Call Management System Design Documentation

Author: Samir Sen

Date: May 31, 2024

Contents

Introduction

This document outlines the architecture and design principles of a Call Management System implemented in Java.

Class Diagram

The class diagram depicts a Call Management System with three classes: User, Call, and CallManager. A User object has a one-to-many relationship with Call objects, meaning a single user can participate in many calls (as caller or receiver), while each call has only one caller and one receiver. The CallManager class has a one-to-many relationship with Call objects, managing a collection of calls within the system.

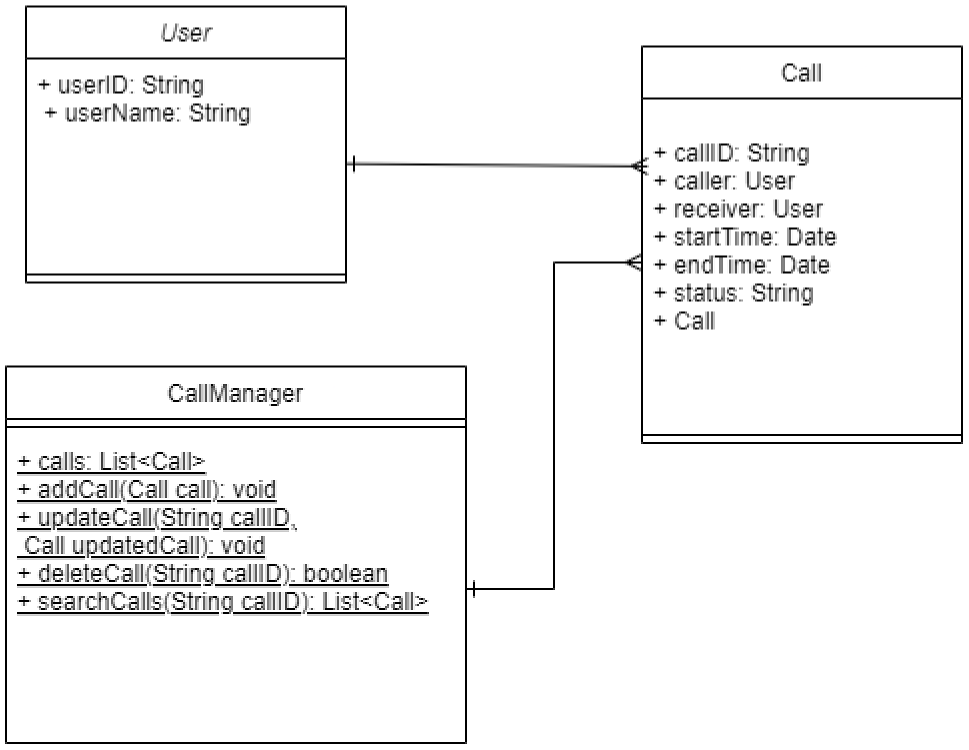

Figure 1 Overall Class Diagram

System Architecture

The system utilizes a layered architecture with three main classes:

call.management.User: Represents a user participating in a call (caller or receiver).

call.management.Call: Represents a single call record with attributes like call ID, caller, receiver, start/end times, and status.

call.management.CallManager: Manages a collection of Call objects and provides methods to add, update, delete, and search for calls. **Class Descriptions and Methods**

1. User Class:

This class represents a user involved in a call (either caller or receiver). It has attributes for user ID and user name.

2. Call Class:

This class represents a single call record. It includes attributes for call ID, caller (User object), receiver (User object), start time (Date object), end time (Date object), and call status (e.g., "missed," "completed"). The constructor initializes the call object with these details.

3. CallManager Class:

This class manages a collection of Call objects stored in an internal List. It provides methods to:

- addCall(Call call): Adds a new Call object to the list.
- updateCall(String callID, Call updatedCall): Updates an existing call based on its call ID. It iterates through the list, searching for the matching call ID, and replaces it with the provided updated Call object.
- deleteCall(String callID): Deletes a call from the list based on its call ID. It iterates through the list, searching for the matching call ID, and removes it if found. Returns true if successful, false otherwise.
- searchCalls(String callID): Returns a list of Call objects matching the provided call ID. It iterates through the list and adds any Call objects with the matching call ID to a new list, which is then returned.

Applying Object-Oriented Programming Concepts

The design adheres to several OOP principles:

- **Encapsulation:** Each class has private attributes and provides public getter/setter methods (not shown here for brevity) to control access to its data.
- **Inheritance:** Not directly implemented in this example, but future extensions could utilize inheritance for different call types (e.g., MissedCall inheriting from Call).
- **Polymorphism:** The searchCalls method demonstrates polymorphism by returning a List<Call>, which can hold calls of any type if inheritance is introduced.

This design promotes modularity, maintainability, and potential for future extension through inheritance.

Conclusion

In conclusion, this Call Management System demonstrates the benefits of object-oriented programming. By utilizing encapsulation, a well-defined class structure, and the potential for inheritance, this design offers a flexible and maintainable foundation for managing call data. Future enhancements could include adding error handling, implementing data persistence mechanisms (e.g., storing calls in a database), and exploring inheritance for different call types.