UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING

SEMESTER 2 2024/2025

SECJ2154 – OBJECT ORIENTED PROGRAMMING

SECTION 2

PROJECT PROPOSAL

MINI PROJECT TITLE: MyPerfume Ordering and Billing System

LECTURER: DR. ZURAINI BINTI ALI SHAH

GROUP 6

STUDENT NAME	MATRIC NO
CHUA JIA LIN	A23CS0069
POH LOK YEE	A23CS0262
MEERASHNI A/P THRUMOORTHY RAO	B24CS0018

INTRODUCTION

Small-scale perfume businesses often rely on manual processes such as handwritten logs, basic calculators, and spreadsheets for managing customer orders and billing. While manageable during low sales volume, this approach becomes increasingly inefficient and error-prone as the business grows. Key operational issues include:

- **Inefficient Sales Tracking**: Without a centralized system, tracking customer orders and identifying best-selling perfumes is cumbersome and lacks real-time insights.
- **Frequent Billing Errors**: Manual calculation of totals, especially for multiple items, increases the risk of human error, leading to inaccurate charges and potential customer dissatisfaction.
- **Disorganized Inventory Monitoring**: Manual stock updates can result in overselling or stockouts, harming customer trust and business reliability.
- **Lack of Receipt Documentation**: Verbal confirmations or paper receipts are prone to loss, miscommunication, and poor record-keeping.

To overcome these limitations, an automated solution is essential. The proposed **MyPerfume Ordering and Billing System** is a Java-based application that leverages object-oriented programming principles to streamline core business operations such as customer management, order processing, billing, inventory tracking, and digital receipt generation. By automating these functions, the system reduces human error, increases efficiency, enhances scalability, and provides better customer service.

PROBLEM STATEMENT

Manual implementation of perfume sales in small business typically relies on handwritten records, spreadsheets, or basic calculators. While this method may work for low-volume sales, it quickly becomes inefficient, error-prone, and difficult to scale as the business grows. Below are several significant challenges encountered in such a manual environment.

1. Lack of Organized Sales Tracking
   Perfume sales usually recorded manually without a centralized system. This leads to difficulty in retrieving past records, calculating overall performance, or monitoring the popularity of certain products over time.

2. Manual Bill Calculation Errors
   Calculations for total costs are done by hand or calculator, especially when multiple perfume types are purchased at once. This increases the likehood of arithmetic errors, undercharging or overcharging customers.

3. Poor Inventory Awareness
   Manual sales tracking does not automatically reflect stock levels. Employees must manually update inventory, which increases the chance of running out of stock unknowingly or ordering excess supply.

4. No automated Receipt Generation
   Paper receipts or verbal confirmations are used instead of digital receipts. These are easy to lose and difficult to organize, leading to poor documentation and accountability.

To address the limitations of manual processes, the MyPerfume Ordering and Billing System application introduces a structured, object-oriented solution developed using Java. This system automates key processes such as customer management, order placement, billing, and sales tracking to improve business efficiency and reduce human error. The following functionalities are included in the system:

1. Automated Sales Tracking per Perfume
   Each perfume item is represented as a Perfume object with built-in attributes for tracking quantity sold and total sales. This allows for real-time monitoring of product performance and simplifies sales reporting

2. Customer Management
   A dedicated Customer class stores the name and contact details of each buyer. This enables future referencing, personalized service, and potential marketing or loyalty

programs.

3. Order handling with Quantity Management
   Orders are captured using an abstract Order class that allows for adding multiple perfumes with specific quantities. This ensures flexibility and efficiency when processing different types of purchases.

4. Total Cost Calculation via Interface
   By implementing a Bill interface, the system provides a clean structure to calculate the total amount due. This ensures billing consistency and avoids calculation errors.

5. Bill Generation in Text Files
   The system generates digital receipts by writing bill details into a .txt file, ensuring each transaction is properly documented. These receipts can be stored, shared, or printed as needed.

6. Object-Oriented Design for Scalability
   The modular code structure using classes, interfaces, and abstraction allows easy enhancement of features, such as integrating discounts, applying tax, or transitioning to a graphical interface in the future.

CLASS DIAGRAM OF THE PROJECT

The following diagram is the class diagram for the MyPerfume Ordering and Billing System.

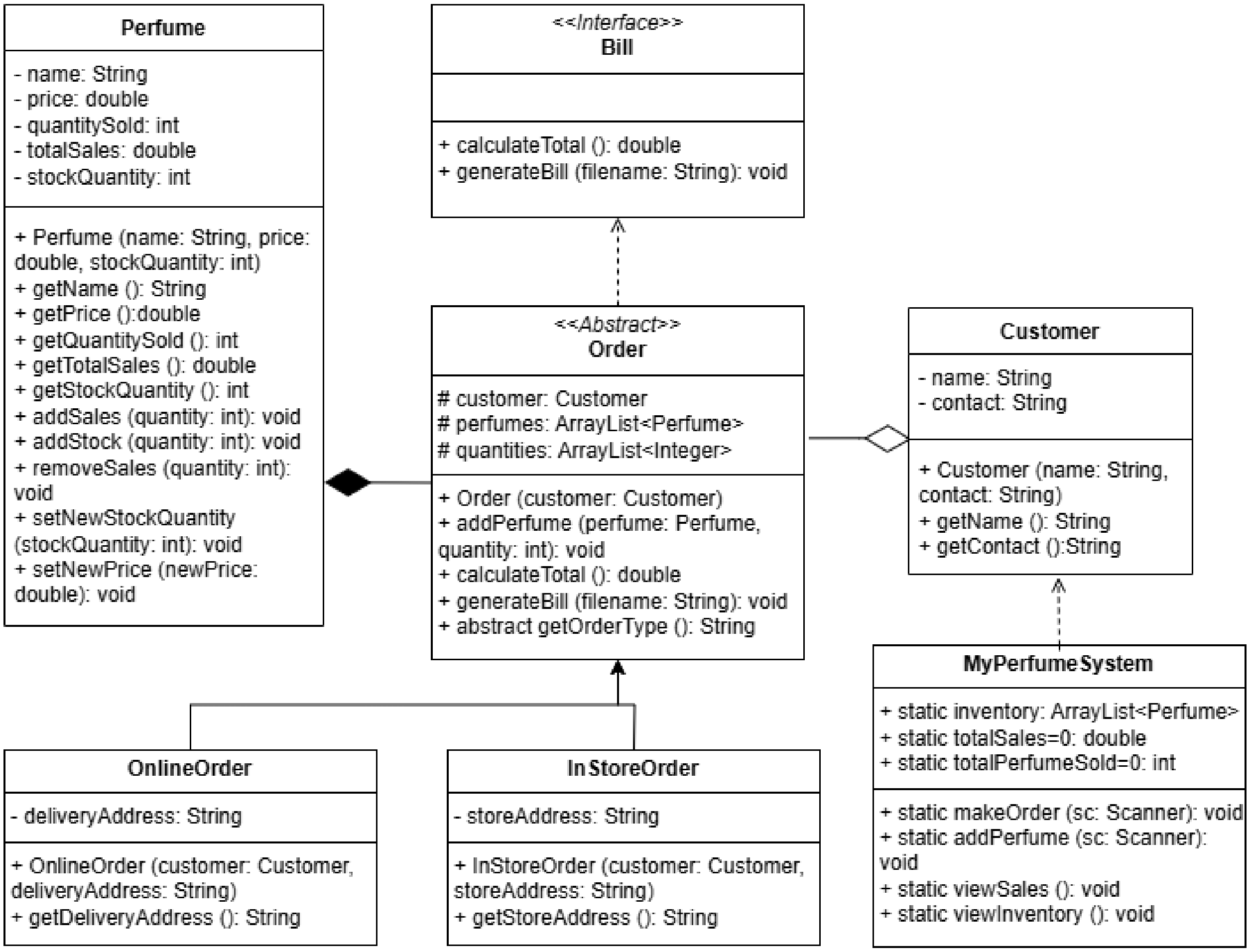
Perfume
- name: String
- price: double
- quantitySold: int
- totalSales: double
- stockQuantity: int
+ Perfume (name: String, price: double, stockQuantity: int)
+ getName (): String
+ getPrice ():double
+ getQuantitySold (): int
+ getTotalSales (): double
+ getStockQuantity (): int
+ addSales (quantity: int): void
+ addStock (quantity: int): void
+ removeSales (quantity: int): void
+ setNewStockQuantity (stockQuantity: int): void
+ setNewPrice (newPrice: double): void
<<Interface>>
Bill
+ calculateTotal (): double
+ generateBill (filename: String): void
<<Abstract>>
Order
# customer: Customer
# perfumes: ArrayList<Perfume>
# quantities: ArrayList<Integer>
+ Order (customer: Customer)
+ addPerfume (perfume: Perfume, quantity: int): void
+ calculateTotal (): double
+ generateBill (filename: String): void
+ abstract getOrderType (): String
Customer
- name: String
- contact: String
+ Customer (name: String, contact: String)
+ getName (): String
+ getContact ():String
MyPerfumeSystem
+ static inventory: ArrayList<Perfume>
+ static totalSales=0: double
+ static totalPerfumeSold=0: int
+ static makeOrder (sc: Scanner): void
+ static addPerfume (sc: Scanner): void
+ static viewSales (): void
+ static viewInventory (): void
OnlineOrder
- deliveryAddress: String
+ OnlineOrder (customer: Customer, deliveryAddress: String)
+ getDeliveryAddress (): String
InStoreOrder
- storeAddress: String
+ InStoreOrder (customer: Customer, storeAddress: String)
+ getStoreAddress (): String

Figure 1 Class Diagram of MyPerfume Ordering and Billing System

The table below shows the person in charge of completing the coding of the classes of the MyPerfume Ordering and Billing System.

Table 1 Person in Charge of Classes

Class	Person In Charge
Customer	Meerashni A/P Thrumoorthy Rao
Perfume	Chua Jia Lin
Order	Poh Lok Yee
OnlineOrder	Poh Lok Yee
InStoreOrder	Poh Lok Yee
Bill	Meerashni A/P Thrumoorthy Rao
MyPerfumeSystem	Chua Jia Lin

TECHNIQUES/SOFTWARE/TOOLS BEING USED

ENUM		**ASSOCIATION***	/
ARRAYLIST / VECTOR	/	COMPOSITION	
ARRAY		AGGREGATION	
ARRAY OF OBJECT		**INHERITANCE***	/
FILE*	/	STATIC METHOD	/
POLYMORPHISM*	/	**ABSTRACT CLASS**	/
INTERFACE CLASS*	/	**EXCEPTION HANDLING***	/
GUI APPLICATION		OTHERS:	

- * compulsory – also comments, style etc

For this project, we plan to use Draw.io to create the class diagram. Besides, we will use an object-oriented approach, which is the Java programming language to develop a solution for the system. Other than that, we will use Visual Studio Code as a platform to complete our coding. At the same time, we will use GitHub for version control.