01-06-2023

**BREIF EXPLANATION OF THE OBJECT-ORIENTED DESIGN DRAWN FOR ECOMMERCE SOLUTION**

A class diagram is a visual representation of the structure and relationships among classes in an object-oriented system. It provides a high-level overview of the system's architecture, depicting the classes, their attributes, and their relationships with other classes. In a class diagram, classes are represented as boxes, with their name at the top and their attributes and methods listed inside. Arrows are used to show the relationships between classes, such as associations, dependencies, inheritances, and aggregations. Attributes represent the data or characteristics associated with a class, while methods represent the behaviours or operations that the class can perform. These attributes and methods define the class's state and behavior.

In the OOD drawn for the ecommerce solution:

* Light Blue colour is used for the class name.
* Light Grey colour is used to depict the class attributes
* Light Yellow colour is used to depict the class functions.

Functions are represented as function name-> input type-> return type

A brief explanation for each of the classes, along with a description for their functions is given below:

1. **Customer:**

In the class diagram of an ecommerce website, the Customer class represents a customer who interacts with the website.

1. Registration:

* register (username, password, email): Registers a new customer account with the provided username, password, and email.

1. Authentication and Account Management:

* Login (username, password): Authenticates a customer by verifying their username and password.
* Logout (): Logs out the currently logged-in customer.
* updateProfile (name, email, address): Updates the customer's profile information, such as name, email, and address.
* changePassword (oldPassword, newPassword): Allows the customer to change their account password.

1. Product Interaction:

* browseProducts (): Allows the customer to browse available products on the website.
* searchProducts (keyword): Enables the customer to search for products using a keyword or phrase.
* viewProduct (productId): Displays detailed information about a specific product.
* addToCart (productId, quantity): Adds a product to the customer's shopping cart with the specified quantity.
* removeFromCart (productId): Removes a product from the customer's shopping cart.
* getCartItems (): Retrieves the list of products in the customer's shopping cart.

1. Order Management:

* placeOrder (): Places an order for the products in the customer's shopping cart.
* cancelOrder (orderId): Cancels a previously placed order by its ID.
* getOrderHistory (): Retrieves the customer's order history, including past orders and their details.

1. Wishlist:

* addToWishlist (productId): Adds a product to the customer's wishlist for future reference or purchase.
* removeFromWishlist (productId): Removes a product from the customer's wishlist.
* getWishlist (): Retrieves the customer's wishlist, containing the products they have saved for later.

1. Reviews and Ratings:

* addReview (productId, rating, comment): Allows the customer to add a review and rating for a product.
* updateReview (reviewId, rating, comment): Updates an existing review and rating given its ID.
* deleteReview (reviewId): Deletes a review and rating based on its ID.

1. Notifications:

* enableNotifications (): Enables the customer to receive notifications about order updates, promotions, etc.
* disableNotifications (): Disables notifications for the customer.

1. **Order:**

In the class diagram of an ecommerce website, the Order entity represents an order placed by a customer. The functions provide the necessary operations to manage the details of an order, including adding or removing products, updating the order date, and tracking the order status.

1. getId (): String: This function returns the unique identifier of the order.
2. getCustomer (): Customer: This function retrieves the customer who placed the order.
3. getProducts (): List<Product>: This function returns the list of products included in the order.
4. getOrderDate (): Date: This function retrieves the date when the order was placed.
5. getStatus (): OrderStatus: This function returns the current status of the order, such as "pending," "shipped," or "delivered." OrderStatus can be an enumeration or a separate class.
6. addProduct (product: Product): void: This function adds a product to the order.
7. removeProduct (product: Product): void: This function removes a product from the order.
8. setOrderDate (date: Date): void: This function sets the order date to the specified value.
9. setStatus (status: OrderStatus): void: This function updates the status of the order.
10. **Order Details:**

In the class diagram of an ecommerce website, the OrderDetails entity represents the details of a specific order. These functions allow you to access and manage the specific details of an order, such as the product, quantity, and subtotal amount associated with that order. Additionally, the calculateSubtotal () function helps ensure that the subtotal is accurately calculated based on the current quantity and price.

1. getId (): String: This function returns the unique identifier of the order details.
2. getOrder (): Order: This function retrieves the order to which these details belong.
3. getProduct (): Product: This function retrieves the product associated with these order details.
4. getQuantity (): int: This function returns the quantity of the product included in the order.
5. getSubtotal (): double: This function returns the subtotal amount for the specific product in the order.
6. setQuantity (quantity: int): void: This function sets the quantity of the product in the order.
7. setSubtotal (subtotal: double): void: This function sets the subtotal amount for the specific product in the order.
8. calculateSubtotal (): void: This function calculates the subtotal amount based on the quantity and price of the product. It can be used to automatically update the subtotal whenever the quantity or price changes.
9. **Products:**

This entity has been represented as a class having only attributes. It doesn’t have any functionalities as in.

1. **Supplier and Shipper:**

Both of these entities have the same functionalities as mentioned below.

The primary difference is that a Supplier is the producer of the product while a Shipper is the deliverer of the product.

1. getId (): String: Retrieves the identifier of the shipper.
2. getName (): String: Retrieves the name of the shipper.
3. getContactNumber (): String: Retrieves the contact number of the shipper.
4. setContactNumber (contactNumber: String): void: Sets the contact number of the shipper.
5. getEmail (): String: Retrieves the email address of the shipper.
6. setEmail (email: String): void: Sets the email address of the shipper.
7. getAddress (): String: Retrieves the address of the shipper.
8. setAddress (address: String): void: Sets the address of the shipper.
9. shipOrder (order: Order): void: Initiates the shipping process for a specific order.
10. updateShippingStatus (order: Order, status: ShippingStatus): void: Updates the shipping status of an order. The ShippingStatus can be an enumeration or a separate class representing different stages of the shipping process (e.g., "pending," "in transit," "delivered").
11. generateTrackingNumber (order: Order): String: Generates a tracking number for an order to track its shipment progress.
12. notifyCustomer (order: Order, message: String): void: Sends a notification to the customer regarding the shipping status or any updates related to their order.
13. **Personalisation:**

In the context of an ecommerce website, a "Personalization" entity can represent the personalized aspects of the user experience. These attributes and functions provide a basic framework for personalization in an ecommerce website. The "Personalization" entity allows you to store and manage personalized data such as preferred categories, recommended products, recently viewed products, cart reminder settings, and notification preferences for each customer.

1. getCustomerId (): String: Retrieves the identifier of the customer.
2. getPreferredCategories (): List<Category>: Retrieves the list of preferred categories for the customer.
3. addPreferredCategory (category: Category): void: Adds a category to the list of preferred categories.
4. removePreferredCategory (category: Category): void: Removes a category from the list of preferred categories.
5. getRecommendedProducts (): List<Product>: Retrieves the list of recommended products for the customer.
6. setRecommendedProducts (products: List<Product>): void: Sets the list of recommended products for the customer.
7. getRecentlyViewedProducts (): List<Product>: Retrieves the list of recently viewed products by the customer.
8. addToRecentlyViewedProducts (product: Product): void: Adds a product to the list of recently viewed products.
9. clearRecentlyViewedProducts (): void: Clears the list of recently viewed products.
10. isCartRemindersEnabled (): boolean: Checks if cart reminders are enabled for the customer.
11. setCartRemindersEnabled (enabled: boolean): void: Sets the flag indicating whether cart reminders are enabled.
12. getNotificationPreferences (): NotificationPreferences: Retrieves the customer's notification preferences.
13. setNotificationPreferences (preferences: NotificationPreferences): void: Sets the customer's notification preferences.