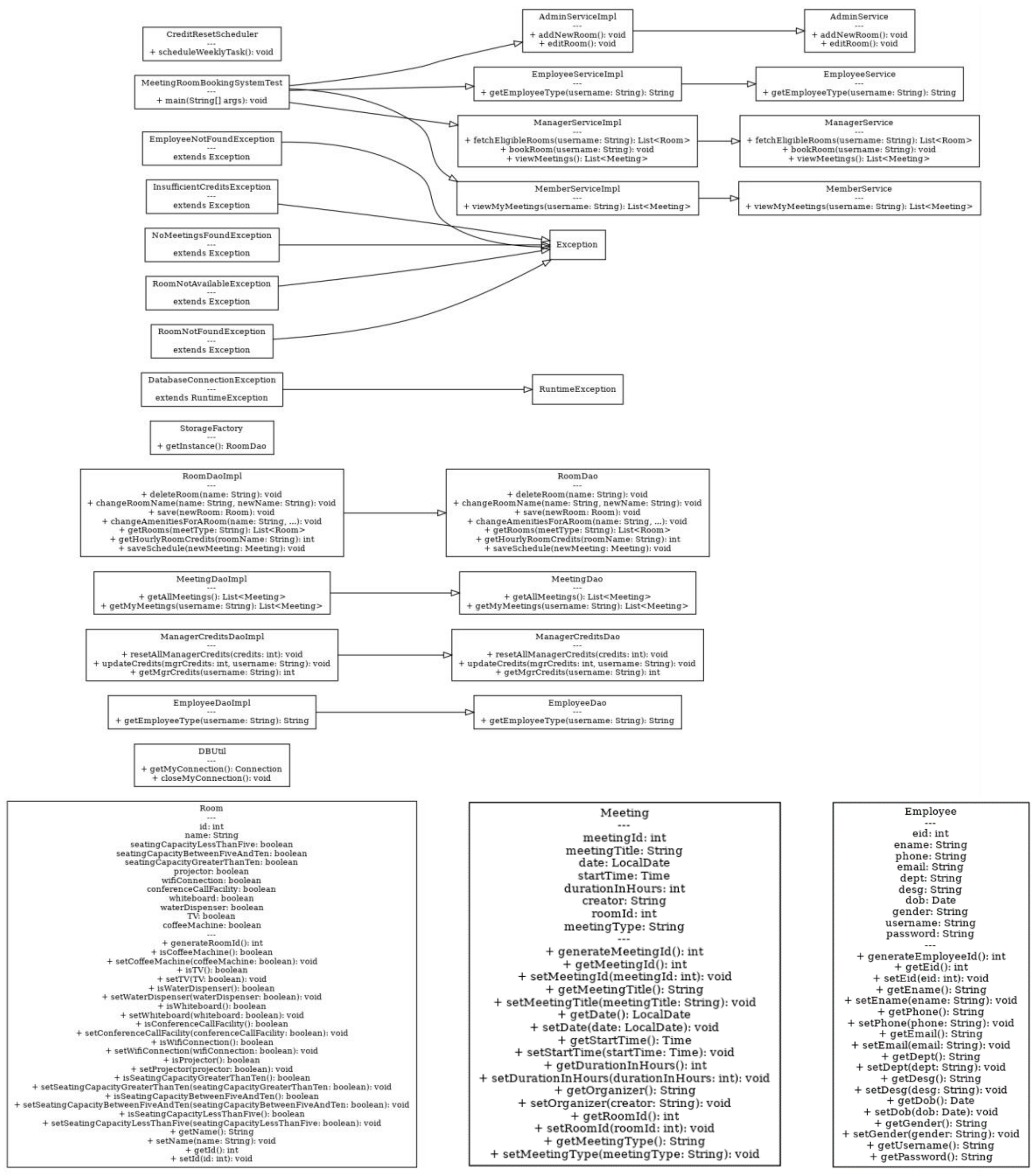
Automated-Meeting-Room-Booking-System

UML Class Diagram


CreditResetScheduler --- + scheduleWeeklyTask(): void
MeetingRoomBookingSystemTest --- + main(String[] args): void
AdminServiceImpl --- + addNewRoom(): void + editRoom(): void
AdminService --- + addNewRoom(): void + editRoom(): void
EmployeeServiceImpl --- + getEmployeeType(username: String): String
EmployeeService --- + getEmployeeType(username: String): String
ManagerServiceImpl --- + fetchEligibleRooms(username: String): List<Room> + bookRoom(username: String): void + viewMeetings(): List<Meeting>
ManagerService --- + fetchEligibleRooms(username: String): List<Room> + bookRoom(username: String): void + viewMeetings(): List<Meeting>
MemberServiceImpl --- + viewMyMeetings(username: String): List<Meeting>
MemberService --- + viewMyMeetings(username: String): List<Meeting>
EmployeeNotFoundException --- extends Exception
InsufficientCreditsException --- extends Exception
NoMeetingsFoundException --- extends Exception
RoomNotAvailableException --- extends Exception
RoomNotFoundException --- extends Exception
Exception
DatabaseConnectionException --- extends RuntimeException
RuntimeException
StorageFactory --- + getInstance(): RoomDao
RoomDaoImpl --- + deleteRoom(name: String): void + changeRoomName(name: String, newName: String): void + save(newRoom: Room): void + changeAmenitiesForARoom(name: String, ...): void + getRooms(meetType: String): List<Room> + getHourlyRoomCredits(roomName: String): int + saveSchedule(newMeeting: Meeting): void
RoomDao --- + deleteRoom(name: String): void + changeRoomName(name: String, newName: String): void + save(newRoom: Room): void + changeAmenitiesForARoom(name: String, ...): void + getRooms(meetType: String): List<Room> + getHourlyRoomCredits(roomName: String): int + saveSchedule(newMeeting: Meeting): void
MeetingDaoImpl --- + getAllMeetings(): List<Meeting> + getMyMeetings(username: String): List<Meeting>
MeetingDao --- + getAllMeetings(): List<Meeting> + getMyMeetings(username: String): List<Meeting>
ManagerCreditsDaoImpl --- + resetAllManagerCredits(credits: int): void + updateCredits(mgrCredits: int, username: String): void + getMgrCredits(username: String): int
ManagerCreditsDao --- + resetAllManagerCredits(credits: int): void + updateCredits(mgrCredits: int, username: String): void + getMgrCredits(username: String): int
EmployeeDaoImpl --- + getEmployeeType(username: String): String
EmployeeDao --- + getEmployeeType(username: String): String
DBUtil --- + getMyConnection(): Connection + closeMyConnection(): void
Room --- id: int name: String seatingCapacityLessThanFive: boolean seatingCapacityBetweenFiveAndTen: boolean seatingCapacityGreaterThanTen: boolean projector: boolean wifiConnection: boolean conferenceCallFacility: boolean whiteboard: boolean waterDispenser: boolean TV: boolean coffeeMachine: boolean --- + generateRoomId(): int + isCoffeeMachine(): boolean + setCoffeeMachine(coffeeMachine: boolean): void + isTV(): boolean + setTV(TV: boolean): void + isWaterDispenser(): boolean + setWaterDispenser(waterDispenser: boolean): void + isWhiteboard(): boolean + setWhiteboard(whiteboard: boolean): void + isConferenceCallFacility(): boolean + setConferenceCallFacility(conferenceCallFacility: boolean): void + isWifiConnection(): boolean + setWifiConnection(wifiConnection: boolean): void + isProjector(): boolean + setProjector(projector: boolean): void + isSeatingCapacityGreaterThanTen(): boolean + setSeatingCapacityGreaterThanTen(seatingCapacityGreaterThanTen: boolean): void + isSeatingCapacityBetweenFiveAndTen(): boolean + setSeatingCapacityBetweenFiveAndTen(seatingCapacityBetweenFiveAndTen: boolean): void + isSeatingCapacityLessThanFive(): boolean + setSeatingCapacityLessThanFive(seatingCapacityLessThanFive: boolean): void + getName(): String + setName(name: String): void + getId(): int + setId(id: int): void
Meeting --- meetingId: int meetingTitle: String date: LocalDate startTime: Time durationInHours: int creator: String roomId: int meetingType: String --- + generateMeetingId(): int + getMeetingId(): int + setMeetingId(meetingId: int): void + getMeetingTitle(): String + setMeetingTitle(meetingTitle: String): void + getDate(): LocalDate + setDate(date: LocalDate): void + getStartTime(): Time + setStartTime(startTime: Time): void + getDurationInHours(): int + setDurationInHours(durationInHours: int): void + getOrganizer(): String + setOrganizer(creator: String): void + getRoomId(): int + setRoomId(roomId: int): void + getMeetingType(): String + setMeetingType(meetingType: String): void
Employee --- eid: int ename: String phone: String email: String dept: String desg: String dob: Date gender: String username: String password: String --- + generateEmployeeId(): int + getEid(): int + setEid(eid: int): void + getEname(): String + setEname(ename: String): void + getPhone(): String + setPhone(phone: String): void + getEmail(): String + setEmail(email: String): void + getDept(): String + setDept(dept: String): void + getDesg(): String + setDesg(desg: String): void + getDob(): Date + setDob(dob: Date): void + getGender(): String + setGender(gender: String): void + getUsername(): String + getPassword(): String

1. Employee Class

- Attributes:
  - ` eid ` : Integer, represents the employee ID.
  - ` ename ` : String, represents the employee's name.
  - ` phone ` : String, represents the employee's phone number.
  - ` email ` : String, represents the employee's email address.
  - ` dept ` : String, represents the employee's department.
  - ` desg ` : String, represents the employee's designation.
  - ` dob ` : Date, represents the employee's date of birth.
  - ` gender ` : String, represents the employee's gender.
  - ` username ` : String, represents the employee's username.
  - ` password ` : String, represents the employee's password.

- Methods:
  - ` generateEmployeeId() ` : Generates a unique employee ID.
  - Getter and setter methods for each attribute.

2. Meeting Class

- Attributes:
  - ` meetingId ` : Integer, represents the unique ID of the meeting.
  - ` meetingTitle ` : String, represents the title of the meeting.
  - ` date ` : LocalDate, represents the date of the meeting.
  - ` startTime ` : Time, represents the start time of the meeting.
  - ` durationInHours ` : Integer, represents the duration of the meeting in hours.
  - ` creator ` : String, represents the creator or organizer of the meeting.
  - ` roomId ` : Integer, represents the ID of the room where the meeting is scheduled.
  - ` meetingType ` : String, represents the type of the meeting.

- Methods:
  - ` generateMeetingId() ` : Generates a unique meeting ID.
  - Getter and setter methods for each attribute.

3. Room Class

- Attributes:
  - `id`: Integer, represents the unique ID of the room.
  - `name`: String, represents the name of the room.
  - Various Boolean attributes to indicate the presence of specific amenities in the room, such as seating capacity, projector, Wi-Fi, etc.

- Methods:
  - `generateRoomId()`: Generates a unique room ID.
  - Getter and setter methods for each attribute.

4. DAO (Data Access Object) Interfaces and Implementations

- EmployeeDao Interface:
  - Method: `getEmployeeType(username: String): String` - Retrieves the type of employee based on the username.

- EmployeeDaoImpl Class:
  - Implements the `EmployeeDao` interface.
  - Provides the implementation for the `getEmployeeType` method.

- ManagerCreditsDao Interface:
  - Methods to reset, update, and get manager credits based on the username.

- ManagerCreditsDaoImpl Class:
  - Implements the `ManagerCreditsDao` interface.
  - Provides implementations for methods related to manager credits.

- MeetingDao Interface:
  - Methods to get all meetings and get meetings for a specific user.

- MeetingDaoImpl Class:

- Implements the `MeetingDao` interface.
  - Provides implementations for methods related to meetings.

- RoomDao Interface:
  - Methods to manage rooms, such as adding, deleting, updating, and fetching room details.

- RoomDaoImpl Class:
  - Implements the `RoomDao` interface.
  - Provides implementations for methods related to room management.

5. Service Layer
- AdminService Interface:
  - Methods to add and edit rooms.

- AdminServiceImpl Class:
  - Implements the `AdminService` interface.
  - Provides implementations for adding and editing rooms.

- EmployeeService Interface:
  - Method: `getEmployeeType(username: String): String` - Retrieves the employee type.

- EmployeeServiceImpl Class:
  - Implements the `EmployeeService` interface.
  - Provides the implementation for the `getEmployeeType` method.

- ManagerService Interface:
  - Methods to fetch eligible rooms, book a room, and view meetings.

- ManagerServiceImpl Class:
  - Implements the `ManagerService` interface.
  - Provides implementations for room booking and meeting management for managers.

- MemberService Interface:

  - Method: `viewMyMeetings(username: String): List<Meeting>` - Retrieves the meetings scheduled for a specific user.

- MemberServiceImpl Class:

  - Implements the `MemberService` interface.

  - Provides the implementation for viewing meetings for a specific user.

6. Utility Classes

- DBUtil Class:

  - Provides methods to establish and close database connections.

- StorageFactory Class:

  - Provides a factory method to get an instance of `RoomDaoImpl`.

7. Scheduler Class

- CreditResetScheduler Class:

  - Schedules a task to reset manager credits weekly.

8. Exception Handling

- Custom exceptions such as `DatabaseConnectionException`, `EmployeeNotFoundException`, `InsufficientCreditsException`, `NoMeetingsFoundException`, `RoomNotAvailableException`, and `RoomNotFoundException` are defined to handle specific errors in the system.

9. Test Class

- MeetingRoomBookingSystemTest Class:

  - Contains the `main` method to run the program, which simulates different user interactions (Admin, Manager, Member) with the system.

Relationships:

- Composition and Aggregation:

  - The `Meeting` class is associated with both the `Employee` and `Room` classes, indicating that a meeting involves an employee and a room.

- Inheritance:

  - Custom exception classes inherit from the base `Exception` class or `RuntimeException`.

General Flow:

- The user interacts with the system through the `MeetingRoomBookingSystemTest` class, which determines the user type and invokes the appropriate services.

- Admins can add or edit rooms, managers can view or book meetings, and members can view their scheduled meetings.

- The service layer interacts with the DAO layer to perform CRUD operations on the database.

- Utility classes manage database connections and factory creation for DAOs.

- Exception handling is implemented to manage specific errors during the execution.

This UML diagram provides a high-level overview of the classes and their relationships within the Meeting Room Booking System, making it easier to understand the system architecture and interactions.