**package** inventoryManagerandShoppingCart;

**import** java.util.Scanner;

/\*\*

\* The StoreFrontApplication class represents an application for managing a store's inventory and shopping cart.

\*/

**public** **class** StoreFrontApplication {

**private** InventoryManager inventoryManager;

**private** ShoppingCart shoppingCart;

/\*\*

\* Constructs a StoreFrontApplication object, initializing the inventory manager and shopping cart.

\*/

**public** StoreFrontApplication() {

inventoryManager = **new** InventoryManager();

shoppingCart = **new** ShoppingCart();

}

/\*\*

\* Displays the inventory with product details.

\*/

**public** **void** displayInventory() {

System.***out***.println("Inventory:");

**int** index = 0;

**for** (SalableProducts product : inventoryManager.getInventory()) {

System.***out***.println("[" + index + "] " + product.getName() + " - " + product.getDescription()

+ " - Price: $" + product.getPrice() + " - Quantity: " + product.getQuantity());

index++;

}

}

/\*\*

\* Purchases a product from the inventory and adds it to the shopping cart.

\*

\* **@param** index the index of the product to purchase

\*/

**public** **void** purchaseProduct(**int** index) {

**if** (index >= 0 && index < inventoryManager.getInventory().size()) {

SalableProducts product = inventoryManager.getInventory().get(index);

**if** (product.getQuantity() > 0) {

inventoryManager.removeProduct(index);

shoppingCart.addProduct(product);

} **else** {

System.***out***.println("Product is out of stock: " + product.getName());

}

} **else** {

System.***out***.println("Invalid product index");

}

}

/\*\*

\* Cancels a purchase by removing a product from the shopping cart and adding it back to the inventory.

\*

\* **@param** index the index of the product to cancel the purchase

\*/

**public** **void** cancelPurchase(**int** index) {

**if** (index >= 0 && index < shoppingCart.getContents().size()) {

SalableProducts product = shoppingCart.getContents().get(index);

inventoryManager.addProduct(product);

shoppingCart.removeProduct(product);

} **else** {

System.***out***.println("Invalid product index");

}

}

/\*\*

\* Displays the contents of the shopping cart.

\*/

**public** **void** displayShoppingCart() {

System.***out***.println("Shopping Cart:");

**int** index = 0;

**for** (SalableProducts product : shoppingCart.getContents()) {

System.***out***.println("[" + index + "] " + product.getName() + " - " + product.getDescription()

+ " - Price: $" + product.getPrice() + " - Quantity: " + product.getQuantity());

index++;

}

}

/\*\*

\* Empties the shopping cart.

\*/

**public** **void** emptyShoppingCart() {

shoppingCart.emptyCart();

}

/\*\*

\* The main entry point of the StoreFrontApplication program.

\*

\* **@param** args command line arguments

\*/

**public** **static** **void** main(String[] args) {

StoreFrontApplication storeFront = **new** StoreFrontApplication();

System.***out***.println("Welcome to the Store Front!");

storeFront.displayInventory();

Scanner scanner = **new** Scanner(System.***in***);

**int** action;

**do** {

System.***out***.println("\nActions:");

System.***out***.println("1. View Inventory");

System.***out***.println("2. Purchase a Product");

System.***out***.println("3. Cancel a Purchase");

System.***out***.println("4. View Shopping Cart");

System.***out***.println("5. Empty Shopping Cart");

System.***out***.println("0. Exit");

System.***out***.print("Enter the action number: ");

action = scanner.nextInt();

**switch** (action) {

**case** 1:

storeFront.displayInventory();

**break**;

**case** 2:

System.***out***.print("Enter the index of the product to purchase: ");

**int** purchaseIndex = scanner.nextInt();

storeFront.purchaseProduct(purchaseIndex);

**break**;

**case** 3:

System.***out***.print("Enter the index of the product to cancel the purchase: ");

**int** cancelIndex = scanner.nextInt();

storeFront.cancelPurchase(cancelIndex);

**break**;

**case** 4:

storeFront.displayShoppingCart();

**break**;

**case** 5:

storeFront.emptyShoppingCart();

**break**;

**case** 0:

**break**;

**default**:

System.***out***.println("Invalid action");

}

} **while** (action != 0);

System.***out***.println("Thank you for using the Store Front Application!");

scanner.close();

}

}

**package** inventoryManagerandShoppingCart;

/\*\*

\* The Weapon class represents a weapon item that can be sold.

\* It extends the SalableProducts class.

\*/

**public** **class** Weapon **extends** SalableProducts {

**private** **int** durability;

/\*\*

\* Constructs a Weapon object with the specified name, description, price, quantity, and durability.

\*

\* **@param** name the name of the weapon item

\* **@param** description the description of the weapon item

\* **@param** price the price of the weapon item

\* **@param** quantity the quantity of the weapon item

\* **@param** durability the durability of the weapon item

\*/

**public** Weapon(String name, String description, **double** price, **int** quantity, **int** durability) {

**super**(name, description, price, quantity);

**this**.durability = durability;

}

/\*\*

\* Gets the durability of the weapon item.

\*

\* **@return** the durability of the weapon

\*/

**public** **int** getDurability() {

**return** durability;

}

/\*\*

\* Sets the durability of the weapon item.

\*

\* **@param** durability the durability of the weapon

\*/

**public** **void** setDurability(**int** durability) {

**this**.durability = durability;

}

}

**package** inventoryManagerandShoppingCart;

/\*\*

\* The Armor class represents an armor item that can be sold.

\* It extends the SalableProducts class.

\*/

**public** **class** Armor **extends** SalableProducts {

**private** **int** durability;

/\*\*

\* Constructs an Armor object with the specified name, description, price, quantity, and durability.

\*

\* **@param** name the name of the armor item

\* **@param** description the description of the armor item

\* **@param** price the price of the armor item

\* **@param** quantity the quantity of the armor item

\* **@param** durability the durability of the armor item

\*/

**public** Armor(String name, String description, **double** price, **int** quantity, **int** durability) {

**super**(name, description, price, quantity);

**this**.durability = durability;

}

/\*\*

\* Gets the durability of the armor item.

\*

\* **@return** the durability of the armor

\*/

**public** **int** getDurability() {

**return** durability;

}

/\*\*

\* Sets the durability of the armor item.

\*

\* **@param** durability the durability of the armor

\*/

**public** **void** setDurability(**int** durability) {

**this**.durability = durability;

}

}

**package** inventoryManagerandShoppingCart;

/\*\*

\* The Health class represents a health item that can be sold.

\*

\* It extends the SalableProducts class.

\*/

**public** **class** Health **extends** SalableProducts {

/\*\*

\* Constructs a Health object with the specified name, description, price, and quantity.

\*

\* **@param** name the name of the health item

\*

\* **@param** description the description of the health item

\*

\* **@param** price the price of the health item

\*

\* **@param** quantity the quantity of the health item

\*/

**public** Health(String name, String description, **double** price, **int** quantity) {

**super**(name, description, price, quantity);

}

}

package inventoryManagerandShoppingCart;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

/\*\*

\* The InventoryManager class represents a manager for the store's inventory of salable products.

\*/

public class InventoryManager {

private List<SalableProducts> inventory;

/\*\*

\* Constructs an InventoryManager object and initializes the inventory.

\*/

public InventoryManager() {

inventory = new ArrayList<>();

initializeInventory();

}

/\*\*

\* Initializes the inventory with some default products.

\*/

private void initializeInventory() {

Weapon weapon1 = new Weapon("Sword", "Golden Sword", 2506.50, 10, (int) 1.9);

Weapon weapon2 = new Weapon("Mace", "Mace of Frost", 1004.20, 12, (int) 15.8);

Armor armor1 = new Armor("Helmet", "Helmet of Wisdom", 16272.89, 3, 0);

Armor armor2 = new Armor("Body Armor", "Armor of Deceit", 25672.12, 7, 0);

Health health1 = new Health("Large Health", "Full Health", 12500.00, 4);

Health health2 = new Health("Medium Health", "Half Health", 107500.00, 17);

Health health3 = new Health("Small Health", "Quarter Health", 1500.00, 25);

inventory.add(weapon1);

inventory.add(weapon2);

inventory.add(armor1);

inventory.add(armor2);

inventory.add(health1);

inventory.add(health2);

inventory.add(health3);

}

/\*\*

\* Removes a product from the inventory at the specified index.

\*

\* @param index the index of the product to remove

\*/

public void removeProduct(int index) {

if (index >= 0 && index < inventory.size()) {

SalableProducts product = inventory.get(index);

int quantity = product.getQuantity();

if (quantity > 0) {

product.setQuantity(quantity - 1);

System.out.println("Product removed: " + product.getName());

} else {

System.out.println("Product out of stock: " + product.getName());

}

} else {

System.out.println("Invalid product index");

}

}

/\*\*

\* Adds a product to the inventory at the specified index.

\*

\* @param index the index of the product to add

\*/

public void addProduct(int index) {

if (index >= 0 && index < inventory.size()) {

SalableProducts product = inventory.get(index);

int quantity = product.getQuantity();

product.setQuantity(quantity + 1);

System.out.println("Product added: " + product.getName());

} else {

System.out.println("Invalid product index");

}

}

/\*\*

\* Retrieves the inventory of salable products.

\*

\* @return the list of products in the inventory

\*/

public List<SalableProducts> getInventory() {

return inventory;

}

/\*\*

\* Adds a product to the inventory.

\*

\* @param product the product to add

\*/

public void addProduct(SalableProducts product) {

inventory.add(product);

}

}

package inventoryManagerandShoppingCart;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* The ShoppingCart class represents a shopping cart that can hold salable products.

\*/

public class ShoppingCart {

private List<SalableProducts> contents;

/\*\*

\* Constructs an empty shopping cart.

\*/

public ShoppingCart() {

contents = new ArrayList<>();

}

/\*\*

\* Adds a product to the shopping cart.

\*

\* @param product the product to be added

\*/

public void addProduct(SalableProducts product) {

contents.add(product);

System.out.println("Product added to the shopping cart: " + product.getName());

}

/\*\*

\* Removes a product from the shopping cart.

\*

\* @param product the product to be removed

\*/

public void removeProduct(SalableProducts product) {

boolean removed = contents.remove(product);

if (removed) {

System.out.println("Product removed from the shopping cart: " + product.getName());

} else {

System.out.println("Product not found in the shopping cart: " + product.getName());

}

}

/\*\*

\* Constructs an empty shopping cart.

\*/

public void emptyCart() {

contents.clear();

System.out.println("The shopping cart is emptied.");

}

/\*\*

\* Retrieves the contents of the shopping cart.

\*

\* @return the list of products in the shopping cart

\*/

public List<SalableProducts> getContents() {

return contents;

}

}

**package** inventoryManagerandShoppingCart;

**import** java.util.Scanner;

/\*\*

\* The StoreFrontApplication class represents an application for managing a store's inventory and shopping cart.

\*/

**public** **class** StoreFrontApplication {

**private** InventoryManager inventoryManager;

**private** ShoppingCart shoppingCart;

/\*\*

\* Constructs a StoreFrontApplication object, initializing the inventory manager and shopping cart.

\*/

**public** StoreFrontApplication() {

inventoryManager = **new** InventoryManager();

shoppingCart = **new** ShoppingCart();

}

/\*\*

\* Displays the inventory with product details.

\*/

**public** **void** displayInventory() {

System.***out***.println("Inventory:");

**int** index = 0;

**for** (SalableProducts product : inventoryManager.getInventory()) {

System.***out***.println("[" + index + "] " + product.getName() + " - " + product.getDescription()

+ " - Price: $" + product.getPrice() + " - Quantity: " + product.getQuantity());

index++;

}

}

/\*\*

\* Purchases a product from the inventory and adds it to the shopping cart.

\*

\* **@param** index the index of the product to purchase

\*/

**public** **void** purchaseProduct(**int** index) {

**if** (index >= 0 && index < inventoryManager.getInventory().size()) {

SalableProducts product = inventoryManager.getInventory().get(index);

**if** (product.getQuantity() > 0) {

inventoryManager.removeProduct(index);

shoppingCart.addProduct(product);

} **else** {

System.***out***.println("Product is out of stock: " + product.getName());

}

} **else** {

System.***out***.println("Invalid product index");

}

}

/\*\*

\* Cancels a purchase by removing a product from the shopping cart and adding it back to the inventory.

\*

\* **@param** index the index of the product to cancel the purchase

\*/

**public** **void** cancelPurchase(**int** index) {

**if** (index >= 0 && index < shoppingCart.getContents().size()) {

SalableProducts product = shoppingCart.getContents().get(index);

inventoryManager.addProduct(product);

shoppingCart.removeProduct(product);

} **else** {

System.***out***.println("Invalid product index");

}

}

/\*\*

\* Displays the contents of the shopping cart.

\*/

**public** **void** displayShoppingCart() {

System.***out***.println("Shopping Cart:");

**int** index = 0;

**for** (SalableProducts product : shoppingCart.getContents()) {

System.***out***.println("[" + index + "] " + product.getName() + " - " + product.getDescription()

+ " - Price: $" + product.getPrice() + " - Quantity: " + product.getQuantity());

index++;

}

}

/\*\*

\* Empties the shopping cart.

\*/

**public** **void** emptyShoppingCart() {

shoppingCart.emptyCart();

}

/\*\*

\* The main entry point of the StoreFrontApplication program.

\*

\* **@param** args command line arguments

\*/

**public** **static** **void** main(String[] args) {

StoreFrontApplication storeFront = **new** StoreFrontApplication();

System.***out***.println("Welcome to the Store Front!");

storeFront.displayInventory();

Scanner scanner = **new** Scanner(System.***in***);

**int** action;

**do** {

System.***out***.println("\nActions:");

System.***out***.println("1. View Inventory");

System.***out***.println("2. Purchase a Product");

System.***out***.println("3. Cancel a Purchase");

System.***out***.println("4. View Shopping Cart");

System.***out***.println("5. Empty Shopping Cart");

System.***out***.println("0. Exit");

System.***out***.print("Enter the action number: ");

action = scanner.nextInt();

**switch** (action) {

**case** 1:

storeFront.displayInventory();

**break**;

**case** 2:

System.***out***.print("Enter the index of the product to purchase: ");

**int** purchaseIndex = scanner.nextInt();

storeFront.purchaseProduct(purchaseIndex);

**break**;

**case** 3:

System.***out***.print("Enter the index of the product to cancel the purchase: ");

**int** cancelIndex = scanner.nextInt();

storeFront.cancelPurchase(cancelIndex);

**break**;

**case** 4:

storeFront.displayShoppingCart();

**break**;

**case** 5:

storeFront.emptyShoppingCart();

**break**;

**case** 0:

**break**;

**default**:

System.***out***.println("Invalid action");

}

} **while** (action != 0);

System.***out***.println("Thank you for using the Store Front Application!");

scanner.close();

}

}