**ASSIGNMENT 03**

**KAIF IKHLAQ**

**BSAIM-S24-027**

**Demo Video Link:**

**https://drive.google.com/file/d/1-5IDCFx5I3py0sw78iqGCakvHhxJRiyU/view?usp=drivesdk**

**Project Overview:**

This project implements a basic shopping cart system where users can browse products, add or remove them from their cart, and checkout with different payment methods (credit or debit). The system involves several key concepts, such as **encapsulation**, **inheritance**, **abstraction**, and **file handling**. Let's break down the functionality of the project:

**1. Item Class and Product Inheritance:**

The Item class represents an individual product with properties such as name, price, and stock. The Product class inherits from the Item class, allowing it to access the attributes and methods of the Item class. This follows the **inheritance** principle, enabling easy extension of product types without duplicating code. Methods like get\_name(), get\_price(), and set\_stock() encapsulate the attributes, ensuring that stock is only modified safely.

**2. Payment Processing via Abstraction:**

The PaymentProcessor class is an abstract class that defines a common interface for processing payments. Two concrete classes, CreditCardPayment and DebitCardPayment, inherit from PaymentProcessor and implement the process\_payment() method, each handling payment in a different way. This demonstrates **abstraction** by providing a common interface and allowing for specific implementations without affecting the rest of the system.

**3. Cart Functionality:**

The Cart class manages the user's shopping cart, allowing products to be added, removed, and displaying the total price. It ensures the stock of each product is updated correctly when added or removed, maintaining integrity. The cart also supports saving and loading its state from a file, making it persistent across sessions. This is done using the save\_cart() and load\_cart() methods. These features ensure that the cart can be saved to a file for later use, or loaded with products from a saved file.

**4. User Interaction:**

The User class handles the user's interaction with the shopping system. Users can add items to their cart, remove them, and proceed to checkout. The checkout process calculates the total price of the cart and asks the user to choose a payment method. Once payment is successful, the cart is cleared, and the cart’s state is saved to a file. The user can also load a previous cart from a file. This class facilitates the overall flow of the shopping session, ensuring a smooth user experience.

**5. Product Management (File Handling):**

Products are either loaded from an external file (products.txt) or a default set is used. The save\_products() function writes the product data back to the file whenever changes are made. This ensures that the product information persists across sessions and can be updated easily. The system allows dynamic product management through the file, ensuring that product data is consistently available for users.

**6. Shopping Session:**

The shopping\_session() function manages the interaction with the user. It prompts the user to choose actions such as adding or removing products, checking out, saving or loading the cart, or exiting the session. Based on user input, the appropriate actions are taken, and feedback is provided. For example, when a product is added to the cart, it’s reflected in the system, and when a checkout occurs, the user is prompted to pay. This ensures that the system reacts dynamically to user decisions.

**7. File Handling for Cart and Products:**

File handling is a key feature of this project, enabling cart data and product information to be saved and loaded from external files. This allows users to continue shopping across sessions without losing their data. For example, a user’s cart is saved to a file after checkout and can be reloaded in future sessions. Similarly, product data is saved to a file (products.txt), ensuring the product inventory is always up-to-date.

**Conclusion:**

In summary, this project combines essential object-oriented programming principles such as inheritance, abstraction, and encapsulation to create a functional shopping cart system. The use of file handling allows data persistence, making the shopping experience more flexible and user-friendly. Users can manage their cart, choose payment methods, and continue their shopping across multiple sessions, while the system ensures product availability and correct inventory management.