Implementation of Facade Design Pattern – Evisa Nela

Structural Design Pattern – Facade

In the Furniture Management System, the Facade Design Pattern is applied to simplify and centralize access to the Inventory module's internal subsystems. Rather than exposing the complexity of multiple collaborating classes such as Stock, SupplierManager, OrderManager, and MaterialShortageRequestHandler, a single high-level class—InventoryFacade—is introduced to provide an easy-to-use interface for inventory-related operations.

- A centralized class InventoryFacade defines methods such as addRawMaterial(...), viewStockLevels(), and handleMaterialShortageRequest(...), which internally coordinate the appropriate logic across the subsystem classes.
- The subsystems include Stock for quantity control, SupplierManager for sourcing materials, OrderManager for managing restocking orders, and MaterialShortageRequestHandler for logging critical shortages.
- The pattern hides subsystem complexity from external actors, promoting maintainability, scalability, and clean architecture.

This design ensures that external modules (e.g., manufacturer employee, operations manager, or admin) interact only with the InventoryFacade, not with the underlying implementation classes. It supports the Open/Closed Principle—new features can be added by extending subsystem classes without modifying the client code.

Subsystem Breakdown

1. Stock

Role: Manages and tracks current quantities of raw materials.

Handles cases like:

- Updating material quantities
- Retrieving current stock levels
- Checking for stock shortages

2. SupplierManager

Role: Identifies the best supplier for a given material request.

Handles cases like:

- Looking up suppliers by material type
- Selecting the most reliable or cost-effective source

3. OrderManager

Role: Creates and manages material orders.

Handles cases like:

- Generating new Order objects
- Assigning materials, quantities, and suppliers to each order

4. MaterialShortageRequestHandler

Role: Logs shortage requests and urgency notes.

Handles cases like:

- Recording material shortage events
- Escalating shortage alerts with notes and metadata

Example Scenario Walkthrough

5. Case 1: A manufacturer employee wants to view stock levels for "Steel"

→ InventoryFacade.viewStockLevels("Steel")
→ Calls Stock.getLevel("Steel") → ✅ Returns current quantity

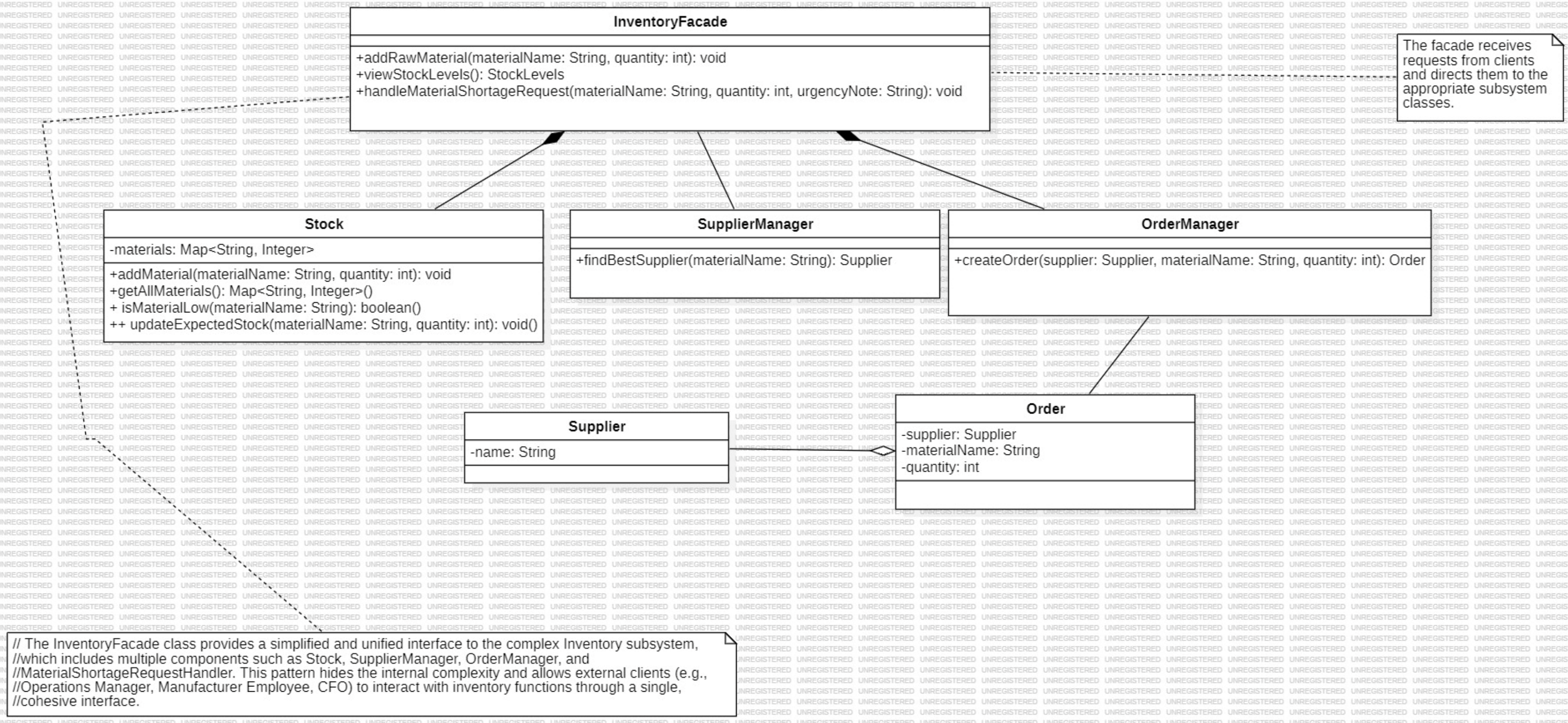
6. Case 2: A material shortage for "Wood" is detected

→ InventoryFacade.handleMaterialShortageRequest("Wood", 100, "Urgent for production")
→ Stock confirms low quantity
→ SupplierManager finds best supplier
→ OrderManager generates a purchase order
→ MaterialShortageRequestHandler logs the shortage event

7. Case 3: The operations manager adds new materials

→ InventoryFacade.addRawMaterial("Metal", 250)
→ Delegates to Stock.updateQuantity(...) → ✅ Material quantity updated

This pattern greatly improves maintainability and modularity by offering a single access point for inventory functions, while hiding internal logic and reducing system-wide coupling. It ensures that each subsystem class handles only the tasks it is responsible for, while the InventoryFacade manages coordination across them. This results in cleaner, more maintainable code and a highly extensible system structure.


InventoryFacade
+addRawMaterial(materialName: String, quantity: int): void
+viewStockLevels(): StockLevels
+handleMaterialShortageRequest(materialName: String, quantity: int, urgencyNote: String): void
The facade receives requests from clients and directs them to the appropriate subsystem classes.
Stock
-materials: Map<String, Integer>
+addMaterial(materialName: String, quantity: int): void
+getAllMaterials(): Map<String, Integer>()
+ isMaterialLow(materialName: String): boolean()
++ updateExpectedStock(materialName: String, quantity: int): void()
SupplierManager
+findBestSupplier(materialName: String): Supplier
OrderManager
+createOrder(supplier: Supplier, materialName: String, quantity: int): Order
Supplier
-name: String
Order
-supplier: Supplier
-materialName: String
-quantity: int
// The InventoryFacade class provides a simplified and unified interface to the complex Inventory subsystem,
//which includes multiple components such as Stock, SupplierManager, OrderManager, and
//MaterialShortageRequestHandler. This pattern hides the internal complexity and allows external clients (e.g.,
//Operations Manager, Manufacturer Employee, CFO) to interact with inventory functions through a single,
//cohesive interface.