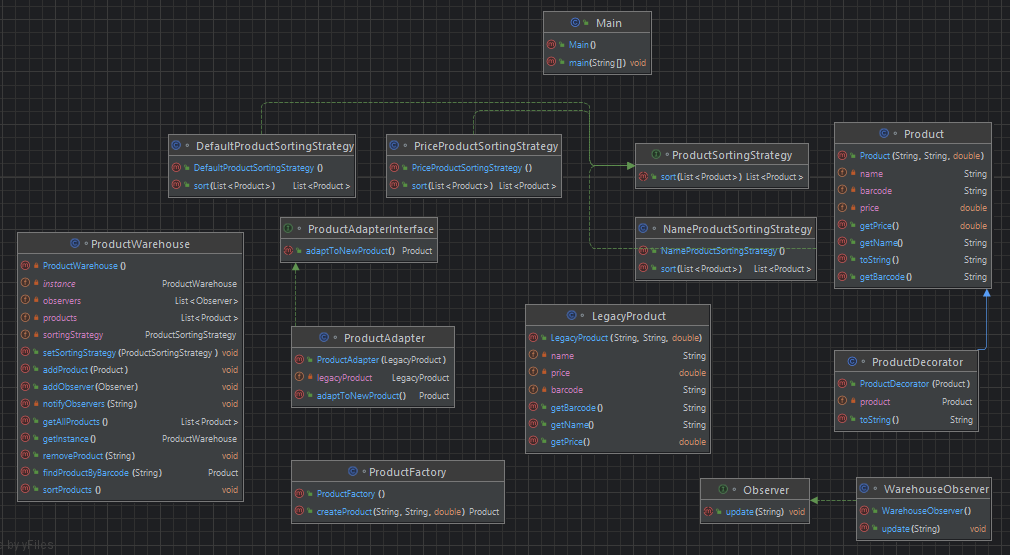
**Challenges and Solutions**

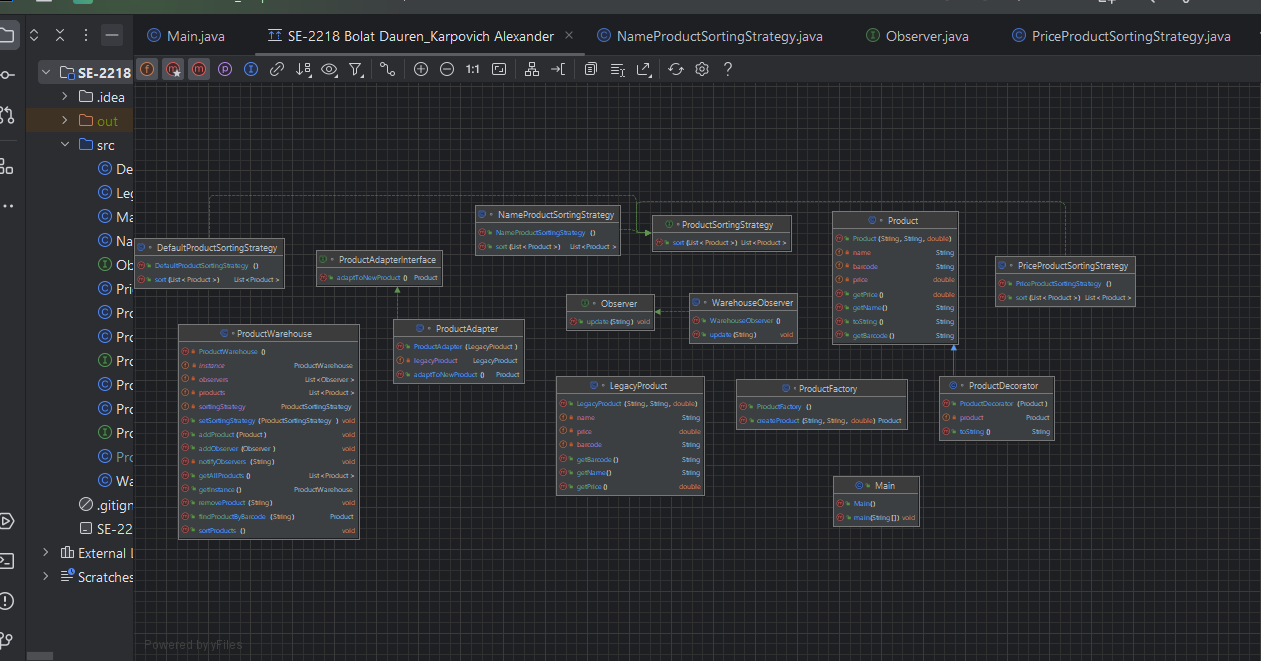
* **Legacy and New Product Formats:**
  + The need to support both legacy and new product formats was addressed using the Adapter pattern.
* **Sorting Strategies:**
  + Implementing sorting strategies required the use of the Strategy pattern, allowing for easy extensibility.
* **Menu-Driven Interface:**
  + Designing a user-friendly menu system involved careful input handling and error checking.

**Future Improvements**

* **User Authentication:**
  + Implement user authentication to secure access to the product warehouse.
* **Database Integration:**
  + Integrate a database for persistent storage of product data.
* **Enhanced Sorting Options:**
  + Provide more sorting options and allow users to define custom sorting criteria.
* **GUI Interface:**
  + Develop a graphical user interface for a more user-friendly experience.
* **Error Handling:**
  + Implement robust error handling mechanisms for better user feedback.

**UML Diagram**

****

****

**Conclusion:**

This project demonstrates the application of several design patterns in creating a flexible and extensible product management system. The use of Singleton, Observer, Adapter, Decorator, Factory, and Strategy patterns enhances the project's maintainability, scalability, and readability. Future improvements aim to address security, database integration, additional sorting options, and a graphical user interface.

**Literature**

1. *Паттерны проектирования*. Паттерны/шаблоны проектирования. (n.d.). <https://refactoring.guru/ru/design-patterns>
2. Zimmer, W. (1995). Relationships between design patterns. *Pattern languages of program design*, *1*, 345-364.
3. Cooper, J. W. (2000). Java design patterns: a tutorial.
4. Marinescu, F. (2002). *EJB design patterns* (pp. 70-75). New York: Wiley.