PES University

Department of Computer Science and Engineering

UE22CS352A: OBJECT ORIENTED ANALYSIS AND DESIGN USING JAVA

Lab-3: UML Class Diagram

Prepare a detailed class diagram for a workspace management system, including all classes, their methods and attributes (with visibility notation), relevant relationships (dependencies, generalizations, associations) with cardinalities, enumeration, etc. as described by the case study given below.

Problem 1: Workspace management system

Imagine designing a workspace management system to simplify how employees, administrators, and managers interact with shared office spaces. The goal is to create an efficient and user-friendly system that caters to diverse requirements, such as booking desks and meeting rooms, managing workspace amenities, and processing payments, while ensuring smooth coordination among all roles involved.

The system should support different user types: employees, who primarily book and use the workspace; admins, who manage users and generate reports; and workspace managers, who oversee space utilization, amenities, and availability. Each user is represented by a User class with attributes like userID, name, email, and role. These users can perform actions such as registering, logging in, updating their profile, and accessing role-specific functionalities.

To facilitate workspace management, the Workspace class is essential. It represents individual office spaces with attributes like workspaceID, name, location, and availability metrics such as totalDesks, meetingRooms, and availableRooms. Managers can add new workspaces, update workspace details, and check availability to ensure optimal resource utilization.

The system needs to manage desks and meeting rooms within workspaces. Desks, represented by the Desk class, have attributes like deskID and status, which indicate whether the desk is available or unavailable, as defined by the AvailabilityStatus enum. Desks also come with a list of associated amenities, enhancing user experience. Similarly, meeting rooms, encapsulated in the Room class, have attributes like roomID, capacity, status, and associated amenities. Users can reserve or release rooms, ensuring efficient usage of shared resources.

Amenities, such as whiteboards, projectors, and charging stations, are represented by the Amenity class. Each amenity has attributes like amenityID, name, and workspaceID. Managers can add or update amenities and check their availability. These amenities are tightly integrated into desk and room functionalities, improving the workspace's overall utility.

Bookings play a central role in this system and are managed by the Booking class. Each booking includes attributes like bookingID, userID, workspaceID, deskID, or roomID, along with booking times and types. Users can create, update, or cancel bookings, ensuring flexibility in managing their schedules. The booking system must seamlessly handle changes in desk and room availability, ensuring the current state of each resource is accurately reflected as either Available or Not Available.

To keep users informed, the Notification class provides timely updates. Notifications are linked to specific users through userID and can communicate booking confirmations, cancellations, or updates. Notifications have attributes like notificationID, message, and status and can be marked as read or sent as required.

Payments are a critical aspect of the system, managed by the Payment class. Payment records include paymentID, userID, amount, date, and paymentStatus. The system must enable users to process payments and view their payment history effortlessly, ensuring smooth financial transactions.

The platform also incorporates higher-level administrative features through the Admin role. Admins can manage users, generate reports, and oversee workspace

usage. They can view available spaces, make bookings, and monitor overall system performance, ensuring that all operations run efficiently.

To unify these diverse functionalities, the system leverages the Notifiable interface, ensuring consistent notification behavior across different classes. This design ensures that all users and resources stay interconnected, providing a seamless experience.

How would you approach building this workspace management system, ensuring scalability, reliability, and ease of use? The challenge lies in designing the relationships between these classes, handling concurrency in bookings, managing availability statuses, and implementing robust notification and payment mechanisms.

References

Title	Link
What is a class diagram?	Class Diagram
Softwares Available	Draw.io , SmartDraw , StarUML

Types of Relationships between Classes

Relationships