Alexandria University
Faculty of Engineering
Computer and Communication Engineering
Specialized Scientific Programs

CC272
Programming II
Fall 2025/2026

Inventory management system

Part one (worth 75% of the grade)

You are required to develop a Java-based inventory management system for a retail store using Object-oriented programming concepts. The system should have two user types: Admins and Employees.
The admin is responsible for adding new employees to the system and removing the data of the employees that no longer work for the retail store.
To model the admin and his responsibilities in our system, we need to define a few classes.

First Class:
We need to define a class named EmployeeUser to represent an employee's personal information. There are five private String variables in the class:
employeeId, Name, Email, Address, PhoneNumber

There is only one constructor in the class:
public EmployeeUser(String employeeId, String name, String email, String address, String phoneNumber)

The class contains two methods:
- **public String lineRepresentation**() : returns the data of the employee comma separated.
- **public String getSearchKey() :** returns the employee id.

The data of the employees is stored in a file named Employees.txt. Each line represents a data record of one employee. For example, the following line represents an employee whose employee id is E1200, name is Ahmed, email is ahmed_1999@gmail.com, address is Alexandria and phone number is 01088877345.

E1200,Ahmed,ahmed_1999@gmail.com,Alexandria,01088877345

Note: The employee id is unique for each employee.

Dr. Layla Abou-Hadeed

Eng. Ahmed El-Sayed
Eng. Miar Mamdouh Eng.
Eng. Mazen Sallam
Eng. Abdelrahman Wael
Eng. Muhannad Bashar

Eng. Ahmed Ashraf
Eng. AbdElaziz Mohamed
Eng. Shams Zayan
Eng. Mohamed Zaytoon

Second Class

Define a class named **EmployeeUserDatabase** which is responsible for accessing the file of the employees (reading from and writing to), accessing, and manipulating (adding to and removing from) the list of **EmployeeUser** objects that hold the data read from the file. There are two private variables in the class: **records (ArrayList<EmployeeUser>), filename(String).**

There is only one constructor in the class:

public EmployeeUserDatabase(String filename)

The class contains eight methods:

1. **public void readFromFile() :** opens the file whose name is stored in the instance variable filename, reads all the records of the employees and stores them in the arrayList referenced by the instance variable records.
2. **public EmployeeUser createRecordFrom(String line) :** returns an EmployeeUser object whose arguments are stored in the parameter that named line comma separated.
3. **public ArrayList<EmployeeUser> returnAllRecords() :** returns a reference on the arrayList referenced by the instance variable records.
4. **public boolean contains(String key) :** searches the arrayList referenced by the instance variable records and returns true if there is an EmployeeUser object whose employee id equals the parameter key.
5. **public EmployeeUser getRecord(String key) :** searches the arrayList referenced by the instance variable records and returns a reference on the EmployeeUser object whose employee id equals the parameter key.
6. **public void insertRecord(EmployeeUser record) :** inserts the object referenced by the parameter record into the arrayList referenced by the instance variable records.

7. **public void deleteRecord(String key) :** searches the arrayList referenced by the instance variable records and deletes the EmployeeUser object whose employee id equals the parameter key.
8. **public void saveToFile() :** deletes the old data stored in the file whose name is stored in filename, writes the data stored in the arrayList referenced by the instance variable records to the file. Each line represents the data of one employee comma separated.

Third Class

Define a class named **AdminRole** that represents the role of the admin. The class contains only one private variable named **database** which is an object from the **EmployeeUserDatabase** class. There is only one constructor in the class and its header is **public AdminRole()**.

The class contains four methods:

1. **public void addEmployee(String employeeId, String name, String email, String address, String phoneNumber) :** adds a new employee to the file named **Employees.txt**
2. **public EmployeeUser[] getListOfEmployees() :** returns an array that contains all the employees stored in the file named **Employees.txt**
3. **public void removeEmployee(String key) :** removes the employee whose employee id equals the parameter key from the file named **Employees.txt**
4. **public void logout() :** writes all unsaved data to the file named Employees.txt

Now, we will describe the role of the employee and the classes needed for this role.

The employee is responsible for adding new products to the inventory, selling products to customers, receiving the products, and adding them back to the inventory when the customers return them. So, we need some more classes for applying the employee's role.

First Class

Define a class named **Product** to represent a product's information.

The class contains:

- four private String variables: **productID, productName, manufacturerName, supplierName**.
- one private int variable**: quantity**.
- one private float variable: **price**.

The quantity variable represents the number of units of a product available to be sold. When a customer takes a unit of the product, the quantity decreases by one and when they return it, the quantity increases by one. If quantity equals zero, it means that all the units have been sold.
There is only one constructor in the class and its header is
public Product(String productID, String productName, String manufacturerName, String supplierName, int quantity, float price)

The class contains four methods:

1. **public int getQuantity()**
2. **public void setQuantity(int quantity)**
3. **public String lineRepresentation():** returns the data of the product comma separated.
4. **public String getSearchKey():** returns the product id.

The products' data are stored in a file named **Products.txt**. Each line represents a data record of one product. For example, the following line represents a product whose product id is P2394, product name is Laptop, manufacturer name is Apple, supplier name is TechSupplier, quantity is 10 and price 1500.

P2394,Laptop,Apple,TechSupplier,10,1500

Note: The product id is unique for each product.

Second Class

A class named **ProductDatabase** is responsible for accessing the file of the products (reading from and writing to), accessing, and manipulating (adding to and removing from) the list of Product objects that hold the data read from the file.

There are two private variables in the class**: records (ArrayList<Product>)** and **filename (String).**

There is only one constructor in the class and its header is
public ProductDatabase (String filename)

The class contains eight methods:

1. **public void readFromFile()** : opens the file whose name is stored in the instance variable filename, reads all the records of the products and stores them in the arrayList referenced by the instance variable records.

2. **public Product createRecordFrom(String line) :** returns a Product object whose arguments are stored in the parameter line comma separated.

3. **public ArrayList<Product> returnAllRecords()** : returns a reference on the arrayList referenced by the instance variable records.
4. **public boolean contains(String key) :** searches the arrayList referenced by the instance variable records and returns true if there is a Product object whose product id equals the parameter key.
5. **public Product getRecord(String key) :** searches the arrayList referenced by the instance variable records and returns a reference on the Product object whose product id equals the parameter key.
6. **public void insertRecord(Product record) :** inserts the object referenced by the parameter record into the arrayList referenced by the instance variable records.
7. **public void deleteRecord(String key) :** searches the arrayList referenced by the instance variable records and delete the Product object whose product id equals the parameter key.
8. **public void saveToFile() :** deletes the old data stored in the file whose name is stored in filename, writes the data stored in the arrayList referenced by the instance variable records to the file. Each line represents the data of one product comma separated.

Third Class

For simplicity:
On any given day, a customer purchases only one product and just only one unit of it.

Define a class named **CustomerProduct** that contains:

- Two private String variables: **customerSSN, productID**.
- One private LocalDate variable: **purchaseDate**.
- One private boolean variable named **paid** that indicates whether the product has been paid for or not.

The **customerSSN** variable represents the SSN of the customer who has purchased one or more units of the product whose id is stored in the **productID** since the date stored in **purchaseDate**. Each object of the **CustomerProduct** class represents a customer's purchase of a specific product.
There is only one constructor in the class and its header is:

public CustomerProduct(String customerSSN, String productID, LocalDate purchaseDate)

The class contains five methods:

1. **public String getCustomerSSN()**
2. **public String getProductID()**
3. **public LocalDate getPurchaseDate()**
4. **public String lineRepresentation() :** // returns the data of the object comma separated.
5. public **boolean isPaid():** returns true if the purchase is paid, false otherwise.
6. public **void setPaid(boolean paid):** updates the payment status.
7. **public String getSearchKey() :**

Dr. Layla Abou-Hadeed

Eng. Ahmed El-Sayed
Eng. Miar Mamdouh Eng.
Eng. Mazen Sallam
Eng. Abdelrahman Wael
Eng. Muhannad Bashar

Eng. Ahmed Ashraf
Eng. AbdElaziz Mohamed
Eng. Shams Zayan
Eng. Mohamed Zaytoon

```
/* Returns a concatenated string in the format: "customerSSN,productID,DD-MM-
YYYY" where DD-MM-YYYY represents the purchaseDate. DD represents the
day, MM represents the month and YYYY represents the year.*/
```

The data of purchasing operations is stored in a file named **CustomersProducts.txt**. Each line is a data record for a single purchasing transaction. The following line, for example, represents an object whose customer SSN (The one who purchased the product) is 7845345678, product id that has been purchased is P2568, purchase date is 12-02-2022 and the purchase has been paid:.

7845345678,P2568,12-02-2022,true.

Fourth Class

A class named **CustomerProductDatabase** is responsible for accessing the file of the purchasing operations (reading from and writing to), accessing, and manipulating (adding to and removing from) the list of **CustomerProduct** objects that hold the data read from the file.

The class contains two private variables:

- **records (ArrayList<CustomerProduct>)**
- **filename(String).**

There is only one constructor in the class and its header is

public CustomerProductDatabase (String filename)

Note: It should be noted that the argument key supplied to the methods described below is a String made up of three parts separated by a comma. The first part is a customer id, the second part is a product id and the third part is the string representation of the purchase date.

The class contains eight methods:

1. **public void readFromFile():** opens the file whose name is stored in the instance variable filename, reads all the records of the products' purchases and stores them in the arrayList referenced by the instance variable records.
2. **public CustomerProduct createRecordFrom(String line):** returns a CustomerProduct object whose arguments are stored in the parameter line comma separated.
3. **public ArrayList<CustomerProduct> returnAllRecords():** returns a reference on the arrayList referenced by the instance variable records.

4. **public boolean contains(String key):** searches the arrayList referred to by the instance variable records and returns true if there is a CustomerProduct object whose customerId, productID and purchaseDate can be combined to form a String = customerId+","+productID+","+ purchaseDate(in the format DD-MM-YYYY) that matches the parameter key.

5. **public CustomerProduct getRecord(String key):** searches the arrayList referred to by the instance variable records and returns a reference to the **CustomerProduct** object whose customerId, productID and purchaseDate can be combined to form a String = customerId+","+productID+","+ purchaseDate(in the format DD-MM-YYYY) that matches the parameter key.
6. **public void insertRecord(CustomerProduct record):** inserts the object referenced by the parameter record into the arrayList referenced by the instance variable records.

7. **public void deleteRecord(String key):** searches the arrayList referenced by the instance variable records and delete the CustomerProduct object whose customerId, productID and purchaseDate can be combined to form a String = customerId+","+productID+","+ purchaseDate(in the format DD-MM-YYYY) that matches the parameter key.

Dr. Layla Abou-Hadeed

Eng. Ahmed El-Sayed
Eng. Miar Mamdouh Eng.
Eng. Mazen Sallam
Eng. Abdelrahman Wael
Eng. Muhannad Bashar

Eng. Ahmed Ashraf
Eng. AbdElaziz Mohamed
Eng. Shams Zayan
Eng. Mohamed Zaytoon

8. **public void saveToFile():** deletes the old data stored in the file whose name is stored in filename, writes the data stored in the arrayList referenced by the instance variable records to the file. Each line represents the data of one product comma separated.

Fifth Class:

Define a class named **EmployeeRole** that represents the role of the employee. The class contains two private variables named **productsDatabase** and **customerProductDatabase** which are objects from **ProductDatabase** class and **CustomerProductDatabase** class respectively.
There is only one constructor in the class and its header is
public EmployeeRole()
The class contains only six methods:

1. **public void addProduct(String productID, String productName, String manufacturerName, String supplierName, int quantity):** adds a new product to the file named **Products.txt.**
2. **public Product[] getListOfProducts():** returns an array that contains all the products stored in the file named **Products.txt.**
3. **public CustomerProduct[] getListOfPurchasingOperations():** returns an array that contains all the purchasing operations stored in the file named **CustomersProducts.txt**

4. **public boolean purchaseProduct(String customerSSN, String productID, LocalDate purchaseDate):**
   returns **false** if the quantity variable of the product whose product id matches the parameter productID is zero. Otherwise, the method decreases the quantity variable of the product by one, adds a new purchasing operation to the file named CustomersProducts.txt, updates the file **Products.txt** and returns **true**.
5. **public double returnProduct(String customerSSN, String productID, LocalDate purchaseDate ,LocalDate returnDate):**
   Customers can return a product within 14 days of purchase. The method will return -1 if:
   - **returnDate** is earlier than **purchaseDate**
   - The product is not listed in Products.txt.
   - The string formed to be equal to **customerSSN+",",productID+","+purchaseDate (**in the format DD-MM-YYYY**)** is not listed in **CustomersProducts.txt.**
   - More than 14 days have passed since the purchase date.

   Otherwise, the method does the following:
   - increments the quantity variable of the product whose product id equals the parameter productID by one.
   - removes the line representing the purchasing operation from the file **CustomersProducts.txt.**
   - updates the file **Products.txt.**
   - returns the product's price.
6. **public boolean applyPayment(String customerSSN, LocalDate purchaseDate):**
   This method marks a customer purchase a product. It searches for the purchase

record in the that matches the given customerSSN and purchaseDate.
If the record exists and is not already marked as paid, the method updates the paid flag to true and saves the update.

7. **public void logout():** writes all unsaved data to the files named **Products.txt** and **CustomersProducts.txt.**

Part 2 (worth 25% of the grade)

By solving question 1, you have got 75% of the assignment mark. The remaining 25% are assigned for refactoring your code. If you give a look at your code in question 1, you will find many duplicated code. So, your task in question 2 to get the full mark is to enhance your code by applying OOP concepts on your code and make your code cleaner and reusable. OOP concepts like inheritance, polymorphism, abstraction, etc. You must apply OOP concepts on your code to get the full mark.

Note: Refactoring your code means enhancing it. Making your code cleaner and removing the duplicated code is what is meant by refactoring.

Alexandria University
Faculty of Engineering
Computer and Communication Engineering
Specialized Scientific Programs

CC272
Programming II
Fall 2025/2026

Required

1. You are required to solve the first question and get a 75% of the assignment mark. If you want to get the full mark, you should refactor your code and modify it to obey the OOP concepts (inheritance, polymorphism, abstraction, …) as mentioned in question 2.
2. A discussion is made with you at your lab time next week on what you have delivered.

What to be delivered

- On the google form, you should deliver a zipped file that contains the .java files of your classes.
- Your zip file should be named as id1_id2_id3_id4_groupNumber. For example, 4678_4557_4558_4559_G2.

Policies:

- You should work in groups of **four**.
- Delivering a copy will be severely penalized for both parties, so delivering nothing is so much better than delivering a copy.
- No late submission is allowed
- Each group must use **GitHub** throughout their lab work and submissions must be made through it.

Dr. Layla Abou-Hadeed

Eng. Ahmed El-Sayed
Eng. Miar Mamdouh Eng.
Eng. Mazen Sallam
Eng. Abdelrahman Wael
Eng. Muhannad Bashar

Eng. Ahmed Ashraf
Eng. AbdElaziz Mohamed
Eng. Shams Zayan
Eng. Mohamed Zaytoon