Program Documentation

Java Final Sprint

Due April 17, 2025

Sarah Murphy, Zachary Collier, Kyle/Scarlett Budgell

User Documentation

We built this program as a management system for a gym that can be accessed by members, trainers, and admins. The application handles a typical gym's day-to-day operations, which include member and membership information, available workout classes, and revenue that the gym brings in. This system is run through your integrated development environment's (IDE) console and can be run easily using the program's menu.

General Overview

This program contains three roles: Admin, Trainer, and Member. These roles determine which output you will receive in the console. Admins have full access to the user management system. This includes financial information about the gym, memberships and workout classes, while Trainers can view their assigned classes and members enrolled in their classes, and Members can see their personal membership details, as well as available workout classes.

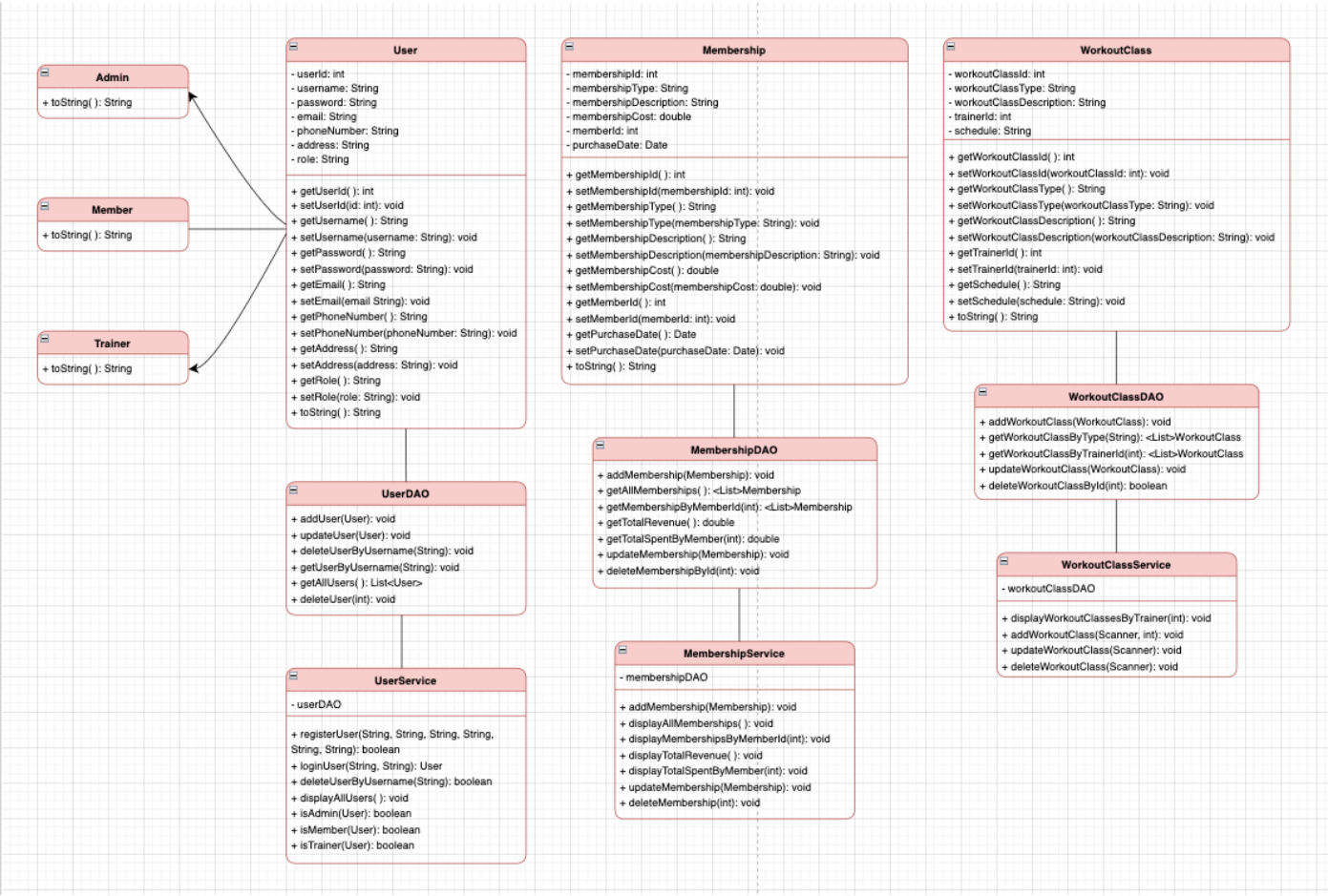
Classes and Interactions

1. User: The User class represents a user in the system. Users include Admins, Members and Trainers.
   Attributes: userId, username, password, email, phoneNumber, address, role.
   This class is used by UserService.java and UserDAO.java to create, authenticate, update and/or delete any information about any Admin, Member or Trainer. The role value used in the User class determines the menu access and available features for the person using the application.
2. UserDAO: This class handles all the database operations for the User objects. It can create, read, update and delete any information in the User class.
   Methods: addUser, getUserByUsername, getAllUsers, updateUser, deleteUserByUsername.
   The UserDAO is called by UserService.java to perform the operations listed above. It also uses a database connection to connect to the database in order to perform the necessary CRUD operations.
3. UserService: UserService is able to handle the user registration, authentication, and checking the role of the user who is signing up or signing in.
   Methods: registerUser, loginUser, deleteUserByUsername, displayAllUsers.
   This class interacts with UserDAO for database interaction. It also provides logic for signing in based on the role the user provides.

4. Membership: The membership class handles the membership plans available to the gym members and trainers. This class also handles the gym revenues, giving the Admins a total revenue from all gym memberships.
   Attributes: membershipId, membershipType, membershipDescription, membershipCost, memberId, purchaseDate.
   This class is referenced by Admins who wants to see membership information or revenue information, as well as Trainers and Users who want to see their membership information as well. It is connected to MembershipDAO and MembershipService to create, read, update and delete memberships.
5. MembershipDAO: handles all database operations for gym memberships.
   Methods: addMembership, getAllMemberships, getMembershipByMemberId, getTotalRevenue, getTotalSpentByMember, updateMembership, deleteMembership.
   The MembershipDAO is called by MembershipService.java to perform the operations listed above. It also uses the databse connection to connect to the database in order to perform the necessary CRUD operations.
6. MembershipService: MembershipService is used to provide the business logic required to manage all of the memberships for the gym.
   Methods: addMembership, displayAllMemberships, displayMembershipsByMemberId, displayTotalRevenue, displayTotalSpentByMember, updateMembership, deleteMembership.
   This service page is used by Admins to manage gym memberships and their cost. It calls MembershipDAO to properly enforce the CRUD operations.
7. WorkoutClass: This class holds all of the available workout classes for Members, and also shows the Trainers the classes that they are enrolled for.
   Attributes: workoutClassID, workoutClassType, workoutClassDescription, trainerID, schedule.
   WorkoutClass is used for Members and Trainers to be able to view classes by ID. This class interacts with WorkoutClassDAO and WorkoutClassService to provide all of the necessary class information.
8. WorkoutClassDAO: This handles the database connection for all workout classes and trainers in the classes. Provides CRUD operations for all classes.
   Methods: addWorkoutClass, getWorkoutClassByType, getWorkoutClassByTrainerId, updateWorkoutClass, deleteWorkoutClassById.
9. WorkoutClassService: This handles all business logic for workout classes to ensure that all workout classes are assigned to Trainers, and is capable of adding, deleting, or listing the workout classes.
   Methods: displayWorkoutClassesByTrainer, addWorkoutClass, updateWorkoutClass, deleteWorkoutClass.

This service page is used to manage all of the workout classes in the system, calling WorkoutClassDAO to perform the CRUD operations to create, read, update and delete any information required for a workout class.

Class Diagram



How to Start & Use the Program

To start this program, you must begin by cloning the repository. This can be done by opening the terminal in your preferred IDE (integrated development environment) such as VSCode or Intellij.

1. To open the terminal in VSCode, simply open VSCode and “Terminal” should appear in the top left of your screen. Click “Terminal”, and then “New Terminal”. Or Control + Shift + ` on Mac. If you’re using Intellij, open the terminal by pressing “View” at the top left of your screen, followed by “Tool Windows” and “Terminal”. Or press Option + F12 on Mac.

2. The next step is to clone your repository. To clone this repo, type “git clone <repo-link>” In your terminal. In this case, it would be “git clone https://github.com/ZCollier-dev/java-midterm-sprint-w2025”.
3. Navigate into the cloned repository to run the code using the **cd** command:
   cd src/main/java/org/sprint
4. Navigate to the GymApp.java file, and click “Run” towards the top of the code. This should produce the menu options.
5. Follow the on-screen instructions in order to register or log in to the system as an Admin, Member or Trainer.

Development Documentation

Javadoc Documentation

Javadocs are used in this program to describe the key methods and classes. In our code, we provided these comments to to describe the classes and what their purpose is, such as:

- UserService.java
  registerUser: Registers a new user after validating and hashing their password.
  loginUser: Authenticates the user login by checking their username and validating the hashed password.
  deleteUserByUsername: Deletes a user by their username (for Admin use only).
  DisplayAllUsers: Displays a list of all registered users (Admin only).
- MembershipService.java
  addMembership: Adds a new membership to the database.
  displayAllMemberships: Displays a list of all memberships.
  displayMembershipByMemberId: Displays a member’s information based on their ID.
  displayTotalRevenue: Displays the gym’s total revenue.
  displayTotalSpentByMember: Displays how much money the member has spent.
  updateMembership: Updates membership details.
  deleteMembership: Deletes a membership from the system.
- WorkoutClassService.java
  displayWorkoutClassesByTrainer: Displays a list of workout classes by trainer ID.
  addWorkoutClass: Adds a workout class with its description and scheduled time.
  updateWorkoutClass: Updates the information and schedule of a workout class.
  deleteWorkoutClass: Deletes a workout class by ID.
- UserDAO.java, MembershipDAO.java, WorkoutClassDAO.java

All of the DAO files perform the CRUD (Create, Read, Update, Delete) operations to interact with the PostgreSQL database.

Project Directory Structure

Our project is contained in a repository on GitHub called java-final-sprint-w2025. Inside of this folder, our sprint is located in a different folder called src/main/java/org/sprint. This folder contains our main app: GymApp.java, which contains the menu system that the program runs on, as well as the README, and other additional files required for running our application. This folder also contains 4 folders separated by subject: database, memberships, user, and workout classes.

The database folder contains the file for our PostgreSQL database connection.

The memberships folder contains all of our code and information regarding memberships: Membership.java, MembershipDAO.java, and MembershipService.java.

The user folder contains a separate folder for the child classes: Admin.java, Member.java and Trainer.java. These child classes connect to the other classes in the user file: User.java, UserDAO.java, and UserService.java.

The workout classes folder contains the WorkoutClass.java class, the WorkoutClassDAO.java class, and the WorkoutClassService.java class.

Build Process & Dependencies

This project was built using Maven to manage dependencies, and is connected to a PostgreSQL database. We used BCrypt (jbcrypt) to securely hash passwords.

Setting Up the Database

The database is an important component of this management system, as it sorts information on users, memberships, and workout classes. These are the steps used to set up the database for this project:

1. Download PostgreSQL: Visit this link https://www.postgresql.org to download the appropriate version for your computer.

2. Install to your computer: Follow the instructions to download the program. Take note of the following information:
   - Username
   - Password (will be necessary during installation)
3. Launch to ensure it is running correctly. After these steps are complete, you are ready to create a database.
4. Configure the database connection in your IDE. The username and password should match what you wrote down during installation:

```
public class DBConnection {
   private static final String URL = "jdbc:postgresql://localhost:5432/gym_management";
  private static final String USER = "your_username";
  private static final String PASSWORD = "your_password";
  public static Connection getConnection() throws SQLException {
      return DriverManager.getConnection(URL, USER, PASSWORD);
   }
}
```

How to Clone the Repository

To open the terminal in VSCode, simply open VSCode and “Terminal” should appear in the top left of your screen. Click “Terminal”, and then “New Terminal”. Or Control + Shift + ` on Mac. If you’re using Intellij, open the terminal by pressing “View” at the top left of your screen, followed by “Tool Windows” and “Terminal”. Or press Option + F12 on Mac.

To clone this repository, type “git clone <repo-link>” In your terminal. In this case, it would be “git clone https://github.com/ZCollier-dev/java-midterm-sprint-w2025”. The repository should populate in your IDE. To run the project, you must navigate to the GymApp.java file found in the lefthand Explorer of your IDE, and press “Run” towards the top of the code. This should open the application menu with various options to register, sign in, or update information.

Enjoy!