🧠 Mind Map: Products Module Design

💡 Key Takeaways

- **Modular Design**: The Products module is designed to be independent yet integrable with other system components
- **Flexibility**: The hierarchical category system allows for easy organization and expansion
- **SOLID Compliance**: All design decisions align with SOLID principles for maintainable, extensible code
- **Clear Responsibilities**: Each class has well-defined roles and responsibilities
- **Scalability**: The system can grow to accommodate more product types and categories without redesign

🚀 Next Steps

1. **Implementation**: Develop the core classes and interfaces
2. **Testing**: Create unit tests to verify behavior
3. **Integration**: Connect with Inventory and Sales modules
4. **Refinement**: Optimize based on performance testing# 🛒 Supermarket Management System - Products Module

🔍 Overview

The Products module is the foundational component of our Supermarket Management System. It defines how products are represented, categorized, and managed within the system - essentially forming the backbone of our entire application!

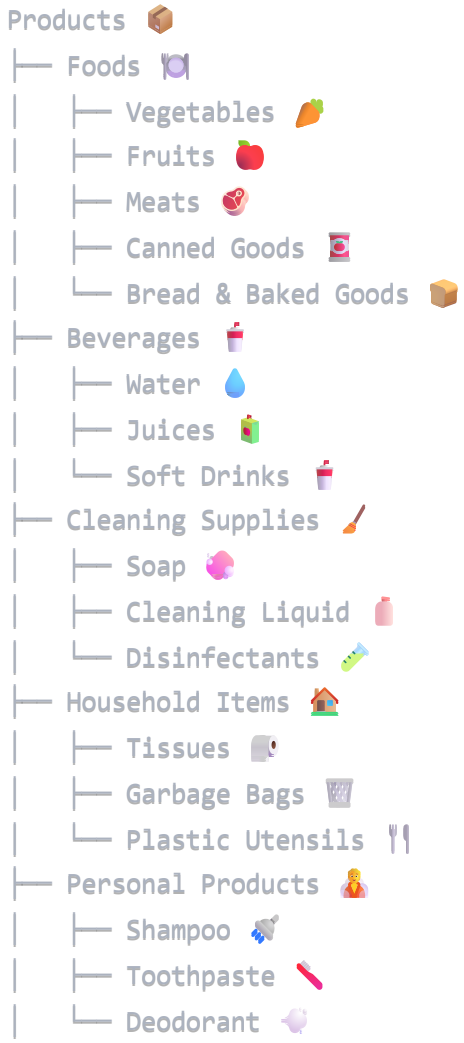
✨ Product Attributes

Each product in the system has the following key characteristics:

1. 📝 **Name**: The name of the product (e.g., "Mango Juice")
2. 🔢 **Unique Identifier (SKU or Barcode)**: To uniquely identify the product
3. 💰 **Price**: The price per unit
4. 🔢 **Available Quantity**: The number of units available in stock
5. 🗓️ **Expiration Date**: The product's expiration date
6. 📁 **Category**: The category the product belongs to (e.g., "Juices")

🌳 Product Categories Hierarchy

Our system implements a hierarchical category structure to organize products logically:



OOP Design for Products

Core Classes and Interfaces

1. **Product Interface** :
   - Defines the contract for all product objects
   - Contains methods for accessing and modifying product attributes
   - Ensures consistency across different types of products
2. **BaseProduct Class** :
   - Implements the Product interface
   - Provides common functionality for all product types
   - Handles basic product operations
3. **Category Class** :
   - Represents a product category in the hierarchy
   - Can contain subcategories (composite pattern)

- Associates products with their appropriate categories

🎯 SOLID Principles Applied

1. **Single Responsibility Principle (SRP)** ✅:
   - Product class handles only product-related concerns
   - Category class manages only categorization logic
   - Separate classes for inventory management, pricing strategies, etc.
2. **Open/Closed Principle (OCP)** 🚪:
   - Design allows adding new product types without modifying existing code
   - Category system can be extended with new categories without changing the core structure
3. **Liskov Substitution Principle (LSP)** 🔄:
   - All product implementations can be used interchangeably where the Product interface is expected
   - Ensures consistent behavior across different product types
4. **Interface Segregation Principle (ISP)** 🧩:
   - Specialized interfaces for specific product behaviors (e.g., perishable products)
   - Prevents classes from implementing methods they don't need
5. **Dependency Inversion Principle (DIP)** ⚙️:
   - High-level modules (like inventory) depend on abstractions (Product interface)
   - Not dependent on concrete implementations

🔄 Product Operations

The Products module supports the following key operations:

1. **Product Creation** ➕:
   - Adding new products to the system with all required attributes
   - Assignment to appropriate categories
2. **Product Updates** 🔄:
   - Modifying product details (price, quantity, expiration date)
   - Tracking changes for inventory management
3. **Product Queries** 🔍:
   - Searching for products by various criteria
   - Filtering products by category, price range, availability, etc.

4. **Product Removal** ❌:
    - Safely removing products from the system
    - Handling associated inventory adjustments

🔗 Integration Points

The Products module integrates with other system components:

1. **Inventory Management** 📦:
    - Products provide the basis for inventory tracking
    - Stock levels are managed through product quantity attributes
2. **Sales System** 💰:
    - Products appear in customer shopping carts and receipts
    - Price information flows from products to sales calculations
3. **Reporting System** 📊:
    - Product data serves as the foundation for various reports
    - Sales analytics, inventory status, and product performance metrics

⚠️ Project Simplifications

For the initial implementation:

- No persistent storage (like databases) 💾
- No tracking of product addition or modification history 📜
- Focus only on objects and their relationships 🔄