Assignment 1 - Java

ESS 201 Programming II

International Institute of Information Technology – Bangalore

(Submission: Domjudge by 6 Nov 23:59:59)

Problem Statement:

You have just joined an e-commerce startup, "ShopEase," as a junior developer. The company sells a variety of products, ranging from electronics to fashion accessories, and they aim to deliver a seamless and secure shopping experience for customers. The CEO wants to keep customers purchase data secure while allowing authorized access to important information, like the purchase history, current cart status, and order details. Your job is to create a Customer Management Module using the principles of encapsulation in Object-Oriented Programming (OOP). This module should ensure that customer data remains private, with controlled access through well-defined methods.

In this assignment, you'll design a program that models a Customer and their shopping cart using encapsulation, with the aim of demonstrating how private data can be managed safely in an e-commerce application. Create a Java program for an e-commerce platform that manages products. Each product needs a unique ID, and this ID should be generated automatically when a new product is created. You will implement this functionality using constructors and constructor overloading.

Requirements:

- **Customer Class**:
    - **Encapsulation**: Use private fields for all attributes.
    - **Constructor**: Initialize customer details (customerId, name, email, address) and instantiate an empty ShoppingCart.
    - **Methods**:
        - getName(): Returns the name of the customer.
        - getAddress(): Returns the customer’s address.
        - viewCustomerInfo(): Displays the customer’s name and address.
        - A reference to the customer’s ShoppingCart.

- **ShoppingCart Class**:

- **Encapsulation**: Use private fields to store a list of Item objects.
- **Methods**:
  - addItem(Item item): Adds a new item to the cart.
  - removeItem(String itemId): Removes an item by its ID.
  - viewCart(): Prints all items in the cart.
  - calculateTotal(): Returns the total price of items in the cart, calculated as price * quantity for each item.

- **Item Class**:
  - **Encapsulation**: Use private fields for item details (itemId, name, price, quantity).
  - **Constructor**: Initialize item details based on parameters.
  - **Methods**:
    - toString(): Format item details as itemId name price quantity.

Updated Requirements:

- **Helper Class**:

  The Helper class is a utility class for handling all input-output operations and command processing in the e-commerce application.

  Purpose:

  - Manages input-output operations, printing messages to a specified PrintStream.
  - Processes commands by interacting with customers and inventory data structures, managing shopping carts, and displaying relevant information.
  - Parses commands, handling quoted strings for parameters that may contain spaces (e.g., customer names and addresses).

  Attributes:

  - out (PrintStream): The output stream for printing messages (e.g., System.out for console output).

  Constructor:

- Helper(PrintStream out): Initializes the Helper instance with the specified output stream for printing.

Methods:

- print(String message):
    - Purpose: Prints a message to the output stream.
    - Parameters:
        - message: The message to be printed.
- displayCart(String customerId, ShoppingCart cart):
    - Purpose: Displays all items in a customer's shopping cart.
    - Parameters:
        - customerId: The unique ID of the customer.
        - cart: The ShoppingCart object containing items added by the customer.
- displayTotalPrice(String customerId, double total):
    - Purpose: Displays the total price of items in a customer's shopping cart.
    - Parameters:
        - customerId: The unique ID of the customer.
        - total: The calculated total price of items in the cart.
- displayCustomerInfo(Customer customer):
    - Purpose: Displays a customer's name and address.
    - Parameters:
        - customer: The Customer object for which information is to be displayed.
- processCommands(Scanner scanner, HashMap<String, Customer> customers, HashMap<String, Item> inventory):
    - Purpose: Processes user commands, interacting with customers and inventory to add items, manage shopping carts, and display information.
    - Parameters:
        - scanner: Input stream for reading commands.

        - customers: Map of all customers by their unique IDs.
        - inventory: Map of all items in inventory by their unique IDs.
    - Behavior: Parses commands, performs actions (e.g., adding items to inventory, adding items to cart, viewing cart, calculating total price, viewing customer details), and handles output through the out stream.
- parseCommand(String command):
    - Purpose: Parses a command string to split tokens, correctly handling parameters with spaces that are enclosed in double quotes (e.g., customer names and addresses).
    - Parameters:
        - command: The input command as a single line of text.
    - Returns: An array of String tokens, with quoted sections treated as single tokens even if they contain spaces.
    - Behavior:
        - Uses a regular expression to split the input string into tokens. Recognizes quoted strings as single tokens, even if they contain spaces, and non-quoted words are treated as separate tokens.

Note: Helper class code is provided and you are not allowed to change it.

- **Main Class**:

  Purpose:
  - To initialize the application components, including customers and items.
  - To handle the command-line interface, allowing users to add customers, manage a shopping cart, and display customer or cart details.

  Key Responsibilities:
  - Creates instances of Customer and Item classes based on input commands.
  - Uses the Helper class to handle input-output operations.

  Attributes:

- customers (HashMap<String, Customer>): Stores all registered customers with unique customer IDs.
- inventory (HashMap<String, Item>): Stores all available items in the inventory with unique item IDs.

Methods:

- main(String[] args):
    - Initializes the Helper instance with System.out.
    - Initializes customers and inventory data structures.
    - Passes the Scanner input stream and data structures to Helper.processCommands() for command processing.

Note: Main class code is provided and you are not allowed to change it.

- **Input Format**
    - ADD_ITEM itemId name price quantity

      Adds an item to the system's inventory with the following:
        - itemId: Unique string ID for the item (e.g., "101").
        - name: Name of the item (a string with no spaces; e.g., "Mobile").
        - price: Double value representing the price of the item (e.g., 15000).
        - quantity: Integer value for the available quantity of the item.
    - ADD_CUSTOMER customerId name email address

      Adds a customer to the system with the following:
        - customerId: Unique string ID for the customer (e.g., "C001").
        - name: Customer's name (a single string, e.g., "RahulVerma").
        - email: Customer's email ID (valid email format).
        - address: Address string without spaces (e.g., "12MG_Road_Delhi").
    - ADD_TO_CART customerId itemId
        - Adds a specified item from the inventory to a customer's shopping cart:
        - customerId: Customer ID (should match an existing customer).
        - itemId: Item ID (should match an existing item).

- VIEW_CART customerId
  - Displays all items currently in the specified customer's cart.
- TOTAL_PRICE customerId
  - Shows the total cost of all items in the specified customer's cart.
- VIEW_CUSTOMER customerId
  - Displays the customer's information, including their name and address.
- EXIT
  - Exits the program.

- **Output Format**
  - The program outputs relevant information for each command:
    - ADD_ITEM: Confirms that the item has been added to inventory.
    - ADD_CUSTOMER: Confirms that the customer has been added to the system.
    - ADD_TO_CART: Confirms that the item has been added to the customer's cart.
    - VIEW_CART: Lists each item in the cart, showing the itemId, name, price, and quantity.
    - TOTAL_PRICE: Displays the total price of items in the customer's cart.
    - VIEW_CUSTOMER: Displays the customer's name and address.
- **Constraints**
  - Unique Identifiers:
    - Each itemId and customerId must be unique within their respective lists.
    - Attempting to add an item or customer with a duplicate ID should not be allowed.
  - Valid Entries:
    - Item price must be non-negative (e.g., price >= 0).
    - Item quantity must be a positive integer (e.g., quantity > 0).
    - Email format should be valid (e.g., contain "@" and a domain).
  - Inventory Limits:

        - Adding to the cart should check that the item exists in the inventory.
        - Adding an item to a cart should decrease its available inventory by the quantity added.
        - Attempting to add an out-of-stock item should be handled gracefully.
    - Operations on Non-existent IDs:
        - Commands like ADD_TO_CART, VIEW_CART, TOTAL_PRICE, and VIEW_CUSTOMER should handle cases where the specified customerId or itemId does not exist and output an appropriate error message.
    - Cart Management:
        - Items in the cart should display correctly with their current quantity.
        - Each TOTAL_PRICE command should recalculate based on the items' latest prices and quantities in the customer's cart.
    - Command Order:
        - ADD_ITEM should be executed before adding the item to any cart.
        - ADD_CUSTOMER should precede any operation involving that customer.

- **Sample Input**

ADD_ITEM 101 Mobile 15000 1
ADD_ITEM 102 Earphones 500 3
ADD_CUSTOMER C001 RahulVerma rahul.verma@gmail.com 12MG_Road_Delhi
ADD_CUSTOMER C002 PriyaSharma priya.sharma@yahoo.in 45SP_Marg_Mumbai
ADD_TO_CART C001 101
ADD_TO_CART C001 102
VIEW_CART C001
TOTAL_PRICE C001
VIEW_CUSTOMER C002
EXIT

- **Sample Output:**

Item added to inventory: 101 Mobile 15000.0 1
Item added to inventory: 102 Earphones 500.0 3

Customer added: C001 "RahulVerma" "rahul.verma@gmail.com" "12MG_Road_Delhi"

Customer added: C002 "PriyaSharma" "priya.sharma@yahoo.in" "45SP_Marg_Mumbai"

Item 101 added to C001's cart

Item 102 added to C001's cart

Items in Cart for C001:

101 Mobile 15000.0 1

102 Earphones 500.0 3

Total Price for C001: 16500.0

Customer Info for C002:

Name: PriyaSharma

Address: 45SP_Marg_Mumbai