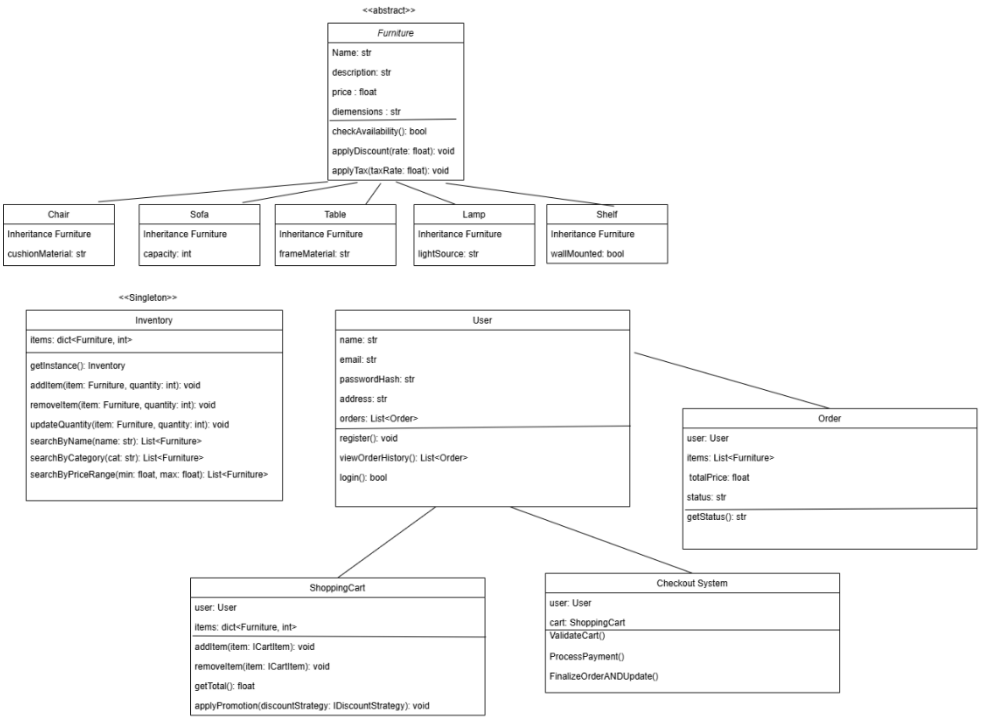
Project design

HW. 4 – First design of the project

This design centers on an abstract Furniture class (with properties like name, price, and dimensions) from which concrete furniture types - Chair, Sofa, Table, Lamp, and Shelf inherit their specialized attributes (e.g., cushionMaterial for chairs, lightSource for lamps). A Singleton Inventory class manages all furniture items, supporting add, remove, update, and searches by name, category, or price range. The User class holds personal information (such as email, password, and address) and can register, log in, and view order history. Each Order associates a user with a list of furniture items, tracking the total price and status. A ShoppingCart (tied to a specific user) allows for adding, removing, and totaling items, with optional promotion/discount handling. Finally, the Checkout System coordinates cart validation, payment processing, and order finalization, completing the purchase flow.

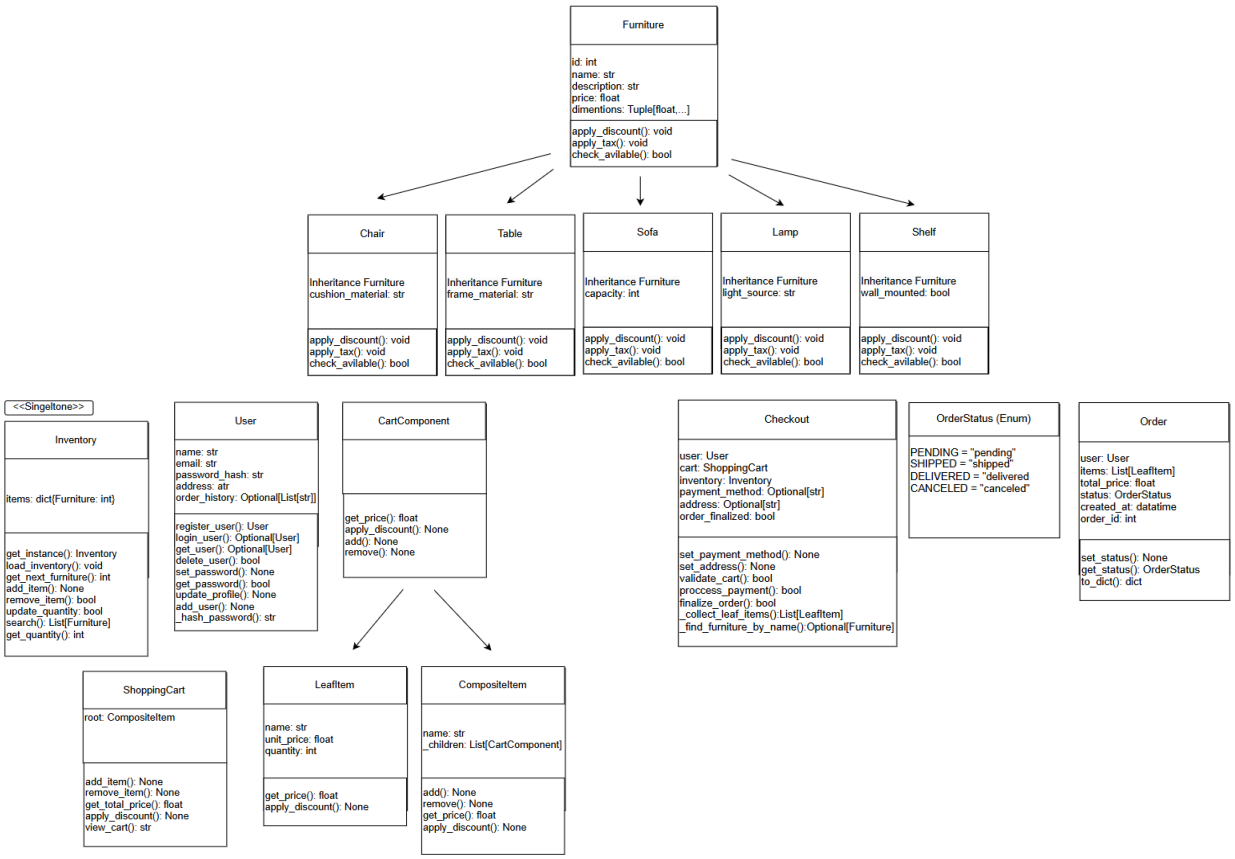


The design through the project

This design begins with the Furniture class (containing base attributes such as id, name, price, and dimensions), which the concrete subclasses - Chair, Table, Sofa, Lamp, and Shelf extend with their own specific fields (e.g., cushion_material in Chair). A singleton Inventory stores and manages furniture data, allowing for quantity updates, item searches, and more.

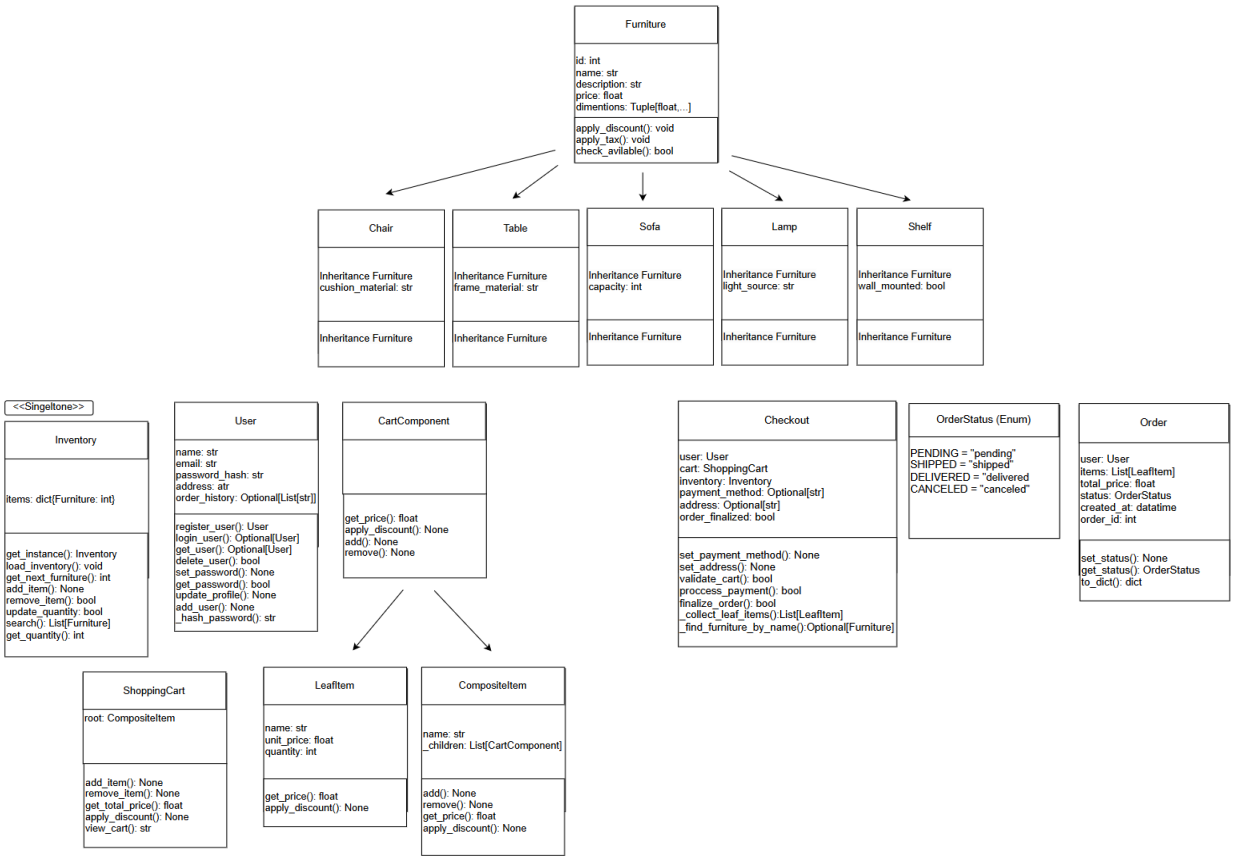
A composite pattern is used for the shopping cart: the abstract CartComponent can represent either a LeafItem (an individual product with a name, unit price, and quantity) or a CompositeItem (a grouping of multiple cart components). The ShoppingCart class holds these items, provides methods to add or remove them, and supports viewing or discounting the entire cart.

The User class manages registration and login, storing user details like email, password hash, and an optional order history. An Order object then tracks the purchased items, total cost, status (defined by the OrderStatus enum), creation date, and a unique order ID. Finally, the Checkout class coordinates the end-to-end purchase process by validating the cart, selecting a payment method, processing payment, and finalizing the order - tying all components together into a coherent flow.



Final design of the project

This design begins with the Furniture class (containing base attributes such as id, name, price, and dimensions), which the concrete subclasses - Chair, Table, Sofa, Lamp, and Shelf extend with their own specific fields (e.g., cushion_material in Chair). The changes we made from the previous design occur as we decide to inherit the methods from the Furniture class and not to implement them in the subclasses. This change define common properties and behaviors once in a base (parent) class and then share them with multiple subclasses, it boosts code reuse, reduces duplication, and makes the system easier to maintain and extend.



Explanation of the Design main changes in implementation

Comparing the first and final design versions of the project, we can see several improvements in the final design that we changed through the project process:

1. Improved Object-Oriented Structure

- In the first design and through the project design, the Furniture class was abstract, but individual furniture types (Chair, Sofa, Table, etc.) were treated more simply, without a clear object hierarchy.
- The final design refines the hierarchy, ensuring all furniture items inherit from Furniture while incorporating essential attributes like id, name, price, and dimensions.

2. Enhanced Inventory Management

- In the first design, the Inventory class was more basic with redundant functions.
- The final design includes functions for loading and saving data, auto-generating furniture IDs, and performing inventory searches in one function.

3. Better Cart and Composite Design Pattern Implementation

- In the first design ShoppingCart contained items directly in a dictionary.
- The final design uses a Composite Pattern (CartComponent with LeafItem and CompositeItem), allowing flexible handling of items. The ShoppingCart class now has a single root composite item (root), improving scalability.

4. Introduction of Order and Order Status

- In the first design The Order class was minimal, only storing basic user and furniture information.
- The final design Refined Order structure, including Order ID and Status tracking (OrderStatus Enum: Pending, Shipped, Delivered, Canceled). Also, a method to convert orders into a dictionary. The OrderStatus Enum ensures orders follow a proper lifecycle.

5. Separation of Concerns & Cleaner Code

The final design adopts solid principles:

- Single Responsibility Principle: Each class is responsible for its relevant information.
- Encapsulation: Business logic is inside relevant classes (Order, ShoppingCart, Checkout).
- Extensibility: New furniture types, discounts, or tax policies can be added easily.