**Short Descriptions for each Class**

* **MarketFacade:** Responsible for representing the overall system, connecting the facades with each other and controlling the overall flow of the system.
* **PaymentServiceHandler:** Responsible for managing the payment services offered by the system: adding, removing, changing, etc.
* **DeliveryServiceHandler:** Responsible for managing the delivery services offered by the system: adding, removing, changing etc.
* **Notifier:** Responsible for notifying the appropriate users with the respective notification, will be expanded into subclasses for specific types of notifications later on. Works similar to an observer.
* **SearchFacade:** Responsible for providing the logic of the different search mechanisms available by the system, specific products with certain filters, searching in a specific store, searching by categories, by tags, etc.
* **AuthenticationManager:** Responsible for the authentication of users who attempt to login to the system, providing session tokens to successful logins of users.
* **Address:** Representing the data required for addresses, useful for many classes throughout our system. Used to identify which selling laws are applicable based on country location, saved billing addresses of a certain member, etc.
* **StoreFacade:** Responsible for managing the stores of the system and providing their functionality overall.
* **Store:** Represents a store, provides the functionality of a store. Includes a list of the products available in a store. Can be active or inactive in case the store was closed.
* PolicyStrategy: Interface representing the strategy design pattern responsible for the policies that are possible for a store.
* **ShoppingCart:** Represents a user’s shopping cart, unique for each user and not shared and includes a list of the shoppingBaskets of the user.
* **ShoppingBasket:** Represents a user’s shopping basket for a specific store, able to add or remove products from the basket. Unique for each <user,store> combination.
* **UserFacade:** Responsible for managing the users of the system and their functionalities as a whole. Includes a list of the overall users.
* **User:** Represents the users of the system. A user can either be a member or a guest depending on their state.
* **State:** Design pattern to allow the user to move from one state to another (User or guest).
* **Guest:** Represents a user that has not logged in or registered yet into the system. Has a unique shoppingCart that gets deleted when the guest exits out of the system. Responsible for the functionality of a guest.
* **Member:** Represents a registered user that has logged in. Has a unique shoppingCart that is saved at all times. Responsible for extending the functionality of a guest with additional functionality. Includes a list of roles (can be empty) that describes the member’s privileges in the system.
* **Notification:** A list of notifications the member received, marks notifications as read or unread and allows the member to see their notifications at all times.
* **Role:** Interface responsible for representing the different roles a member can have in our system. A user may have multiple roles and can have the same role multiple times (a member can have multiple storeOwner roles for example since they can own multiple stores, but a member can’t have multiple StoreManager roles).
* **SystemManager:** Role of a member that represents a manager of the overall system. Responsible for defining a system manager’s functionalities and privileges in the system. Each user can be an EcommerceManager only once. Includes the parentId of the EcommerceManager that promoted the member to an EcommerceManager.
* **StoreManager:** Role of a member that represents a manager of a store, includes the parentId of the storeManager/storeOwner that promoted said user to a manager. Defines the functionalities and privileges of the member as a manager with respect to the given permissions by the parent.

* **StoreOwner:** Role of a member that represents an owner of a store, includes the parentId of the storeOwner that promoted said user to an owner (the original owner has a unique parentId that describes said user as the original owner). Defines the functionalities and privileges of the store owner.
* **RoleFacade:** Responsible for managing and storing the roles of the system/stores and their relationships between the roles.
* **PurchaseFacade:** Responsible for the functionalities and logic of performing a purchase in the system. Supports the many types of purchases in our system that will be implemented. Utilizes the purchaseServiceHandler and the DeliveryServiceHandler for successful purchases.
* **Purchase:** Interface that represents the different types of purchases possible in the system.
* **Discount:** Interface that represents the different types of discounts available in the system.
* **Rating:** Interface that represents the different types of ratings available in the system. More Specifically, it represents the ratings of a product and a store. Users can only rate a product they purchased and store they purchased from.
* **Product:** Represents a physical product in a store, has the unique attributes to said product, for example: productId, date of expiration, condition, etc. Can’t have two of the same product. Each product is connected to their respective productSpecification.
* **ProductSpecification:** Abstract representation of a product and its description, includes the attributes: name of the product, manufacturer, tags, rating, description, weight etc. Not unique per store.
* **Category:** The categories that a productSpecification falls under, used for searching via categories. Composite object, meaning that categories can have sub categories.
* **ProductConditions:** Enumeration responsible for describing the different types of conditions that a physical product can be in. For now, the system supports two conditions: new and used.