```java
package salableProduct;

/**
 * Represents a salable product in the store.
 */
public class SalableProduct {
    private String name;
    private String description;
    private double price;
    private int quantity;

    /**
     * Constructs a SalableProduct object with the given attributes.
     *
     * @param name        the name of the product
     * @param description the description of the product
     * @param price       the price of the product
     * @param quantity    the quantity of the product available in the store
     */
    public SalableProduct(String name, String description, double price, int
quantity) {
        this.name = name;
        this.description = description;
        this.price = price;
        this.quantity = quantity;
    }

    /**
     * Retrieves the name of the product.
     *
     * @return the name of the product
     */
    public String getName() {
        return name;
    }

    /**
     * Retrieves the description of the product.
     *
     * @return the description of the product
     */
    public String getDescription() {
        return description;
    }

    /**
     * Retrieves the price of the product.
     *
     * @return the price of the product
     */
    public double getPrice() {
        return price;
    }

    /**
```

```
     * Retrieves the quantity of the product available in the store.
     *
     * @return the quantity of the product
     */
    public int getQuantity() {
        return quantity;
    }

    /**
     * Sets the name of the product.
     *
     * @param name the name of the product
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * Sets the description of the product.
     *
     * @param description the description of the product
     */
    public void setDescription(String description) {
        this.description = description;
    }

    /**
     * Sets the price of the product.
     *
     * @param price the price of the product
     */
    public void setPrice(double price) {
        this.price = price;
    }

    /**
     * Sets the quantity of the product available in the store.
     *
     * @param quantity the quantity of the product
     */
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    @Override
    public String toString() {
        return "Product Name: " + name + "\n"
                + "Description: " + description + "\n"
                + "Price: $" + price + "\n"
                + "Quantity: " + quantity;
    }
}
```

```java
package salableProduct;

import java.util.List;

/**
 * Represents the store front that interacts with the inventory manager and shopping cart.
 */
public class StoreFront {
    private InventoryManager inventoryManager;
    private ShoppingCart shoppingCart;

    /**
     * Constructs a StoreFront object with the given inventory manager.
     *
     * @param inventoryManager the inventory manager for the store
     */
    public StoreFront(InventoryManager inventoryManager) {
        this.inventoryManager = inventoryManager;
        this.shoppingCart = new ShoppingCart();
    }

    /**
     * Initializes the state of the store.
     */
    public void initializeStore() {
        // Initialize the state of the store
    }

    /**
     * Purchases a salable product and adds it to the shopping cart.
     *
     * @param product the salable product to purchase
     */
    public void purchaseProduct(SalableProduct product) {
        // Add the product to the shopping cart
        shoppingCart.addItem(product);

        // Reduce the quantity of the product in the inventory
        inventoryManager.reduceQuantity(product);
    }

    /**
     * Cancels the purchase of a salable product and removes it from the shopping cart.
     *
     * @param product the salable product to cancel the purchase
     */
    public void cancelPurchase(SalableProduct product) {
        // Remove the product from the shopping cart
        shoppingCart.removeItem(product);

        // Increase the quantity of the product in the inventory
        inventoryManager.increaseQuantity(product);
    }
```

```
    public void displayInventory() {
        List<SalableProduct> products = inventoryManager.getAllProducts();
        System.out.println("------ Inventory ------");
        for (SalableProduct product : products) {
            System.out.println(product);
            System.out.println("-----------------------");
        }
    }

    public void displayShoppingCart() {
        List<SalableProduct> items = shoppingCart.getItems();
        System.out.println("------ Shopping Cart ------");
        for (SalableProduct item : items) {
            System.out.println(item);
            System.out.println("---------------------------");
        }
    }

    public static void main(String[] args) {
        // Create an inventory manager
        InventoryManager inventoryManager = new InventoryManager();

        // Add products to the inventory
        inventoryManager.addProduct(new SalableProduct("Product 1", "Description 1",
10.99, 5));
        inventoryManager.addProduct(new SalableProduct("Product 2", "Description 2",
19.99, 8));
        inventoryManager.addProduct(new SalableProduct("Product 3", "Description 3",
5.99, 3));

        // Create a store front with the inventory manager
        StoreFront storeFront = new StoreFront(inventoryManager);

        // Display the initial inventory
        storeFront.displayInventory();

        // Purchase a product
        SalableProduct product1 = inventoryManager.getProduct("Product 1");
        storeFront.purchaseProduct(product1);

        // Display the updated inventory and shopping cart
        storeFront.displayInventory();
        storeFront.displayShoppingCart();

        // Cancel the purchase
        storeFront.cancelPurchase(product1);

        // Display the updated inventory and shopping cart
        storeFront.displayInventory();
        storeFront.displayShoppingCart();
    }
}
```

```java
package salableProduct;

import java.util.ArrayList;
import java.util.List;

/**
 * Represents the inventory manager that manages the inventory of salable products.
 */
public class InventoryManager {
    private List<SalableProduct> products;

    /**
     * Constructs an InventoryManager object.
     */
    public InventoryManager() {
        this.products = new ArrayList<>();
    }

    /**
     * Adds a salable product to the inventory.
     *
     * @param product the salable product to add
     */
    public void addProduct(SalableProduct product) {
        products.add(product);
    }

    /**
     * Removes a salable product from the inventory.
     *
     * @param product the salable product to remove
     */
    public void removeProduct(SalableProduct product) {
        products.remove(product);
    }

    /**
     * Retrieves a salable product from the inventory based on its name.
     *
     * @param name the name of the product to retrieve
     * @return the salable product if found, null otherwise
     */
    public SalableProduct getProduct(String name) {
        for (SalableProduct product : products) {
            if (product.getName().equals(name)) {
                return product;
            }
        }
        return null;
    }

    /**
     * Retrieves all salable products in the inventory.
     *
     * @return a list of all salable products
```

```
     */
    public List<SalableProduct> getAllProducts() {
        return products;
    }

    /**
     * Reduces the quantity of a salable product in the inventory by 1.
     *
     * @param product the salable product to reduce the quantity
     */
    public void reduceQuantity(SalableProduct product) {
        int currentQuantity = product.getQuantity();
        if (currentQuantity > 0) {
            product.setQuantity(currentQuantity - 1);
        }
    }

    /**
     * Increases the quantity of a salable product in the inventory by 1.
     *
     * @param product the salable product to increase the quantity
     */
    public void increaseQuantity(SalableProduct product) {
        int currentQuantity = product.getQuantity();
        product.setQuantity(currentQuantity + 1);
    }
}
```

```
package salableProduct;

import java.util.ArrayList;
import java.util.List;

/**
 * Represents the shopping cart that manages selected products for purchase.
 */
public class ShoppingCart {
    private List<SalableProduct> items;

    /**
     * Constructs a ShoppingCart object.
     */
    public ShoppingCart() {
        this.items = new ArrayList<>();
    }

    /**
     * Adds a salable product to the shopping cart.
     *
     * @param product the salable product to add to the cart
     */
    public void addItem(SalableProduct product) {
        items.add(product);
```

```
    }

    /**
     * Removes a salable product from the shopping cart.
     *
     * @param product the salable product to remove from the cart
     */
    public void removeItem(SalableProduct product) {
        items.remove(product);
    }

    /**
     * Retrieves all salable products in the shopping cart.
     *
     * @return a list of all salable products in the cart
     */
    public List<SalableProduct> getItems() {
        return items;
    }

    /**
     * Clears the shopping cart, removing all products.
     */
    public void clearCart() {
        items.clear();
    }
}
```