Mall Management System

OOP Project Proposal

Group J:

ABDULLAH SHAHID (BSSE-24001)
HUSSAIN DAUD SYED (BSSE-24021)
MUHAMMAD UZAIR QAZI (BSSE-24044)
MUHAMMAD HASSAN ALI (BSSE-24038)
MUHAMMAD MUZAHIR ABBAS (BSSE-24060)

Mall Management System

Scenario:

1. Store & Inventory Management

The mall has clothing, electronics, and food stores. Each store has an inventory, and the system keeps track of stock.

🏬 **Example:** A clothing store runs out of summer shirts. The system alerts the owner to restock.

2. Employee & Security Management

The mall hires cashiers, managers, and security guards. Employees have different roles, and security ensures safe access.

👮 **Example:** A manager can check financial reports, but only security can access the control room.

3. Customer & Billing System

Customers are regular or VIP members, and they get discounts based on their status. Every purchase generates an invoice.

💳 **Example:** A VIP customer buys a phone and gets a 20% discount automatically.

4. Parking & Advertisement System

The mall has parking for cars, bikes, and trucks. Ads and promotions run on LED screens inside the mall.

🚗 **Example:** A concert is happening, and video ads are displayed near the food court.

5. Online Booking & Event Management

Customers can book movie tickets or restaurant tables. The system also handles concerts and other events.

🎟️ **Example:** A customer books a dinner and a concert ticket, and the system ensures no double booking.

Project Task Distribution

Abdullah Shahid:

(Store Management + Inventory & Stock Management)

Store Management

- A **Mall** consists of various stores, including clothing stores, electronics stores, and food courts. This module will handle different store types using **inheritance**, where a general Store class will be extended by **ClothingStore**, **ElectronicsStore**, and **FoodCourt**.
- The **composition** concept is applied where a Mall will have multiple Store objects as part of its structure.
- **Operator overloading** will be used to compare store revenues, for instance, determining which store has the highest sales **(store1 > store2).**
- To ensure data persistence, **filing** will be implemented, allowing store data to be saved and managed efficiently in **stores.json**.

Inventory & Stock Management

Stores need to manage their inventory, ensuring that stock levels are monitored and updated.

- An **aggregation** relationship is established where Store aggregates Inventory, meaning that even if a store is removed, its inventory data can exist independently.
- **Polymorphism** is used in stock management by creating subclasses of StockItem, namely Perishable and NonPerishable, each with different restocking rules.
- **Operator overloading** will allow comparisons between stock levels of different inventories **(inventory1 == inventory2**).
- Stock updates and changes will be tracked using **filing**, where inventory data is stored in **inventory.json**.

Hussain Daud Syed:

(Employee Management + Security & Access Control)

Employee Management

- Employees working in the mall are categorized into different roles such as **Cashier, Security, and Manager**. These roles inherit from a **base class Employee**, implementing **inheritance**.
- A store **aggregates** employees, meaning that employees work for a store but can also exist independently if needed.

- Employee records will be stored in **employees.json** using **filing** for maintaining data.

Security & Access Control

- A **SecuritySystem** will be **composed** of LogManager and AccessController, ensuring secure access management within the mall.
- **Polymorphism** will be used in access control, where different access rules apply to EmployeeAccess and CustomerAccess.
- **Operator overloading** is implemented to compare log sizes **(log1 > log2)**.
- A **singleton pattern** will be applied to ensure a centralized security system instance is used.

Muhammad Uzair Qazi:

(Customer & Membership Management + Billing & Sales System)

Customer & Membership Management

- Customers visiting the mall are classified as either RegularCustomer or VIPCustomer, following an **inheritance** structure.
- A **polymorphic** discount system will be implemented, with different discount strategies for VIP members and seasonal discounts (VIPDiscount, SeasonalDiscount).
- Customers will be associated with stores and transactions through **association** relationships.
- All customer records will be saved in **customers.json** using **filing**.

Billing & Sales System

- Each transaction in the mall is recorded using an **Invoice**, which is **composed** of a Customer and Transaction.
- **Operator overloading** will be used to apply discounts (e.g., **invoice * 0.2**).
- Stores will **aggregate** invoices, meaning all sales records are managed under each store.
- Invoices and sales transactions will be stored in sales.json using **filing**.

Muhammad Hassan Ali:

(Parking Management + Advertisement & Promotions)

Parking Management

- Parking spaces within the mall are managed through ParkingLot, which is **aggregated** by the mall. Parking lots exist independently but are associated with the mall structure.
- Vehicles are categorized into different types (Car, Bike, Truck) using **inheritance**.

- **Operator overloading** will be used to compare parking occupancy levels **(parking1 < parking2).**
- Vehicle entry and exit logs will be recorded in **parking.json** using **filing**.

Advertisement & Promotions

- Ad campaigns are structured using **composition**, where an AdCampaign consists of Promotion and Schedule.
- Different types of advertisements (BannerAd, VideoAd) are implemented using **polymorphism**.
- **Operator overloading** will be used to compare advertisement performances **(ad1 > ad2).**
- Promotional records will be stored in **ads.json** using **filing**.

Muhammad Muzahir Abbas:

(Online Booking + Events & Entertainment Management)

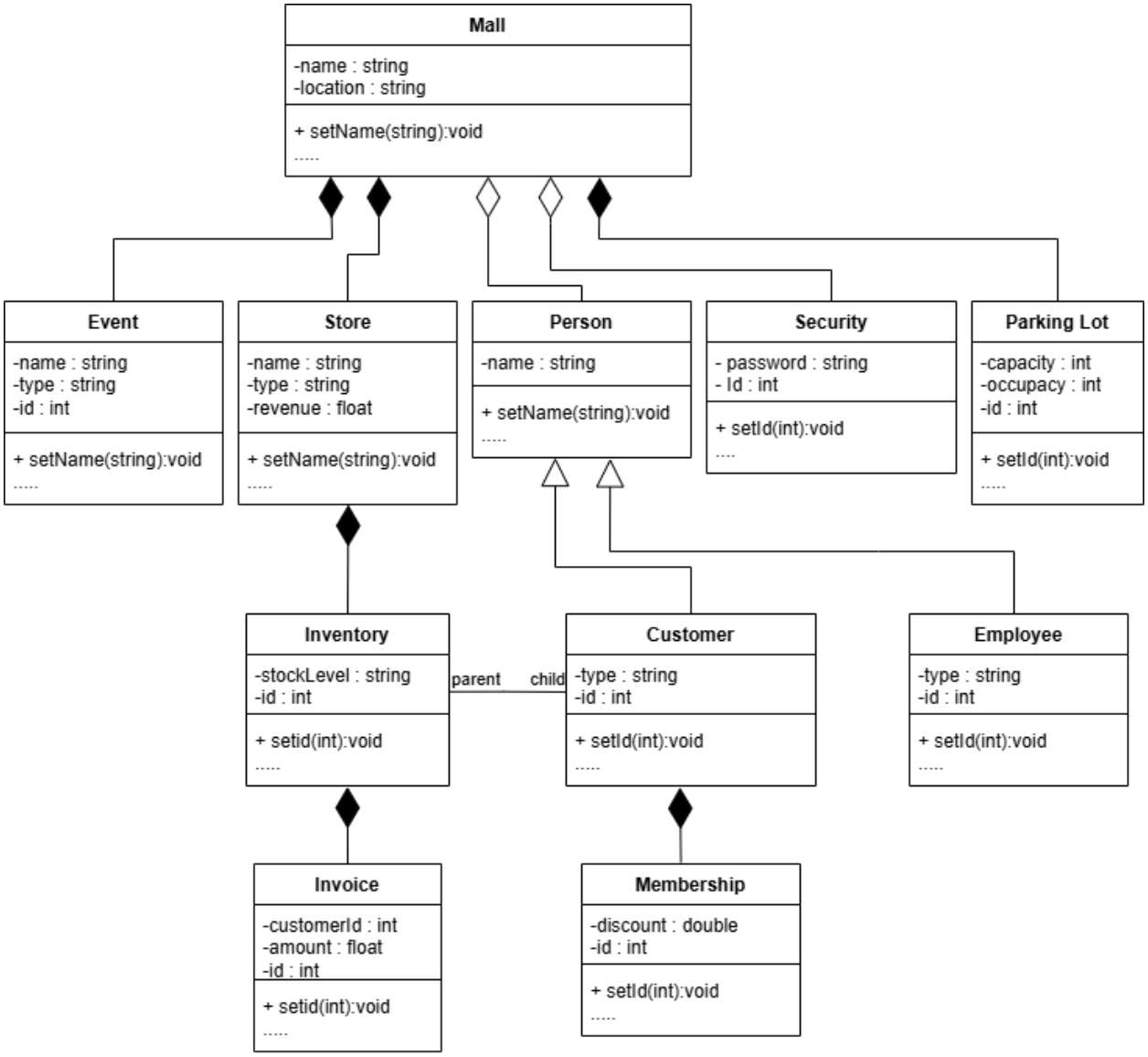
Online Booking

- Customers can book tickets for events, movies, and restaurant reservations. The booking system follows an **inheritance** structure where Booking is extended by MovieTicket and RestaurantReservation.
- Each booking is **composed** of a Customer and a Service (e.g., a movie or restaurant table).
- **Operator overloading** will be applied to compare booking durations (booking1 >= booking2).
- All booking records will be stored in **reservations.json** using **filing**.

Events & Entertainment Management

- Events happening in the mall, such as concerts and workshops, are managed under the Event class, which is **aggregated** by the mall.
- Different event types (Concert, Workshop) are implemented using **polymorphism**.
- **Operator overloading** is used to compare event attendance (event1 > event2).
- All event records will be stored in **events.json** using **filing**.

Basic UML Diagram


Mall
-name : string
-location : string
+ setName(string):void
.....
Event
-name : string
-type : string
-id : int
+ setName(string):void
.....
Store
-name : string
-type : string
-revenue : float
+ setName(string):void
.....
Person
-name : string
+ setName(string):void
.....
Security
- password : string
- Id : int
+ setId(int):void
....
Parking Lot
-capacity : int
-occupacy : int
-id : int
+ setId(int):void
.....
Inventory
-stockLevel : string
-id : int
+ setid(int):void
.....
parent
child
Customer
-type : string
-id : int
+ setId(int):void
.....
Employee
-type : string
-id : int
+ setId(int):void
.....
Invoice
-customerId : int
-amount : float
-id : int
+ setid(int):void
.....
Membership
-discount : double
-id : int
+ setId(int):void
.....