import java.io.\*;

import java.util.\*;

import java.time.LocalDate;

// Abstract Person class

abstract class Person {

private String name;

private int age;

private double weight;

private double height;

public Person(String name, int age, double weight, double height) {

this.name = name;

this.age = age;

this.weight = weight;

this.height = height;

}

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public int getAge() { return age; }

public void setAge(int age) { this.age = age; }

public double getWeight() { return weight; }

public void setWeight(double weight) { this.weight = weight; }

public double getHeight() { return height; }

public void setHeight(double height) { this.height = height; }

public abstract void provideRecommendations();

}

// Goal class for fitness goals

class Goal implements Serializable {

private String category; // Added category field

private String description;

private double targetValue;

private double currentValue;

private LocalDate deadline;

public Goal(String category, String description, double targetValue, LocalDate deadline) {

this.category = category;

this.description = description;

this.targetValue = targetValue;

this.currentValue = 0;

this.deadline = deadline;

}

public String getCategory() { return category; }

public String getDescription() {

return description;

}

public void updateProgress(double progress) {

currentValue += progress;

if (currentValue > targetValue) {

currentValue = targetValue;

}

}

public boolean isGoalCompleted() {

return currentValue >= targetValue;

}

public void displayProgress() {

double percentage = (currentValue / targetValue) \* 100;

System.out.println("[" + category + "] Goal: " + description + " | Deadline: " +

(deadline != null ? deadline : "None"));

System.out.println("Progress: " + (int) percentage + "%");

System.out.print("[");

int completed = (int) (percentage / 10);

for (int i = 0; i < 10; i++) {

System.out.print(i < completed ? "=" : " ");

}

System.out.println("]");

if (isGoalCompleted()) {

System.out.println("🎉 Goal achieved!");

}

}

}

// User class inherits Person class

class User extends Person implements Serializable {

private static final long serialVersionUID = 1L;

private static int idCounter = 1001;

private int userId;

private List<Goal> fitnessGoals;

private List<String> workoutRoutine;

private List<String> nutritionLog;

private double totalCaloriesBurned;

private double dailyWaterIntake;

private double sleepHours;

private List<String> moodLog;

private double totalWorkoutTime;

private Map<String, List<String>> activityLog;

private static final double WATER\_GOAL = 2.0;

public User(String name, int age, double weight, double height) {

super(name, age, weight, height);

this.userId = idCounter++;

this.fitnessGoals = new ArrayList<>();

this.workoutRoutine = new ArrayList<>();

this.nutritionLog = new ArrayList<>();

this.moodLog = new ArrayList<>();

this.activityLog = new HashMap<>();

this.totalCaloriesBurned = 0;

this.dailyWaterIntake = 0;

this.totalWorkoutTime = 0;

this.sleepHours = 0;

}

public int getUserId() { return userId; }

public void addGoal(String category, String description, double targetValue, LocalDate deadline) {

fitnessGoals.add(new Goal(category, description, targetValue, deadline));

System.out.println("Goal added successfully under category: " + category);

}

public void logCaloriesBurned(double calories) {

totalCaloriesBurned += calories;

for (Goal goal : fitnessGoals) {

if (goal.getCategory().equalsIgnoreCase("Calories")) {

goal.updateProgress(calories);

}

}

}

public void addWorkout(String workout) {

workoutRoutine.add(workout);

}

public void viewWorkoutRoutine() {

System.out.println("\n--- Workout Routine ---");

if (workoutRoutine.isEmpty()) {

System.out.println("No workouts added yet!");

} else {

for (String workout : workoutRoutine) {

System.out.println("- " + workout);

}

}

}

public void addNutrition(String foodItem) {

nutritionLog.add(foodItem);

}

public void viewNutritionLog() {

System.out.println("\n--- Nutrition Log ---");

if (nutritionLog.isEmpty()) {

System.out.println("No food items logged yet!");

} else {

for (String item : nutritionLog) {

System.out.println("- " + item);

}

}

}

public void viewGoals() {

System.out.println("\n--- Fitness Goals ---");

if (fitnessGoals.isEmpty()) {

System.out.println("No goals set yet!");

} else {

Map<String, List<Goal>> categorizedGoals = new HashMap<>();

for (Goal goal : fitnessGoals) {

categorizedGoals.putIfAbsent(goal.getCategory(), new ArrayList<>());

categorizedGoals.get(goal.getCategory()).add(goal);

}

categorizedGoals.forEach((category, goals) -> {

System.out.println("Category: " + category);

goals.forEach(Goal::displayProgress);

});

}

}

public double calculateBMI() {

double heightInMeters = getHeight() / 100;

return getWeight() / (heightInMeters \* heightInMeters);

}

public void logWaterIntake(double liters) {

dailyWaterIntake += liters;

if (dailyWaterIntake > WATER\_GOAL) {

dailyWaterIntake = WATER\_GOAL;

}

System.out.printf("Logged %.2f liters of water. Total: %.2f/%.2f liters.\n", liters, dailyWaterIntake, WATER\_GOAL);

}

public void viewWaterIntake() {

double percentage = (dailyWaterIntake / WATER\_GOAL) \* 100;

System.out.printf("\nWater Intake: %.2f/%.2f liters (%.0f%%)\n", dailyWaterIntake, WATER\_GOAL, percentage);

System.out.print("[");

int completed = (int) (percentage / 10);

for (int i = 0; i < 10; i++) {

System.out.print(i < completed ? "=" : " ");

}

System.out.println("]");

}

public void logSleepHours(double hours) {

sleepHours += hours;

for (Goal goal : fitnessGoals) {

if (goal.getCategory().equalsIgnoreCase("Sleep Hours")) {

goal.updateProgress(hours);

}

}

System.out.printf("Logged %.2f hours of sleep. Total: %.2f hours.\n", hours, sleepHours);

}

public void viewSleepHours() {

System.out.printf("\nTotal Sleep Hours: %.2f\n", sleepHours);

if (sleepHours < 7) {

System.out.println("Recommendation: Try to get at least 7-8 hours of sleep per night.");

} else {

System.out.println("Great! You're getting enough rest.");

}

}

public void logWorkoutTime(double minutes) {

totalWorkoutTime += minutes;

for (Goal goal : fitnessGoals) {

if (goal.getCategory().equalsIgnoreCase("Workout Time")) {

goal.updateProgress(minutes);

}

}

System.out.printf("Logged %.2f minutes of workout. Total: %.2f minutes.\n", minutes, totalWorkoutTime);

}

public void viewWorkoutTime() {

System.out.printf("\nTotal Workout Time: %.2f minutes\n", totalWorkoutTime);

if (totalWorkoutTime < 150) {

System.out.println("Recommendation: Aim for at least 150 minutes of moderate exercise per week.");

} else {

System.out.println("Great! You're meeting or exceeding workout recommendations.");

}

}

public void logMood(String mood) {

moodLog.add(mood);

System.out.println("Mood logged: " + mood);

}

public void viewMoodLog() {

System.out.println("\n--- Mood Log ---");

if (moodLog.isEmpty()) {

System.out.println("No moods logged yet.");

} else {

for (String mood : moodLog) {

System.out.println("- " + mood);

}

}

}

public void logActivity(String category, String activity) {

activityLog.putIfAbsent(category, new ArrayList<>());

activityLog.get(category).add(activity);

System.out.printf("Logged activity: %s under %s.\n", activity, category);

}

public void viewActivityLog() {

System.out.println("\n--- Activity Log ---");

if (activityLog.isEmpty()) {

System.out.println("No activities logged yet.");

} else {

activityLog.forEach((category, activities) -> {

System.out.println(category + ":");

for (String activity : activities) {

System.out.println(" - " + activity);

}

});

}

}

@Override

public void provideRecommendations() {

System.out.println("\n--- Recommendations ---");

for (Goal goal : fitnessGoals) {

if (!goal.isGoalCompleted()) {

switch (goal.getCategory()) {

case "Calories" -> System.out.println("Focus on burning more calories. Try cardio or HIIT workouts.");

case "Sleep Hours" -> System.out.println("Improve sleep hygiene. Aim for 7-8 hours of sleep.");

case "Workout Time" -> System.out.println("Add strength training or longer workout sessions.");

default -> System.out.println("Keep pushing towards your goals!");

}

} else {

System.out.println("Well done on completing your " + goal.getCategory() + " goal: " + goal.getDescription());

}

}

}

}

public class FitnessTrackerApp {

private static Map<Integer, User> userDatabase = new HashMap<>();

private static Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

System.out.println("Welcome to the Fitness Tracker!");

while (true) {

System.out.println("\n--- Main Menu ---");

System.out.println("1. Create new user");

System.out.println("2. Log in with User ID");

System.out.println("3. Exit");

System.out.print("Choose an option: ");

int mainChoice = scanner.nextInt();

switch (mainChoice) {

case 1 -> createNewUser();

case 2 -> loginUser();

case 3 -> {

System.out.println("Exiting Fitness Tracker. Stay healthy!");

return;

}

default -> System.out.println("Invalid choice. Please try again.");

}

}

}

private static void createNewUser() {

scanner.nextLine(); // Consume leftover newline

System.out.print("Enter your name: ");

String name = scanner.nextLine();

System.out.print("Enter your age: ");

int age = scanner.nextInt();

System.out.print("Enter your weight (kg): ");

double weight = scanner.nextDouble();

System.out.print("Enter your height (cm): ");

double height = scanner.nextDouble();

User newUser = new User(name, age, weight, height);

userDatabase.put(newUser.getUserId(), newUser);

System.out.println("User created successfully!");

System.out.println("Your User ID is: " + newUser.getUserId());

}

private static void loginUser() {

System.out.print("Enter your User ID: ");

int userId = scanner.nextInt();

User loggedInUser = userDatabase.get(userId);

if (loggedInUser == null) {

System.out.println("Invalid User ID. Please try again.");

} else {

System.out.println("Welcome back, " + loggedInUser.getName() + "!");

runFitnessTracker(loggedInUser);

}

}

private static void runFitnessTracker(User user) {

while (true) {

System.out.println("\n--- Fitness Tracker Menu ---");

System.out.println("1. Add a Fitness Goal");

System.out.println("2. View Fitness Goals");

System.out.println("3. Log Calories Burned");

System.out.println("4. Add Workout to Routine");

System.out.println("5. View Workout Routine");

System.out.println("6. Log Nutrition");

System.out.println("7. View Nutrition Log");

System.out.println("8. Log Water Intake");

System.out.println("9. View Water Intake");

System.out.println("10. Log Sleep Hours");

System.out.println("11. View Sleep Hours");

System.out.println("12. Log Mood");

System.out.println("13. View Mood Log");

System.out.println("14. Log Activity");

System.out.println("15. View Activity Log");

System.out.println("16. View BMI");

System.out.println("17. Recommendations");

System.out.println("18. Log WorkoutTime");

System.out.println("19. View WorkoutTime Log");

System.out.println("0. Log out");

System.out.print("Choose an option: ");

int choice = scanner.nextInt();

switch (choice) {

case 1 -> {

scanner.nextLine();

System.out.println("Choose a category for the goal:");

System.out.println("1. Calories");

System.out.println("2. Sleep Hours");

System.out.println("3. Workout Time");

System.out.println("4. Other");

System.out.print("Enter your choice: ");

int categoryChoice = scanner.nextInt();

scanner.nextLine();

String category = switch (categoryChoice) {

case 1 -> "Calories";

case 2 -> "Sleep Hours";

case 3 -> "Workout Time";

default -> "Other";

};

System.out.print("Enter goal description: ");

String description = scanner.nextLine();

System.out.print("Enter target value: ");

double targetValue = scanner.nextDouble();

System.out.print("Enter deadline (yyyy-mm-dd) or leave blank: ");

scanner.nextLine();

String deadlineInput = scanner.nextLine();

LocalDate deadline = deadlineInput.isEmpty() ? null : LocalDate.parse(deadlineInput);

user.addGoal(category, description, targetValue, deadline);

}

case 2 -> user.viewGoals();

case 3 -> {

System.out.print("Enter calories burned: ");

double calories = scanner.nextDouble();

user.logCaloriesBurned(calories);

}

case 4 -> {

scanner.nextLine();

System.out.print("Enter workout: ");

String workout = scanner.nextLine();

user.addWorkout(workout);

}

case 5 -> user.viewWorkoutRoutine();

case 6 -> {

scanner.nextLine();

System.out.print("Enter food item: ");

String foodItem = scanner.nextLine();

user.addNutrition(foodItem);

}

case 7 -> user.viewNutritionLog();

case 8 -> {

System.out.print("Enter water intake (liters): ");

double liters = scanner.nextDouble();

user.logWaterIntake(liters);

}

case 9 -> user.viewWaterIntake();

case 10 -> {

System.out.print("Enter sleep hours: ");

double hours = scanner.nextDouble();

user.logSleepHours(hours);

}

case 11 -> user.viewSleepHours();

case 12 -> {

scanner.nextLine();

System.out.print("Enter mood: ");

String mood = scanner.nextLine();

user.logMood(mood);

}

case 13 -> user.viewMoodLog();

case 14 -> {

scanner.nextLine();

System.out.print("Enter activity category: ");

String activityCategory = scanner.nextLine();

System.out.print("Enter activity: ");

String activity = scanner.nextLine();

user.logActivity(activityCategory, activity);

}

case 15 -> user.viewActivityLog();

case 16 -> {

double bmi = user.calculateBMI();

System.out.printf("Your BMI is: %.2f\n", bmi);

}

case 17 -> user.provideRecommendations();

case 18 -> {

System.out.print("Enter workout duration (in minutes): ");

double minutes = scanner.nextDouble();

user.logWorkoutTime(minutes);

}

case 19 -> user.viewWorkoutTime();

case 0 -> {

System.out.println("Logging out...");

return;

}

default -> System.out.println("Invalid choice. Please try again.");

}

}

}

}