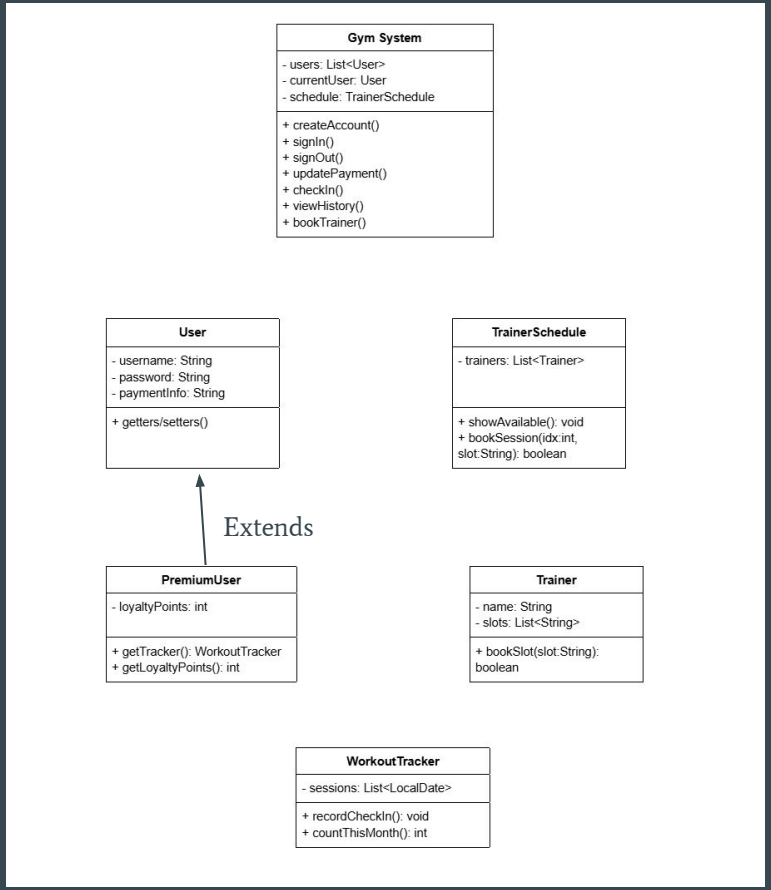
Gym Membership App

Olivia Bandurski, Madeline Brockman, Charlie Mixson, Zeb Stone, Ben Kim

UML Class Diagram

- Gym System is the core class that holds most methods to execute the menu options
- TrainerSchedule manages the Trainer class
- WorkoutTracker keeps count of how many teams each user worked out



User

Represents a gym member's account

- Uses private fields like username, password, paymentInfo, and workoutTracker

Methods

- getUsername() and getPassword for login cecks
- setPaymentInfo() and getPaymentInfo() to update and retrieve billing info
- getTracker() gives access to workout tracking

```
1  public class User {
2      private String username;
3      private String password;
4      private String paymentInfo;
5      private WorkoutTracker tracker;
6
7      public User(String username, String password) {
8          this.username = username;
9          this.password = password;
10         this.tracker = new WorkoutTracker();
11         this.paymentInfo = "Not Set";
12     }
13
14     public String getUsername() { return username; }
15     public String getPassword() { return password; }
16
17     public void setPaymentInfo(String info) {
18         this.paymentInfo = info;
19     }
20
21     public String getPaymentInfo() {
22         return paymentInfo;
23     }
24
25     public WorkoutTracker getTracker() {
26         return tracker;
27     }
28 }
29
```

GymSystem

Core Logic of the App

- Handles users, sign in, workouts, payments, and scheduling
- ArrayList<User> for storing all user accounts
- TrainerSchedule for managing trainer session
- CurrentUser keeps track of who is logged in

Functions

- Account creation, sign-in/out, deleting accounts
- Workout check-ins and viewing monthly workout history
- Updating payment info
- Displaying and booking training sessions

```java
import java.util.*;

public class GymSystem {
    private ArrayList<User> users;
    private User currentUser;
    private TrainerSchedule schedule;

    public GymSystem() {
        users = new ArrayList<>();
        schedule = new TrainerSchedule();
    }

    public void createAccount(String username, String password) {
        users.add(new User(username, password));
        System.out.println("Account created successfully.");
    }

    public void signIn(String username, String password) {
        for (User u : users) {
            if (u.getUsername().equals(username) && u.getPassword().equals(password)) {
                currentUser = u;
                System.out.println("Signed in successfully.");
                return;
            }
        }
        System.out.println("Invalid credentials.");
    }

    public void signOut() {
        currentUser = null;
        System.out.println("Signed out.");
    }

    public void deleteAccount(String username) {
        users.removeIf(u -> u.getUsername().equals(username));
        System.out.println("Account deleted.");
    }

    public boolean isSignedIn() {
        return currentUser != null;
    }

    public void updatePayment(String info) {
        currentUser.setPaymentInfo(info);
        System.out.println("Payment info updated.");
    }

    public void checkIn() {
        currentUser.getTracker().checkIn();
    }

    public void viewWorkoutHistory() {
        int count = currentUser.getTracker().getMonthlyWorkouts();
        System.out.println("You have worked out " + count + " times this month.");
    }

    public void scheduleTrainer(int index, String slot) {
        schedule.bookTrainer(index, slot);
    }

    public void showTrainerSchedule() {
        schedule.displayTrainers();
    }
    public void createPremiumAccount(String username, String password) {
        users.add(new PremiumUser(username, password));
        System.out.println("Premium account created successfully.");
    }
}
```

WorkoutTracker

Tracks when a user checks into the gym.

- Uses an ArrayList<LocateDate> to store each check-in

Methods

- CheckIn() adds today's date to the list (like scanning a QR code)
- getMonthlyWorkouts() counts how many times the user worked out this month

```java

import java.time.LocalDate;
import java.util.ArrayList;

public class WorkoutTracker {
    private ArrayList<LocalDate> checkIns;

    public WorkoutTracker() {
        checkIns = new ArrayList<>();
    }

    public void checkIn() {
        checkIns.add(LocalDate.now());
        System.out.println("Check-in successful. QR Code scanned.");
    }

    public int getMonthlyWorkouts() {
        LocalDate now = LocalDate.now();
        return (int) checkIns.stream()
            .filter(d -> d.getMonth() == now.getMonth() && d.getYear() == now.getYear())
            .count();
    }
}

```

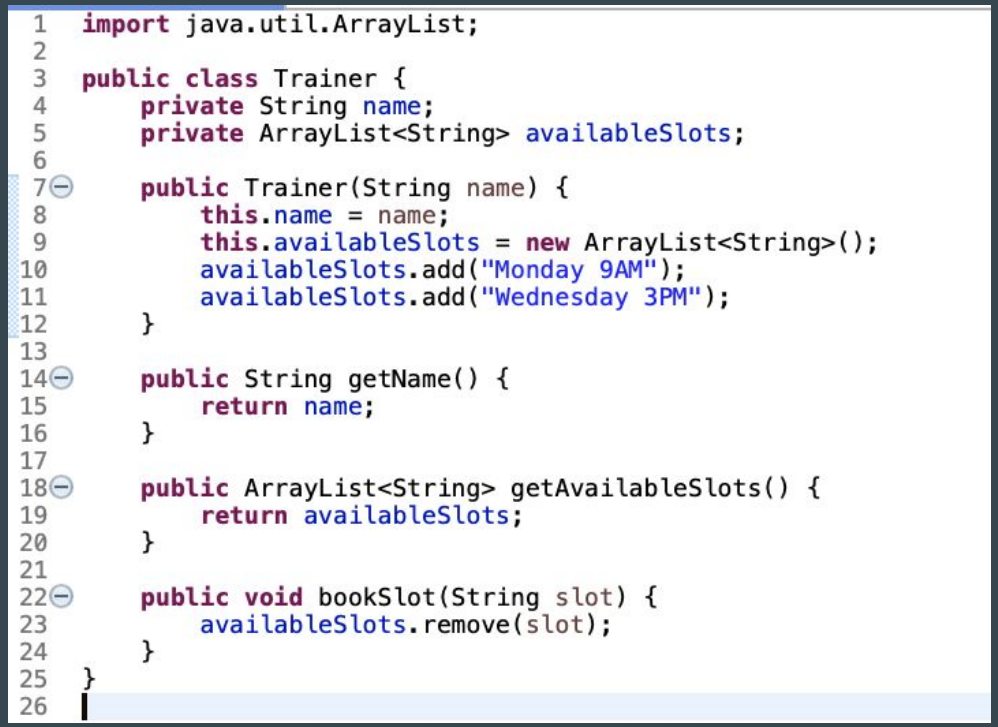
Trainer

Represents a trainer and their availability

- Each trainer has a name and a list of available time slots

Methods

- getName() and getAvailableSlots() to display trainer information
- bookSlot() removes a slot from their availability when booked



```java
import java.util.ArrayList;

public class Trainer {
    private String name;
    private ArrayList<String> availableSlots;

    public Trainer(String name) {
        this.name = name;
        this.availableSlots = new ArrayList<String>();
        availableSlots.add("Monday 9AM");
        availableSlots.add("Wednesday 3PM");
    }

    public String getName() {
        return name;
    }

    public ArrayList<String> getAvailableSlots() {
        return availableSlots;
    }

    public void bookSlot(String slot) {
        availableSlots.remove(slot);
    }
}
```

TrainerSchedule

Manages a list of trainers and lets users schedule sessions

- Holds an ArrayList of Trainer objects

Methods

- displayTrainers() shows each trainer and their open slots
- bookTrainer() books a session by removing a slot from a selected trainer

```java
import java.util.ArrayList;

public class TrainerSchedule {
    private ArrayList<Trainer> trainers;

    public TrainerSchedule() {
        trainers = new ArrayList<>();
        trainers.add(new Trainer("Alice"));
        trainers.add(new Trainer("Bob"));
    }

    public void displayTrainers() {
        int idx = 1;
        for (Trainer t : trainers) {
            System.out.println(idx + ". " + t.getName() + " - " + t.getAvailableSlots());
            idx++;
        }
    }

    public void bookTrainer(int trainerIndex, String slot) {
        Trainer trainer = trainers.get(trainerIndex - 1);
        if (trainer.getAvailableSlots().contains(slot)) {
            trainer.bookSlot(slot);
            System.out.println("Session booked with " + trainer.getName() + " at " + slot);
        } else {
            System.out.println("Slot unavailable.");
        }
    }
}
```

PremiumUser

Extends the User class to inherit basic account features

- overrides check-in behavior to reward loyalty points
- loyaltyPoints tracks the number of points earned from check-ins

Method

- getTracker() returns a customized workout tracker that adds loyalty points on each check-in
- getLoyaltyPoints() returns the current total of loyalty points

```
1
2  public class PremiumUser extends User {
3      private int loyaltyPoints;
4
5      public PremiumUser(String username, String password) {
6          super(username, password);
7          this.loyaltyPoints = 0;
8      }
9
10     @Override
11     public WorkoutTracker getTracker() {
12         // Runtime polymorphism: overrides how tracker behaves by adding points
13         return new WorkoutTracker() {
14             @Override
15             public void checkIn() {
16                 super.checkIn();
17                 loyaltyPoints += 10;
18                 System.out.println("You've earned 10 loyalty points! Total: " + loyaltyPoints);
19             }
20         };
21     }
22
23     public int getLoyaltyPoints() {
24         return loyaltyPoints;
25     }
26 }
27
```

Main

The menu system for interacting with the app

- While loop to keep the app running
- First menu appears if no one is signed in (Sign/Create/Delete/Quit)
- Second menu appears when signed in (Workout, Payment, Trainer bookings, Check-in, Sign out)
- Interaction: Uses scanner to get input from the user and calls GymSystem method based on menu choices

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        GymSystem system = new GymSystem();
        Scanner sc = new Scanner(System.in);
        String choice;

        while (true) {
            if (!system.isSignedIn()) {
                System.out.println("A - Sign In\nB - Create Account\nC - Delete Account\nD - Quit\n\nChoose an option:");

                choice = sc.nextLine().toUpperCase();
                switch (choice) {
                    case "A":
                        System.out.print("Username: ");
                        String user = sc.nextLine();
                        System.out.print("Password: ");
                        String pass = sc.nextLine();
                        system.signIn(user, pass);
                        break;
                    case "B":
                        System.out.print("New Username: ");
                        user = sc.nextLine();
                        System.out.print("New Password: ");
                        pass = sc.nextLine();
                        system.createAccount(user, pass);
                        break;
                    case "C":
                        System.out.print("Username to delete: ");
                        user = sc.nextLine();
                        system.deleteAccount(user);
                        break;
                    case "D":
                        System.out.println("Goodbye!");
                        return;
                }
            } else {
                System.out.println("\nA - Update Payment Info\nB - Workout History\nC - Schedule Trainer\nD - Check In\nE - Sign Out\n\nChoose an option:");
                choice = sc.nextLine().toUpperCase();
                switch (choice) {
                    case "A":
                        System.out.print("Enter new payment info: ");
                        String info = sc.nextLine();
                        system.updatePayment(info);
                        break;
                    case "B":
                        system.viewWorkoutHistory();
                        break;
                    case "C":
                        system.showTrainerSchedule();
                        System.out.print("Select trainer number: ");
                        int idx = Integer.parseInt(sc.nextLine());
                        System.out.print("Enter time slot: ");
                        String slot = sc.nextLine();
                        system.scheduleTrainer(idx, slot);
                        break;
                    case "D":
                        system.checkIn();
                        break;
                    case "E":
                        system.signOut();
                        break;
                }
            }
        }
    }
}
```

OOP Principles

- **Encapsulation:** For example, the User class keeps username and password private, as well as the Trainer class keeping availableSlots private, etc. while allowing access to these variables when a getter/setter method(s) is called.
- **Inheritance:** The PremiumUser subclass inherits the User class's variables and methods while also giving future programmers the ability to add their own features as seen fit.
- **Polymorphism:** If a user has a premium membership, the check-in method is overwritten to add loyalty points if a user checks in.

Applicability in Business Context

Although self-explanatory, the gym membership program would serve as a great foundation for both private and commercial gyms alike to build upon.

- Provides an all-in-one place for members to access a wide variety of services; added benefit of **convenience** and overall customer satisfaction.
- **Streamlines** otherwise manual/lengthy processes such as user check-ins, member registration, etc.
- **Scalability** (especially for commercial gyms).
- **Data tracking benefits** (ex: Can track drop-off(s) in overall check-ins during certain times of the year, increase staff during high member traffic)
- Allows for easier implementation of **customer retention strategies** such as loyalty rewards.

Potential Future Work

Purchase Drink/Snack:

- Many gyms either feature vending machines or offer snacks/drinks (ex: protein bars, shakes, sports drinks, etc.) within their facilities.
- As a result, linking a payment method(s) to a user would be greatly beneficial in terms of convenience on the customers' end.
- Members would simply input their preferred payment method, and use their QR code, when prompted by staff, to be charged for the snack/drink.

Loyalty Rewards:

- Users can earn points for activities (check-ins, trainer bookings, purchases) and redeem them for rewards like discounts, free items, and eventual membership upgrades.
- Will expand upon our premium user so users will get access to perks like unlimited visits per month, access to facilities, priority in trainer session sign-ups
- Users will select "Loyalty Rewards" option from the menu to view their points balance, see available rewards, and can redeem their points for selected rewards.
- Encourages member retention and engagement by rewarding frequent gym use, increasing revenue through more visits and purchases.