ÇANKAYA UNIVERSITY
SOFTWARE ENGINEERING DEPARTMENT
SOFTWARE PROJECT I

Name Surname	**Duru KARACAN**
Identity Number	**202128022**
Course	**SENG 271**
Experiment	**Experiment 2**
E-mail	**c2128022@student.cankaya.edu.tr**

PROBLEM STATEMENT

The Goals of the Programming Assignment:

The primary goals of the programming assignment for the MyBazaar Online Shopping Application are to create a sophisticated, user-friendly platform that caters to both service providers (administrative and technical staff) and clients (customers). The key objectives include:

1. User Management:

- Implement a system for users to view essential personal information.

- Enable customers to change passwords, update account balances, and access real-time information about ongoing campaigns.

- Implement shopping cart operations with proper validation checks.

- Automatically update customer status based on spending, assigning CLASSIC, SILVER, or GOLDEN status with associated discounts.

- Override the toString() method in the Customer class for a comprehensive display of customer data.

2. Employee Functionality:

- Provide employees with capabilities like displaying stock amounts, listing available item types, and identifying VIP customers.

- Allow admins to add new customers, admins, and technicians dynamically.

- Implement a personalized data display method for admins.

- Empower admins to launch and manage campaigns dynamically.

3.Technician Responsibilities:

- Facilitate senior technicians in viewing detailed information about customer orders.

- Allow technicians to display comprehensive information about specific items and add new items dynamically.

4. Item Categories and Attributes:

- Define and categorize items into Cosmetic, Electronic, and Office Supplies with specific attributes.

- Assign unique attributes for items in each category.

- Initialize item stock values based on the input file named item.txt.

5. Shopping and Orders:

- Display order information efficiently, including order date, customerID, total cost, and the number of bought items.

- Implement campaigns with start date, end date, item type, and a discount rate.

What are the normal inputs to the program?

The normal inputs to the MyBazaar Online Shopping Application are provided through a **combination of user interactions and system commands.**

Each user interaction and system command has its own unique format.

User Interactions:

1. **Customers:**
    - View personal data
    - Change password
    - Update balance
    - View active campaigns
    - Perform shopping cart operations (add items, clear cart, place orders)
    - Receive post-order updates on customer status
2. **Employees:**
    - Display stock amount
    - List item types
    - Identify VIP customers
3. **Admins:**
    - Add new customers
    - Add admins and technicians
    - Display customer data
    - Display all customers
    - Override display personal data
    - Launch campaigns
4. **Technicians:**
    - Display all orders (senior technicians only)
    - Display item information
    - Add new item

System Commands:

1. **Customer Actions:**
    - Add a new customer
    - Show customer info
    - List customers
    - Launch campaigns
2. **Employee Actions:**
    - Add admins and technicians
    - Display item information
    - List existing items
    - Show items with low stock
    - Show VIP customers
    - Display items under specific types
3. **Shopping Actions:**
    - Add items to the cart
    - Place orders
    - Display campaigns
    - Empty the cart
4. **Item Management:**
    - Add new items
5. **Password Actions:**
    - Change passwords for customers
6. **Financial Transactions:**
    - Deposit money into a customer's account

What output should the program create?

The program should generate appropriate output based on the executed commands.

1**. Add a new Customer:**

- Output: Customer added successfully or an error message if unsuccessful.

2. Show customer info:

- Output: Comprehensive information about the specified customer or an error message if the customer doesn't exist.

3. List Customers:

- Output: A list of all customers' basic information.

4. Show admin information:

- Output: Detailed information about the specified admin or an error message if the admin doesn't exist.

5. Launch a campaign:

- Output: Confirmation message for the successful launch of the campaign or an error message if unsuccessful.

6. Add a new employee:

- Output: Confirmation message for the successful addition of a new admin or technician or an error message if unsuccessful.

7. List existing items:

- Output: A list of all available items with basic information.

8. Show items types with low stock:

- Output: A list of item types that have a stock level lower than the specified threshold.

9. Show VIP customers:

- Output: A list of VIP customers and their details.

10. Display items under specific types:

- Output: A list of items falling under the specified types.

11. Add a new item:

- Output: Confirmation message for the successful addition of a new item or an error message if unsuccessful.

12. Show orders:

- Output: A detailed list of orders made by customers, visible only to senior technicians.

13. Change password:

- Output: Confirmation message for the successful password change or an error message if unsuccessful.

14. Load money:

- Output: Confirmation message for the successful deposit of money into the customer's account or an error message if unsuccessful.

15. Display campaigns:

- Output: A list of ongoing campaigns for the specified customer.

16. Add to cart:

- Output: Confirmation message for the successful addition of an item to the shopping cart or an error message if unsuccessful.

17. Empty cart:

- Output: Confirmation message for the successful removal of all items from the shopping cart or an error message if unsuccessful.

18. Place a new purchase order:

- Output: Confirmation message for the successful placement of a new order or an error message if unsuccessful.

What error handling was required?

1. **User Management:**
    - Verify the presence of admins for operations related to customers.
    - Ensure email uniqueness when adding new customers.
    - Validate the accuracy of customer transactions.
2. **Campaign and Employee Management:**
    - Authenticate admin presence for campaign creation.
    - Validate campaign dates, product types, and employee details.
3. **Product Management:**
    - Validate input data for product operations.
    - Ensure consistency for various system operations.
4. **Order and Technical Staff Operations:**
    - Authenticate senior technician presence for order-related operations.
    - Check the accuracy of technical staff operations.
5. **Customer Operations:**
    - Validate input data for customer operations.
    - Confirm customer presence, correct passwords, and sufficient balance.
6. **General System Operations:**
    - Implement consistent data validation for various system operations.

DESIGN

1. **Design Decisions:**

    • Object-Oriented Approach: You’ve utilized classes and inheritance, encapsulating related properties and methods. This promotes code reusability and modularity.

    • Static Methods and Lists: Many methods are static and operate on static lists, centralizing data management.

2. **Data Structures:**

    • Lists: You’ve extensively used ArrayList to manage collections of objects like Admin, Customer, Technician, and Item. This choice is suitable for dynamic data management.

    • Class Inheritance: Multiple classes inherit properties and methods from parent classes, forming a hierarchy.

3. **Algorithms:**

    • Search Algorithms: Methods like findAdminByName and findCustomerByID iterate over lists to find specific objects. These are linear search implementations.

    • Sorting and Filtering: While not explicitly implemented, the structure suggests potential for sorting and filtering operations on lists.

4. **Pros/Cons:**

    • **Pros:**

- Modularity and Reusability: Object-oriented design allows for code reusability and easier maintenance.

- Scalability: Using lists for data management supports dynamic data operations.

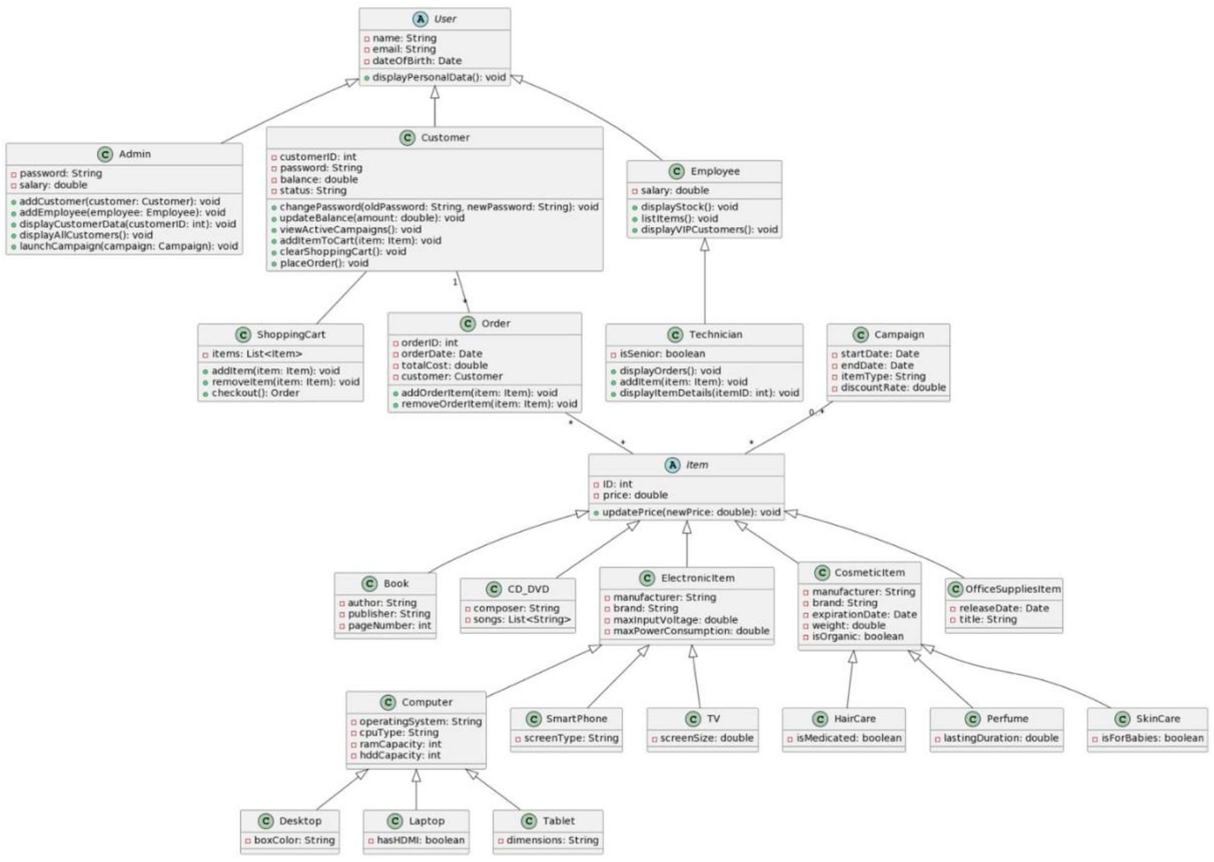
- **Cons:**

- Performance: Linear search algorithms may become inefficient with large datasets.

- Tight Coupling: Heavy reliance on static methods and data can lead to tight coupling, making changes more complex.

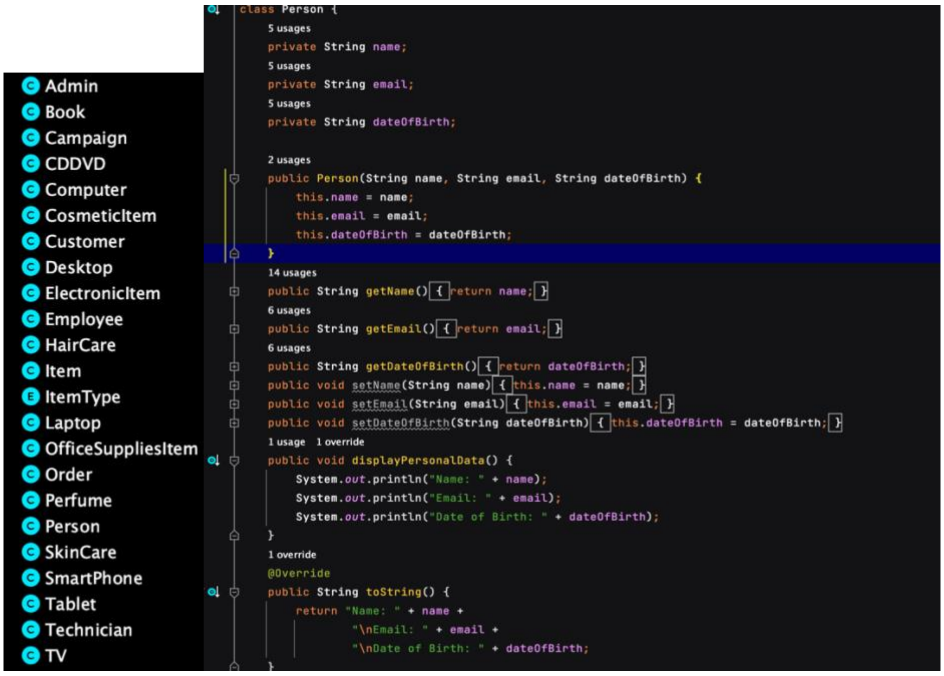
IMPLEMENTATION DETAILS

- **Class diagram**

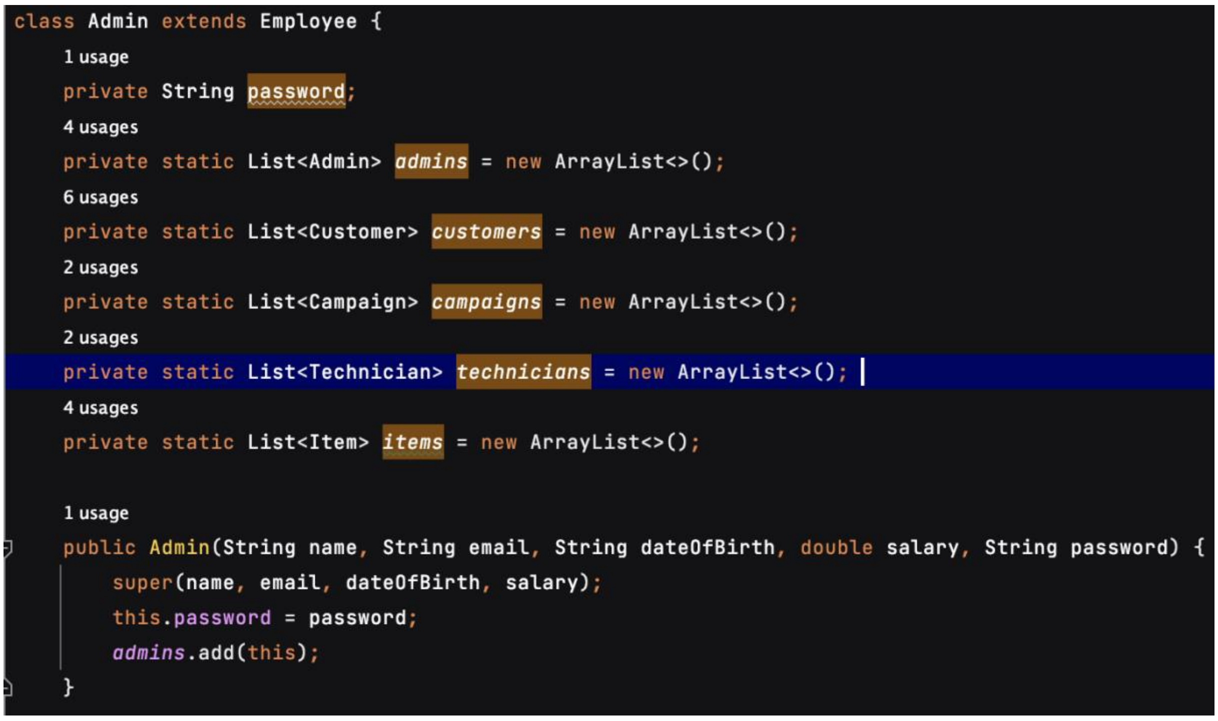


- **What sample code did you start with?**

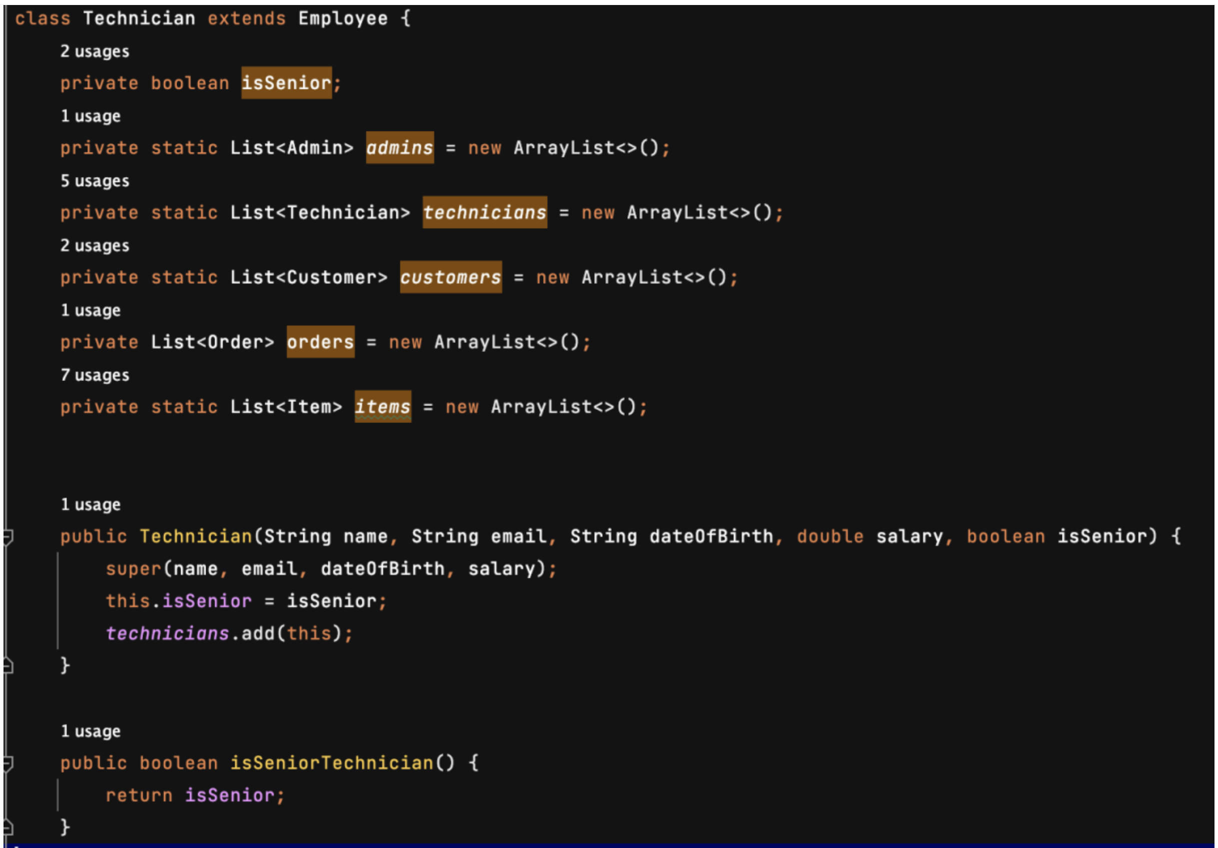
  A simple skeleton for the code I will write;



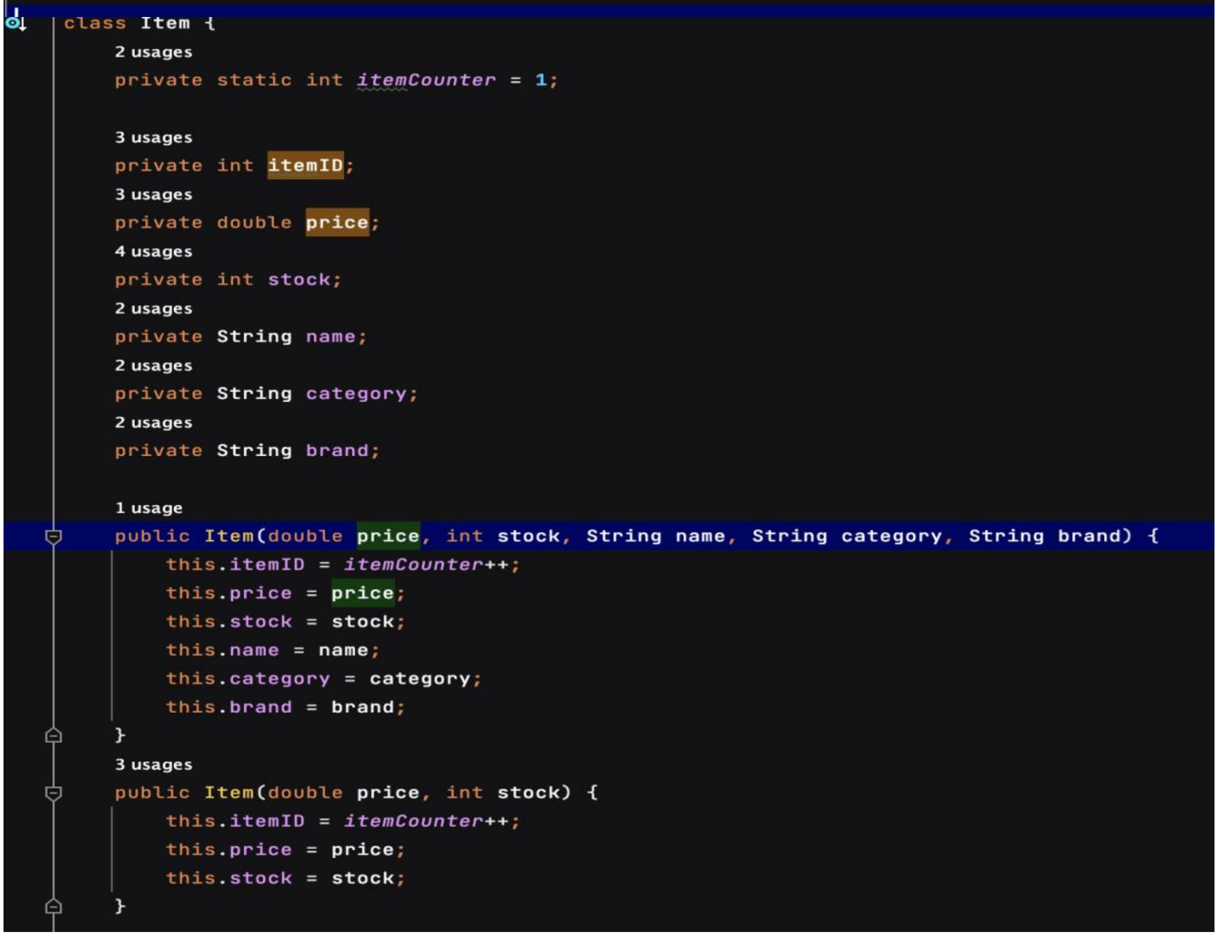
```java
class Person {
    5 usages
    private String name;
    5 usages
    private String email;
    5 usages
    private String dateOfBirth;

    2 usages
    public Person(String name, String email, String dateOfBirth) {
        this.name = name;
        this.email = email;
        this.dateOfBirth = dateOfBirth;
    }
    14 usages
    public String getName() { return name; }
    6 usages
    public String getEmail() { return email; }
    6 usages
    public String getDateOfBirth() { return dateOfBirth; }
    public void setName(String name) { this.name = name; }
    public void setEmail(String email) { this.email = email; }
    public void setDateOfBirth(String dateOfBirth) { this.dateOfBirth = dateOfBirth; }
    1 usage  1 override
    public void displayPersonalData() {
        System.out.println("Name: " + name);
        System.out.println("Email: " + email);
        System.out.println("Date of Birth: " + dateOfBirth);
    }
    1 override
    @Override
    public String toString() {
        return "Name: " + name +
                "\nEmail: " + email +
                "\nDate of Birth: " + dateOfBirth;
    }
```



```java
class Admin extends Employee {
    1 usage
    private String password;
    4 usages
    private static List<Admin> admins = new ArrayList<>();
    6 usages
    private static List<Customer> customers = new ArrayList<>();
    2 usages
    private static List<Campaign> campaigns = new ArrayList<>();
    2 usages
    private static List<Technician> technicians = new ArrayList<>();
    4 usages
    private static List<Item> items = new ArrayList<>();

    1 usage
    public Admin(String name, String email, String dateOfBirth, double salary, String password) {
        super(name, email, dateOfBirth, salary);
        this.password = password;
        admins.add(this);
    }
```



```java
class Technician extends Employee {
    2 usages
    private boolean isSenior;
    1 usage
    private static List<Admin> admins = new ArrayList<>();
    5 usages
    private static List<Technician> technicians = new ArrayList<>();
    2 usages
    private static List<Customer> customers = new ArrayList<>();
    1 usage
    private List<Order> orders = new ArrayList<>();
    7 usages
    private static List<Item> items = new ArrayList<>();


    1 usage
    public Technician(String name, String email, String dateOfBirth, double salary, boolean isSenior) {
        super(name, email, dateOfBirth, salary);
        this.isSenior = isSenior;
        technicians.add(this);
    }

    1 usage
    public boolean isSeniorTechnician() {
        return isSenior;
    }
```



```java
class Item {
    2 usages
    private static int itemCounter = 1;

    3 usages
    private int itemID;
    3 usages
    private double price;
    4 usages
    private int stock;
    2 usages
    private String name;
    2 usages
    private String category;
    2 usages
    private String brand;

    1 usage
    public Item(double price, int stock, String name, String category, String brand) {
        this.itemID = itemCounter++;
        this.price = price;
        this.stock = stock;
        this.name = name;
        this.category = category;
        this.brand = brand;
    }
    3 usages
    public Item(double price, int stock) {
        this.itemID = itemCounter++;
        this.price = price;
        this.stock = stock;
    }
```

- **How did you extend or adapt this code?**

  First, I created my classes one by one, then I determined the necessary attributes and methods. Then I started to establish the logic one by one. The thing that took the code to the next level for me was the item class. While creating the item class, I defined abstract and then determined and implemented the necessary interfaces. And I also added order and campaign classes

- **What was your development timeline?**

  Week 1: Project Initiation and Requirement Analysis
  During the first week, I initiated the project by outlining its overall goals and conducting a thorough analysis of the requirements

  Week 2: Foundation Coding
  In the second week, I commenced the coding phase by laying the foundation of the application. Key classes such as User, Item, ShoppingCart, and Order were created during this period.

  Week 3: Refinement and Expansion
  Moving into the third week, I focused on refining the existing codebase and introducing additional helper classes. The development during this week aimed to create a robust structure capable of supporting the evolving needs of the application.

TESTING NOTES

- **Describe how you tested your program.**

  I haven't tested my program yet, but I'm thinking of testing it with the sample txt files in the description document given when introducing the project.

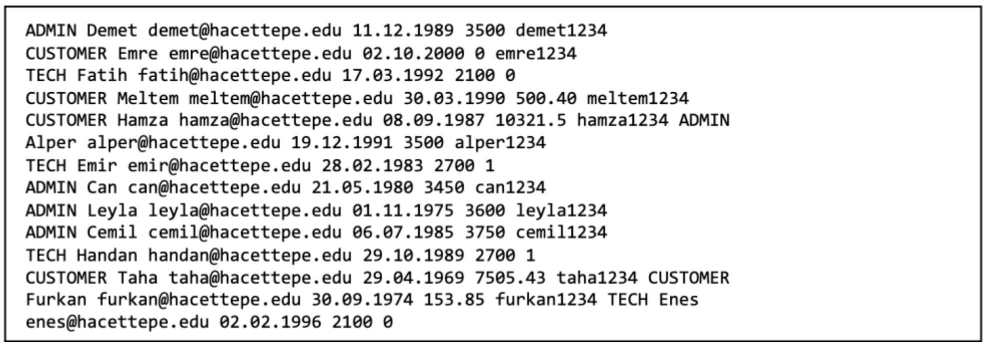
  I plan to put the project through extensive testing using a two-phase approach;

  First, unit tests can be used to verify the basic functionality of base classes such as User, Item, Cart, and Order.

  User scenarios and flow tests can then be performed to evaluate more complex functions and user interactions..

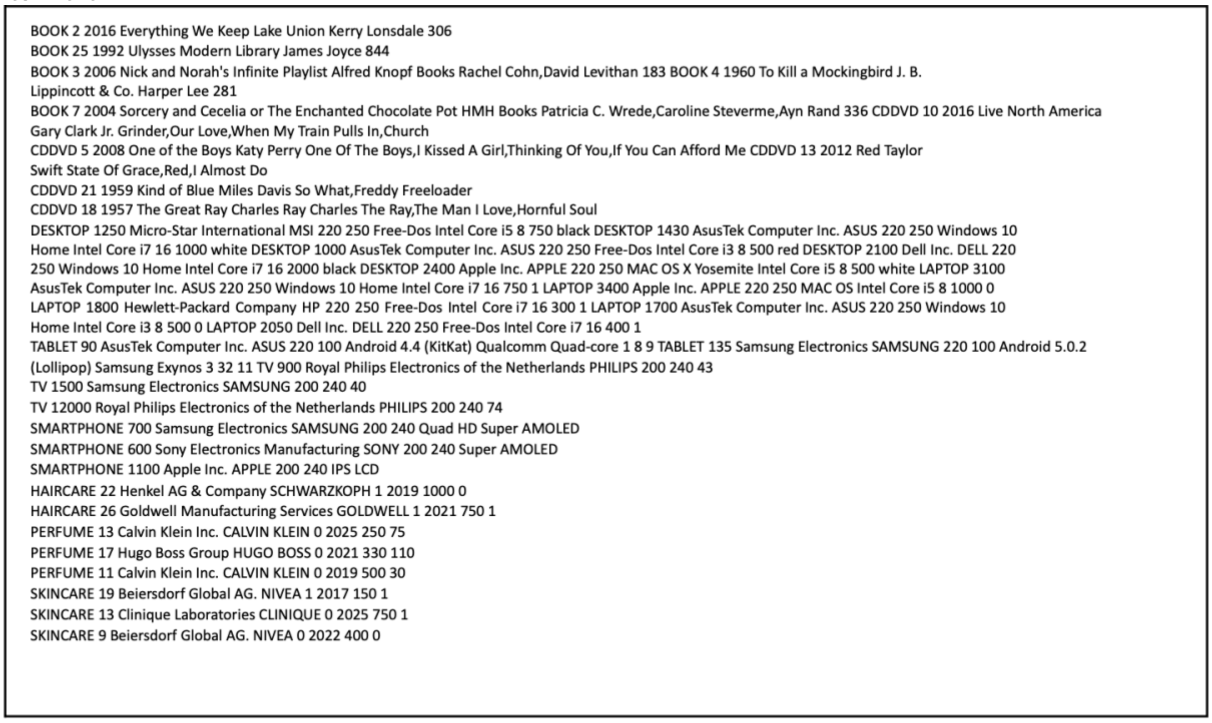
- **What were the normal inputs you used?**
  I assumed the 3 files shared with me as my basic input file.

  user.txt:



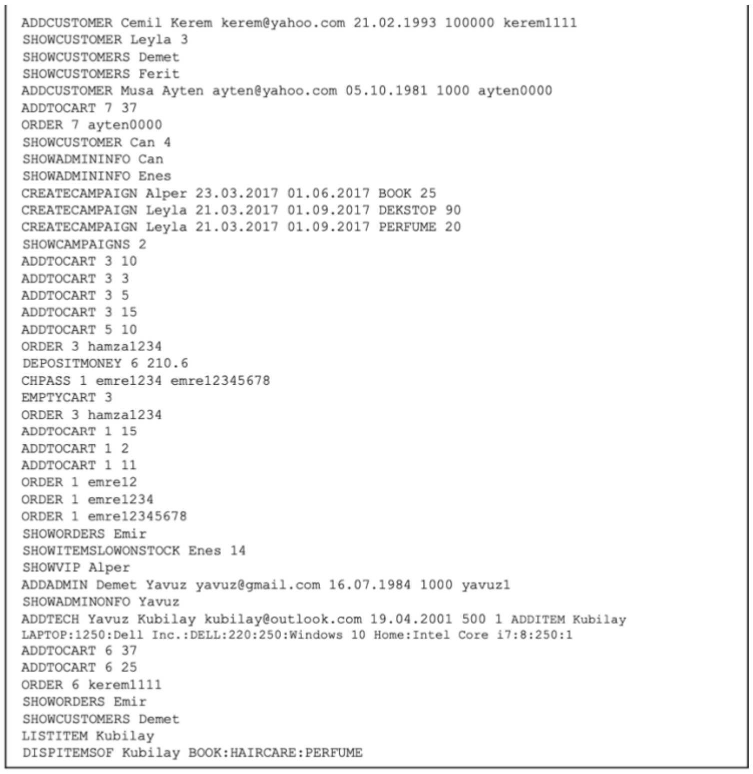
```
ADMIN Demet demet@hacettepe.edu 11.12.1989 3500 demet1234
CUSTOMER Emre emre@hacettepe.edu 02.10.2000 0 emre1234
TECH Fatih fatih@hacettepe.edu 17.03.1992 2100 0
CUSTOMER Meltem meltem@hacettepe.edu 30.03.1990 500.40 meltem1234
CUSTOMER Hamza hamza@hacettepe.edu 08.09.1987 10321.5 hamza1234 ADMIN
Alper alper@hacettepe.edu 19.12.1991 3500 alper1234
TECH Emir emir@hacettepe.edu 28.02.1983 2700 1
ADMIN Can can@hacettepe.edu 21.05.1980 3450 can1234
ADMIN Leyla leyla@hacettepe.edu 01.11.1975 3600 leyla1234
ADMIN Cemil cemil@hacettepe.edu 06.07.1985 3750 cemil1234
TECH Handan handan@hacettepe.edu 29.10.1989 2700 1
CUSTOMER Taha taha@hacettepe.edu 29.04.1969 7505.43 taha1234 CUSTOMER
Furkan furkan@hacettepe.edu 30.09.1974 153.85 furkan1234 TECH Enes
enes@hacettepe.edu 02.02.1996 2100 0
```

  item.txt:



```
BOOK 2 2016 Everything We Keep Lake Union Kerry Lonsdale 306
BOOK 25 1992 Ulysses Modern Library James Joyce 844
BOOK 3 2006 Nick and Norah's Infinite Playlist Alfred Knopf Books Rachel Cohn,David Levithan 183 BOOK 4 1960 To Kill a Mockingbird J. B.
Lippincott & Co. Harper Lee 281
BOOK 7 2004 Sorcery and Cecelia or The Enchanted Chocolate Pot HMH Books Patricia C. Wrede,Caroline Steverme,Ayn Rand 336 CDDVD 10 2016 Live North America
Gary Clark Jr. Grinder,Our Love,When My Train Pulls In,Church
CDDVD 5 2008 One of the Boys Katy Perry One Of The Boys,I Kissed A Girl,Thinking Of You,If You Can Afford Me CDDVD 13 2012 Red Taylor
Swift State Of Grace,Red,I Almost Do
CDDVD 21 1959 Kind of Blue Miles Davis So What,Freddy Freeloader
CDDVD 18 1957 The Great Ray Charles Ray Charles The Ray,The Man I Love,Hornful Soul
DESKTOP 1250 Micro-Star International MSI 220 250 Free-Dos Intel Core i5 8 750 black DESKTOP 1430 AsusTek Computer Inc. ASUS 220 250 Windows 10
Home Intel Core i7 16 1000 white DESKTOP 1000 AsusTek Computer Inc. ASUS 220 250 Free-Dos Intel Core i3 8 500 red DESKTOP 2100 Dell Inc. DELL 220
250 Windows 10 Home Intel Core i7 16 2000 black DESKTOP 2400 Apple Inc. APPLE 220 250 MAC OS X Yosemite Intel Core i5 8 500 white LAPTOP 3100
AsusTek Computer Inc. ASUS 220 250 Windows 10 Home Intel Core i7 16 750 1 LAPTOP 3400 Apple Inc. APPLE 220 250 MAC OS Intel Core i5 8 1000 0
LAPTOP 1800 Hewlett-Packard Company HP 220 250 Free-Dos Intel Core i7 16 300 1 LAPTOP 1700 AsusTek Computer Inc. ASUS 220 250 Windows 10
Home Intel Core i3 8 500 0 LAPTOP 2050 Dell Inc. DELL 220 250 Free-Dos Intel Core i7 16 400 1
TABLET 90 AsusTek Computer Inc. ASUS 220 100 Android 4.4 (KitKat) Qualcomm Quad-core 1 8 9 TABLET 135 Samsung Electronics SAMSUNG 220 100 Android 5.0.2
(Lollipop) Samsung Exynos 3 32 11 TV 900 Royal Philips Electronics of the Netherlands PHILIPS 200 240 43
TV 1500 Samsung Electronics SAMSUNG 200 240 40
TV 12000 Royal Philips Electronics of the Netherlands PHILIPS 200 240 74
SMARTPHONE 700 Samsung Electronics SAMSUNG 200 240 Quad HD Super AMOLED
SMARTPHONE 600 Sony Electronics Manufacturing SONY 200 240 Super AMOLED
SMARTPHONE 1100 Apple Inc. APPLE 200 240 IPS LCD
HAIRCARE 22 Henkel AG & Company SCHWARZKOPH 1 2019 1000 0
HAIRCARE 26 Goldwell Manufacturing Services GOLDWELL 1 2021 750 1
PERFUME 13 Calvin Klein Inc. CALVIN KLEIN 0 2025 250 75
PERFUME 17 Hugo Boss Group HUGO BOSS 0 2021 330 110
PERFUME 11 Calvin Klein Inc. CALVIN KLEIN 0 2019 500 30
SKINCARE 19 Beiersdorf Global AG. NIVEA 1 2017 150 1
SKINCARE 13 Clinique Laboratories CLINIQUE 0 2025 750 1
SKINCARE 9 Beiersdorf Global AG. NIVEA 0 2022 400 0
```

commands.txt:



```
ADDCUSTOMER Cemil Kerem kerem@yahoo.com 21.02.1993 100000 kerem1111
SHOWCUSTOMER Leyla 3
SHOWCUSTOMERS Demet
SHOWCUSTOMERS Ferit
ADDCUSTOMER Musa Ayten ayten@yahoo.com 05.10.1981 1000 ayten0000
ADDTOCART 7 37
ORDER 7 ayten0000
SHOWCUSTOMER Can 4
SHOWADMININFO Can
SHOWADMININFO Enes
CREATECAMPAIGN Alper 23.03.2017 01.06.2017 BOOK 25
CREATECAMPAIGN Leyla 21.03.2017 01.09.2017 DEKSTOP 90
CREATECAMPAIGN Leyla 21.03.2017 01.09.2017 PERFUME 20
SHOWCAMPAIGNS 2
ADDTOCART 3 10
ADDTOCART 3 3
ADDTOCART 3 5
ADDTOCART 3 15
ADDTOCART 5 10
ORDER 3 hamza1234
DEPOSITMONEY 6 210.6
CHPASS 1 emre1234 emre12345678
EMPTYCART 3
ORDER 3 hamza1234
ADDTOCART 1 15
ADDTOCART 1 2
ADDTOCART 1 11
ORDER 1 emre12
ORDER 1 emre1234
ORDER 1 emre12345678
SHOWORDERS Emir
SHOWITEMSLOWONSTOCK Enes 14
SHOWVIP Alper
ADDADMIN Demet Yavuz yavuz@gmail.com 16.07.1984 1000 yavuz1
SHOWADMINONFO Yavuz
ADDTECH Yavuz Kubilay kubilay@outlook.com 19.04.2001 500 1 ADDITEM Kubilay
LAPTOP:1250:Dell Inc.:DELL:220:250:Windows 10 Home:Intel Core i7:8:250:1
ADDTOCART 6 37
ADDTOCART 6 25
ORDER 6 kerem1111
SHOWORDERS Emir
SHOWCUSTOMERS Demet
LISTITEM Kubilay
DISPITEMSOF Kubilay BOOK:HAIRCARE:PERFUME
```

- **What were the special cases you tested?**
  Invalid Commands in commands.txt:
  Scenario: Add unknown commands to commands.txt.
  Expected: System gracefully handles unknown commands, logs errors.

  Missing info in user.txt:
  Scenario: Remove essential user info in user.txt.
  Expected: System handles missing/invalid info, logs errors.

  Invalid Item Info in items.txt:
  Scenario: Modify essential item info in items.txt.
  Expected: System handles missing/invalid info, logs errors.

- **Did everything work as expected?**

  The testing process revealed notable issues with the output generation. In several instances, the system failed to produce the expected outputs accurately, particularly in terms of data representation and operation results. Additional testing and code corrections are necessary to address these issues effectively.

- Include sample input/output from your program.