ProductCatalogManager

Product Attribute

- name: string // e.g., "OS"

- value: string // e.g., "Android 15"

+ constructor (name, value)

Product

-id: string

- categoryid: string

- name: string

- price: number

- image URL: string

- attributes: Map<string, string> // e.g., {"OS": "Android 15"}

+ constructor (id, name, catId, price, imgUrl, attrs)

- id: string

- name: string

- attributes: string[] // e.g., ["OS", "RAM"]

+ constructor(id, name, attributes)

Category

-Categories: Category []

-products: Products []

+ init(): void

+ load Data (): void

+ saveData (): void

+ showNotification (msg, type): void

+ addCategory(name, attrs): void

+ renderCategories(): void

+ addProduct(name, catId, price, imgUrl, attrs): void

+ updateProduct(id, name, catId, price, imgUrl, attrs): void

+ deleteProduct(id): void

+ renderProducts(filterCatId): void

+ clearProductForm(): void

0..\*

1

1

Attributes

1

0..\*…..

0..\*

**Class Diagram**

**Class Relationships and Responsibilities**

**1. ProductCatalogManager**

* **Responsibility:** This class is the **central control hub** of Ecommerce application. It acts as the orchestrator, managing the entire product catalog, including categories and individual products. It's also responsible for handling data persistence (saving and loading) and user interactions.
* **Key Methods:**
  + init(): Gets the application ready, loading data and setting up the display.
  + loadData(): Fetches saved categories and products from your browser's local storage.
  + saveData(): Stores the current state of categories and products back into local storage.
  + showNotification(msg, type): Displays brief messages (like "Product saved!") to inform the user about actions.
  + addCategory(name, attrs): Creates and adds new product categories with their defining attributes.
  + renderCategories(): Updates the part of the webpage that shows all the categories.
  + addProduct(...): Adds a brand new product to the catalog.
  + updateProduct(...): Changes the details of an existing product.
  + deleteProduct(id): Removes a product permanently.
  + renderProducts(filterCatId): Updates the display of products on the webpage, optionally showing only products from a specific category.
  + clearProductForm(): Resets all the input fields in the product entry form.
* **Relationships:**
  + **Manages Category (Aggregation):** The line with the empty diamond (o--) shows that ProductCatalogManager **manages** many Category objects. It holds lists of categories, but the categories could technically exist independently of this specific manager instance (though in this app, it's the main way they're handled).
    - "1": Means there's one ProductCatalogManager instance.
    - "0..\*": Means it can manage zero to many Category objects.
  + **Manages Product (Aggregation):** Similar to categories, the ProductCatalogManager **manages** many Product objects, holding them in its internal lists and orchestrating their creation, updates, and deletion.
    - "1": Means there's one ProductCatalogManager instance.
    - "0..\*": Means it can manage zero to many Product objects.

**2. Category**

* **Responsibility:** This class defines a **type** or group of products. It specifies the common characteristics (attributes) that products belonging to this type should have.
* **Key Attributes:**
  + id: A unique code for this category (e.g., "cat-smartphones").
  + name: The display name of the category (e.g., "Smartphones").
  + attributes: A list of strings that name the custom attributes expected for products in this category (e.g., ["OS", "RAM", "Battery Size"]).

**3. Product**

* **Responsibility:** This class represents an **individual item** that is being sold. It holds all the specific details about that particular product.
* **Key Attributes:**
  + id: A unique code for this specific product item.
  + categoryId: Links this product to its parent Category, indicating its type.
  + name: The specific name of the product.
  + price: The cost of the product.
  + imageUrl: The web address of the product's image.
  + attributes: This is a map (or dictionary) that stores the *actual values* for the dynamic characteristics of this product (e.g., {"OS": "Android 15", "RAM": "12GB"}).
* **Relationships:**
  + **Has ProductAttribute values (Aggregation):** The empty diamond (o--) here signifies that a Product **has** or *uses* many ProductAttribute values. While the diagram shows ProductAttribute as a separate class, in your code, these are often directly embedded as key-value pairs within the Product's attributes map. This relationship emphasizes that the attribute values are part of the product's description, and while tightly linked, they are conceptual individual parts of the product's data.
    - "1": Means one Product object.
    - "0..\*": Means a product can have zero or many Product Attribute values.

**4. Product Attribute**

* **Responsibility:** This class represents a **single, specific characteristic** of a product, like a color, size, or operating system detail. It pairs an attribute name with its corresponding value.
* **Key Attributes:**
  + name: The specific name of the attribute (e.g., "Dial Color").
  + value: The data for that attribute (e.g., "Black").