National College of Ireland

Academic Year 2024/2025

Adrián Tarín Martínez

X23388978

X23388978@student.ncirl.ie

Software Development CA

Content

Program Overview

The FitnessApp application is a program developed in the Java programming language that allows users to receive advice and precautions regarding sports activities based on the input parameters. Interaction between the user and the application is carried out through the console.

The application prompts the user to enter several data points, which are validated to ensure they meet the required data type and fall within an allowed range. For example, the user must enter a humidity percentage, which must be an integer (int) between 0 and 100.

The application collects meteorological data, including temperature, humidity, wind speed, UV index, and whether there is heavy rainfall. Based on this information, the user can obtain recommendations for suitable exercises and precautions to consider according to the weather conditions. Additionally, the user has access to various options, such as receiving advice on sports activities to practice, calculating the heat index (Heat Index), obtaining recommendations based on wind conditions, or receiving precautions related to UV radiation levels.

Furthermore, the application allows the user to calculate the number of calories burned. To do this, it requires information about the user's body weight, exercise duration, and the selection of a sport from the options provided.

The user can interact with the application indefinitely. They also have the option to select "Reset Application", which clears previously saved data, or "Exit Application" to terminate the program. In the latter case, the user will be asked if they wish to analyze another set of data. If they respond affirmatively, the application will restart. If they respond negatively, the program will end after displaying a farewell message.

Use Case Diagram

To schematically represent the various functions of the application, how they relate to each other, and how the user interacts with the program, a use case diagram of the application is created. This diagram provides an overview of the program's functionality and allows for structured planning of its development, enabling work on each part separately.

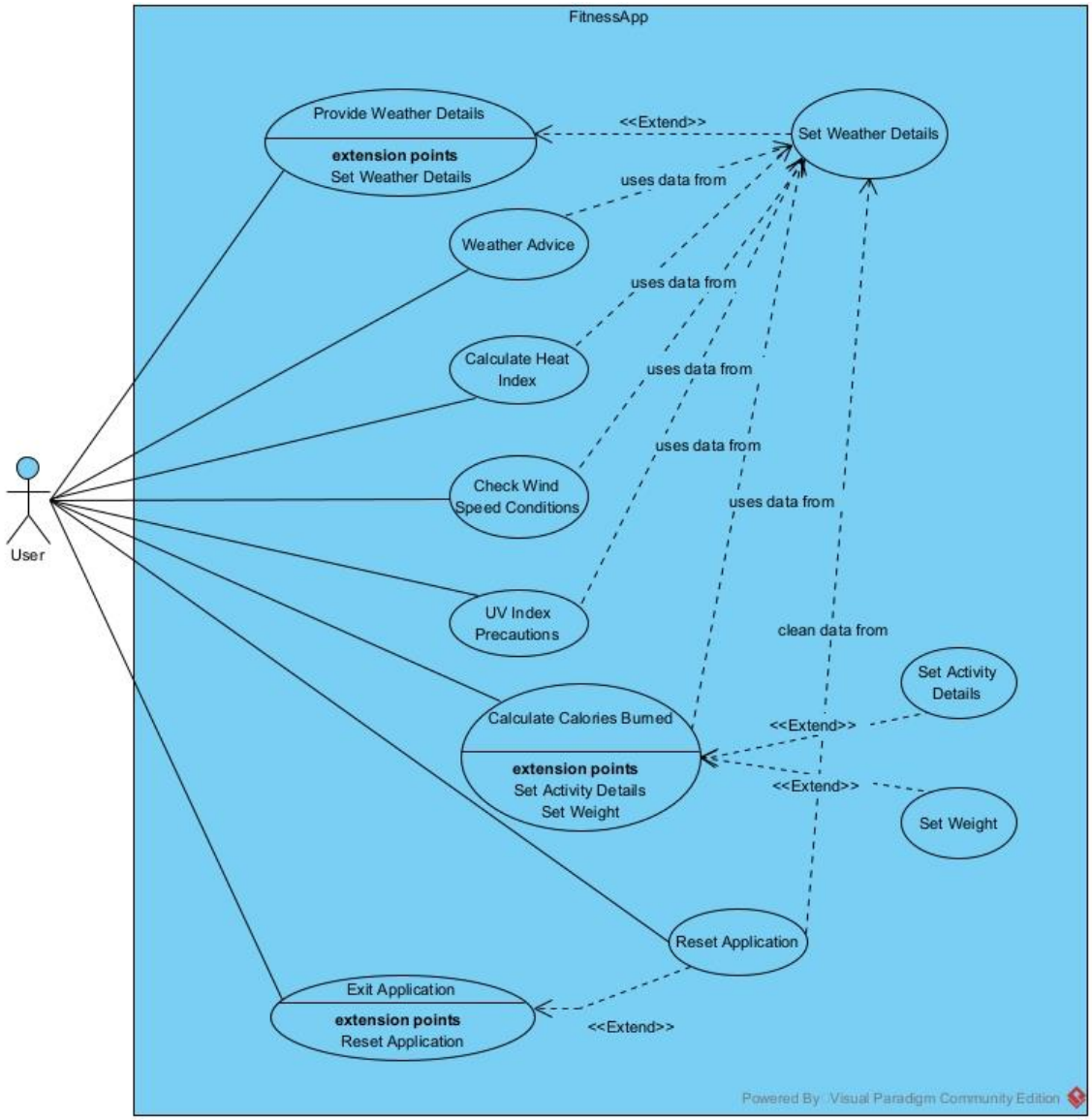

Figure 1. Use Case Diagram

Input, Processing and Output (IPO)

The Input-Processing-Output (IPO) cycle is a fundamental concept in software development, aimed at describing how data is received, processed, and delivered as output.

INPUT

The program takes various types of input from the user to perform different fitness and weather-related computations.

User Inputs:

1. **Menu Selection**:
    - A numerical choice (1-8) to select the desired operation from the application menu.

2. **Weather Details**:
    - **Temperature**: Integer, range from -20°C to 50°C.
    - **Humidity**: Integer, range from 0% to 100%.
    - **Wind Speed**: Integer, range from 0 to 50 m/s.
    - **UV Index**: Integer, range from 0 to 11.
    - **Rain**: String, either "yes" or "no".
3. **Activity Details** (for calculating calories burned):
    - **Activity Type**: Integer, mapped to specific activities (e.g., jogging, cycling).
    - **Duration**: Double, representing hours of exercise.
4. **Weight**:
    - Double, representing the user's weight in kilograms (range: 30 kg to 200 kg).
5. **Reset Confirmation**:
    - String, either "yes" or "no" to reset the application.
6. **Exit Confirmation:**
    - String, either "yes" or "no" to reset the application.

PROCESSING

The program executes various computations and decisions based on the user inputs.

Core Processes:

1. **Weather Details Validation**:
    - The program validates each weather-related input against predefined ranges and re-prompts the user in case of invalid input.
2. **Weather Advice**:
    - Based on temperature and rain conditions, the program provides advice on suitable outdoor activities.
3. **Heat Index Calculation**:
    - Computes the heat index using temperature and humidity values to evaluate discomfort levels.
4. **Wind Speed Conditions**:
    - Evaluates the wind speed to recommend suitable outdoor activities or safety precautions.
5. **UV Index Precautions**:
    - Suggests safety measures based on the UV index level (e.g., sunscreen, avoiding peak hours).

6. **Calories Burned Calculation**:
    - Estimates calories burned based on the user's weight, activity type, and duration.
7. **Application Reset**:
    - Clears all stored data and allows the user to start over.
8. **Exit Handling**:
    - Exits the application after confirming whether the user wants to check another set of conditions.

Error Handling:

- Extensive use of try-catch blocks ensures the program handles invalid inputs.
- Re-prompts the user in case of exceptions or out-of-range values.

OUTPUT

The program provides outputs in the form of recommendations, calculations, and confirmations.

Outputs:

1. **Confirmation Messages**:
    - E.g., "Data has been successfully saved," "Application reset successfully."
2. **Weather Advice**:
    - E.g., "Recommended outdoor activities such as running, cycling, or tennis."
3. **Heat Index**:
    - Displays the computed heat index and display advice if it is appropriated.
4. **Wind Speed Advice**:
    - E.g., "Avoid activities like cycling or running in strong winds."
5. **UV Index Precautions**:
    - E.g., "Apply sunscreen and avoid direct sunlight during peak hours."
6. **Calories Burned**:
    - Displays the estimated calories burned during the selected activity.
7. **Error Messages**:
    - For invalid inputs or menu selections.
8. **Exit Message**:
    - "Thank you for using FitnessApp. Goodbye."

Class Diagram

The following class diagram visually represents the structure of the classes used in the application, including their attributes and methods. The **FitnessApp** class manages the front-end of the application, handling user interactions while the **Fitness** class serves as the back-end, responsible for executing the required processes and calculations.

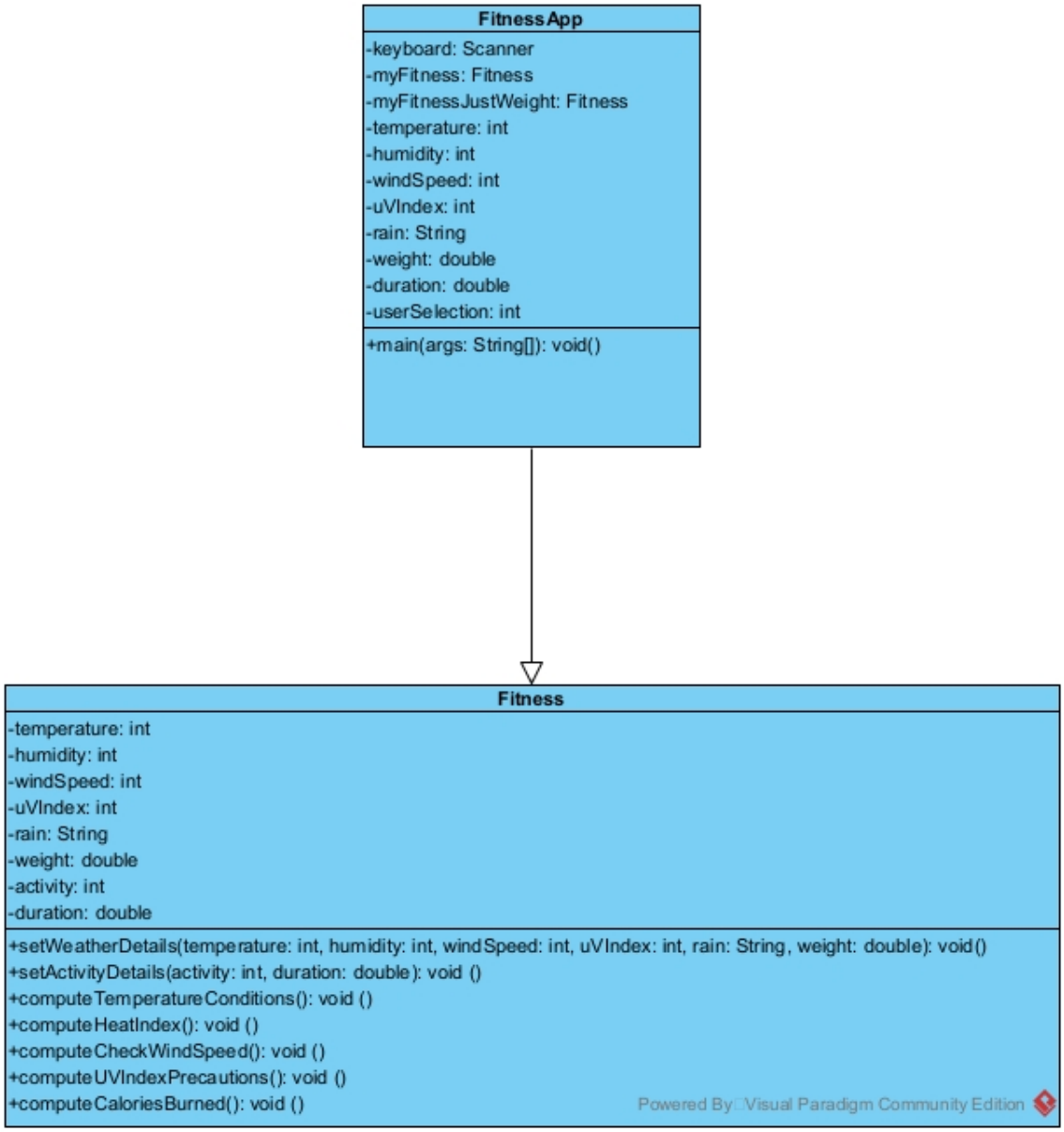

Figure 2. Class Diagram

Design and Developing Process

In the design of the application, the decision was made to separate the logic of the processes into two classes, each in a separate file. On one hand, the FitnessApp class handles the front-end, managing user interaction. On the other hand, the Fitness class performs the necessary calculations, acting as the back-end.

Upon starting the application, the user is presented with a menu containing eight options to choose from. In addition to the options specified originally in the guidelines, two more options were added to enrich the user experience and provide more functionality.

```
C:\Users\Usuario\Desktop\FitnessApp.java>java FitnessApp

Application Menu

1.      Provide weather details (Temperature, Humidity, Wind Speed, UV Index and rain)
2.      Weather Advice (Temperature and rain conditions)
3.      Calculate Heat Index
4.      Check Wind Speed conditions
5.      UV Index precautions
6.      Calculate the number of calories burned for an activity
7.      Reset Application
8.      Exit Application

Enter your choice
```

The user must select a valid option; otherwise, the program will display an error message and prompt the user to make a new selection, showing the available options again. This functionality is implemented using a loop that remains active until the user selects the option to exit the application, combined with error handling managed through try-catch statements. This ensures that the user is redirected both when selecting a number outside the available options and when entering any invalid data.

```
while (true){
    try {
        //Show Application Menu with options to the user
        System.out.println(applicationMenu);
        userSelection = keyboard.nextInt();
        //User selection: 1. Provide Weather Details
        if (userSelection == 1){

    //first try catch
    } catch (Exception e) {
        System.out.println("Selection not valid. Please enter a valid choice.");
        keyboard.next();
        }
```

In these code snippets, the functionality of asking the user to select an option is implemented. The "try" statement prompts the program to attempt processing the user's input. If the input is of a type that causes an error, such as a letter or symbol, the "catch" exception will be triggered, informing the user that they must make a valid selection. Then, the "keyboard" object's data is cleared so that it can be reused when the loop starts again.

```
            System.out.println("Application reset successfully. All data has been cleared.");
        //User selection: 8. Exit Application
        } else if (userSelection == 8){
            System.out.println("Would you like to check another set of conditions?");
            checkAnotherSet = keyboard.next();
            if (checkAnotherSet.equals("yes")){
                myFitness = null;
                myFitnessJustWeight = null;
                System.out.println("Application reset successfully. All data has been cleared.");
            } else {
                break;
            }
        //User selection is valid data type but not an option between 1-8
        } else {
            System.out.println("Selection not valid. Please enter a valid choice.");
        }

    //User input is not an int
    } catch (Exception e) {
        System.out.println("Selection not valid. Please enter a valid choice.");
        keyboard.next();
        }
```

If the user enters a valid data type, such as an int, but it is not one of the available options, the last block of the if statement will be triggered, informing the user once more that an invalid selection was made before the main while loop restarts, showing the menu options to the user again.

```
Command Prompt - java  FitnessApp
Java HotSpot(TM) 64-Bit Server VM (build 22.0.2+9-70, mixed mode, sharing)

C:\Users\Usuario>cd C:\Users\Usuario\Desktop\FitnessApp.java

C:\Users\Usuario\Desktop\FitnessApp.java>javac FitnessApp.java

C:\Users\Usuario\Desktop\FitnessApp.java>java FitnessApp

Application Menu

1.      Provide weather details (Temperature, Humidity, Wind Speed, UV Index and rain)
2.      Weather Advice (Temperature and rain conditions)
3.      Calculate Heat Index
4.      Check Wind Speed conditions
5.      UV Index precautions
6.      Calculate the number of calories burned for an activity
7.      Reset Application
8.      Exit Application

Enter your choice


x
Selection not valid. Please enter a valid choice.

Application Menu

1.      Provide weather details (Temperature, Humidity, Wind Speed, UV Index and rain)
2.      Weather Advice (Temperature and rain conditions)
3.      Calculate Heat Index
4.      Check Wind Speed conditions
5.      UV Index precautions
6.      Calculate the number of calories burned for an activity
7.      Reset Application
8.      Exit Application

Enter your choice
```

Similarly, if the user selects any option other than entering the data (options 2 through 5), they will be reminded that they must first input the weather details.

```
Enter your choice


2
Please enter weather details.

Application Menu

1.      Provide weather details (Temperature, Humidity, Wind Speed, UV Index and rain)
2.      Weather Advice (Temperature and rain conditions)
3.      Calculate Heat Index
4.      Check Wind Speed conditions
5.      UV Index precautions
6.      Calculate the number of calories burned for an activity
7.      Reset Application
8.      Exit Application

Enter your choice
```

This functionality is achieved by checking whether the user has already entered the weather details using if-else statements. Initially, the program sets the `myFitness` object to `null`. Only after the user has entered the data correctly will the `myFitness` object be created, and the program will then allow the user to check the remaining options. The same process applies to options 2, 3, 4, and 5.

```
        }
        //Create Fitness object and set details
        myFitness = new Fitness();
        myFitness.setWeatherDetails(temperature, humidity, windSpeed, uVIndex, rain, weight);
        System.out.println("Data has been succesfully saved.");

    //User selection: 2. Weather Advice (temperature and rain conditions)
    } else if (userSelection == 2){
        if (myFitness != null){
            myFitness.computeTemperatureConditions();
        } else {
            System.out.println("Please enter weather details.");
        }
    //User selection: 3. Calculate Heat Index
    } else if (userSelection == 3){
        if (myFitness != null){
            myFitness.computeHeatIndex();
        } else {
            System.out.println("Please enter weather details.");
        }
```

For option 6, which calculates the number of calories burned, the user must always provide the exercise duration and select a sport from the list offered by the application. If the user has already entered the weather details, no additional data will be requested, as the weight has already been provided. If no data has been entered previously, the program will prompt the user to provide their body weight on the same screen, so that the user can directly calculate the calories burned without needing to enter unnecessary information if no other functionality is required.

```
Enter your choice


6
Choose your activity:

1. Jogging    2. Cycling     3. Swimming
4. Yoga       5. Strength    6. Hiking
7. Running    8. Tennis      9. Water Sports


8
Duration of exercise in hours>>
2
Your weight in Kg>>
75
Estimated Calories Burned: 600.0Kcal
```

```
//User selection: 6. Calculate Calories Burned
} else if (userSelection == 6){
    System.out.println(activitiesMenu);
    activity = keyboard.nextInt();
    System.out.println("Duration of exercise in hours>>");
    duration = keyboard.nextDouble();
    if (myFitness == null){
        if (myFitnessJustWeight == null){
            myFitnessJustWeight = new Fitness();
            System.out.println("Your weight in Kg>>");
            weight = keyboard.nextDouble();
            myFitnessJustWeight.setWeight(weight);
            myFitnessJustWeight.setActivityDetails(activity, duration);
            myFitnessJustWeight.computeCaloriesBurned();
        } else {
            myFitnessJustWeight.setActivityDetails(activity, duration);
            myFitnessJustWeight.computeCaloriesBurned();
        }

    } else {
        myFitness.setActivityDetails(activity, duration);
        myFitness.computeCaloriesBurned();
    }
```

When entering the body weight data on this screen, the object **myFitnessJustWeight**, which was initially set to null, is created. The program now checks if this data has already been provided. Therefore, if the user selects option 6 again, they will only be asked to select a sport and enter the exercise duration, as the weight has already been stored.

```
Enter your choice


6
Choose your activity:

1. Jogging    2. Cycling     3. Swimming
4. Yoga       5. Strength    6. Hiking
7. Running    8. Tennis      9. Water Sports


9
Duration of exercise in hours>>
1
Estimated Calories Burned: 825.0Kcal
```

If the user wants to check the calories burned for a different weight, they can delete all stored data by selecting option 7, "Reset Application." This will cause the program to restart automatically.

```
Enter your choice


7
Application reset successfully. All data has been cleared.

Application Menu

1.      Provide weather details (Temperature, Humidity, Wind Speed, UV Index and rain)
2.      Weather Advice (Temperature and rain conditions)
3.      Calculate Heat Index
4.      Check Wind Speed conditions
5.      UV Index precautions
6.      Calculate the number of calories burned for an activity
7.      Reset Application
8.      Exit Application

Enter your choice
```

If the user selects option 8, "Exit Application," the program will ask if they want to check other parameters. If the answer is yes, the application will reset and the stored data will be deleted. If the answer is no, the application will exit.

```
Enter your choice


8
Would you like to check another set of conditions?
yes
Application reset successfully. All data has been cleared.

Application Menu

1.      Provide weather details (Temperature, Humidity, Wind Speed, UV Index and rain)
2.      Weather Advice (Temperature and rain conditions)
3.      Calculate Heat Index
4.      Check Wind Speed conditions
5.      UV Index precautions
6.      Calculate the number of calories burned for an activity
7.      Reset Application
8.      Exit Application

Enter your choice


8
Would you like to check another set of conditions?
no
Thank You for using FitnessApp.
Good Bye.

C:\Users\Usuario\Desktop\FitnessApp.java>
```

```
    }
//User selection: 7. Reset Application
} else if (userSelection == 7){
    myFitness = null;
    myFitnessJustWeight = null;
    System.out.println("Application reset successfully. All data has been cleared.");
//User selection: 8. Exit Application
} else if (userSelection == 8){
    System.out.println("Would you like to check another set of conditions?");
    checkAnotherSet = keyboard.next();
    if (checkAnotherSet.equals("yes")){
        myFitness = null;
        myFitnessJustWeight = null;
        System.out.println("Application reset successfully. All data has been cleared.");
    } else {
        break;
    }
```

Up to this point, we have seen how the application works when the user does not provide the necessary details and how errors made by the user when entering invalid data are handled. Next, we will proceed to enter the data and test the full functionalities of the application.

```
Enter your choice


1
Temperature in ºC>>
Hot
Please enter a valid input. It should be a number between -20 and 50.
Temperature in ºC>>
28
Humidity Percentage>>
40
Wind Speed in m/s>>
10
UV Index>>
5
Is it raining heavily or stormy weather (yes/no)?
no
Your Weight in Kg>>
75
Data has been succesfully saved.
```

Upon selecting option 1, the program begins requesting the various data. If incorrect or out-of-range data is provided, the application informs the user of the error and gives instructions on how to enter the data correctly. This same system has been applied to all the data the user must input. Once all the data is entered correctly, it is stored in the Fitness class, and the user is notified that the data has been saved.

At this point, the user can explore options 2, 3, 4, and 5, which will provide various metrics, recommendations, and precautions based on the data entered previously.

```
Enter your choice


2
Weather Advice:
Recommended swimming or water sports, and suggest doing outdoor activities in the early morning or evening to avoid the heat.

Application Menu

1.      Provide weather details (Temperature, Humidity, Wind Speed, UV Index and rain)
2.      Weather Advice (Temperature and rain conditions)
3.      Calculate Heat Index
4.      Check Wind Speed conditions
5.      UV Index precautions
6.      Calculate the number of calories burned for an activity
7.      Reset Application
8.      Exit Application

Enter your choice


3
Heat Index: 30.0ºC
There is no specific guidance based on that temperature.
```

```
Enter your choice


4
Wind Advice:
Recommended activities like jogging or hiking in sheltered areas.

Application Menu

1.      Provide weather details (Temperature, Humidity, Wind Speed, UV Index and rain)
2.      Weather Advice (Temperature and rain conditions)
3.      Calculate Heat Index
4.      Check Wind Speed conditions
5.      UV Index precautions
6.      Calculate the number of calories burned for an activity
7.      Reset Application
8.      Exit Application

Enter your choice


5
UV Protection Advice:
It is advised to use sunscreen and wear a hat.

Application Menu

1.      Provide weather details (Temperature, Humidity, Wind Speed, UV Index and rain)
2.      Weather Advice (Temperature and rain conditions)
3.      Calculate Heat Index
4.      Check Wind Speed conditions
5.      UV Index precautions
6.      Calculate the number of calories burned for an activity
7.      Reset Application
8.      Exit Application

Enter your choice
```

```java
//computation methods
public void computeTemperatureConditions(){
    System.out.println("Weather Advice:");
    if (temperature < 0 || rain.equals("yes")){
        System.out.println("Recommended indoor activities like yoga or strength training.");
    } else if (temperature >= 0 && temperature <= 10) {
        System.out.println("Recommended outdoor activities like jogging or hiking, and advise dressing warmly.");
    } else if (temperature > 10 && temperature <= 25){
        System.out.println("Recommended outdoor activities such as running, cycling, or tennis.");
    } else {
        System.out.println("Recommended swimming or water sports, and suggest doing outdoor activities in the early morning or evenin
    }
}

public void computeHeatIndex(){
    //heatIndex =  temperature + (0.33 * humidity) - (0.7 * windSpeed) -4;
    if (temperature >= 20 && humidity <= 40){
        heatIndex =  Math.round(temperature + (0.33 * humidity) - (0.7 * windSpeed) -4);
        System.out.println("Heat Index: " + heatIndex + "°C");
        //System.out.printf("Heat Index: %.1f°C%n", heatIndex);
        if (heatIndex > 30){
            //System.out.println("Heat Index: " + heatIndex );
            System.out.println("Avoid strenuous outdoor activities. Consider alternatives like walking or swimming.");
        } else {
        System.out.println("There is no specific guidance based on that temperature.");
        }
    } else {
        System.out.println("There is no specific guidance based on that temperature.");
    }
}
//1m/s is equal 3.6km/h
public void computeCheckWindSpeed(){
    System.out.println("Wind Advice: ");
    if (windSpeed > (40/3.6)){
        System.out.println("Avoid activities like cycling or running.");
    } else if (windSpeed >= (20/3.6) && windSpeed <= (40/3.6)){
        System.out.println("Recommended activities like jogging or hiking in sheltered areas.");
    } else {
        System.out.println("All outdoor activities are safe to perform.");
    }
}
```

Source Code

FitnessApp.java

```java
/*
 * @author: Adrian Tarin
 * @date: 28/10/2024
 * @file: FitnessApp.java
 */

import java.util.Scanner;

public class FitnessApp{
    public static void main(String args[]){
        //declare variables
        int temperature;
        int humidity;
        int windSpeed;
        int uVIndex;
        String rain;
        int activity;
        double weight;
        double duration;
        int userSelection;
        String checkAnotherSet;

        //declare/create objects
        Scanner keyboard = new Scanner(System.in);
        Fitness myFitness = null;
        Fitness myFitnessJustWeight = null;
```

```
        //prompt user for input
        String applicationMenu = """

                                 Application Menu

                                 1.   Provide weather details
(Temperature, Humidity, Wind Speed, UV Index and rain)
                                 2.   Weather Advice
(Temperature and rain conditions)
                                 3.   Calculate Heat Index
                                 4.   Check Wind Speed
conditions
                                 5.   UV Index precautions
                                 6.   Calculate the number of
calories burned for an activity
                                 7.   Reset Application
                                 8.   Exit Application

                                 Enter your choice

                                 """;
        String activitiesMenu = """
                                Choose your activity:

                                1. Jogging    2. Cycling     3.
Swimming
                                4. Yoga       5. Strength    6.
Hiking
                                7. Running    8. Tennis      9.
Water Sports

                                """;


        while (true){
            try {
                //Show Application Menu with options to the user
                System.out.println(applicationMenu);
                userSelection = keyboard.nextInt();
                //User selection: 1. Provide Weather Details
                if (userSelection == 1){
                    //Temperature validation
                    while (true) {
                        try {
                            System.out.println("Temperature in
°C>>");
                            temperature = keyboard.nextInt();
                            if (temperature < -20 || temperature
> 50){
                                System.out.println("Please enter
a valid input. It should be a number between -20 and 50.");
                                keyboard.next();
                            } else {
                                break;
                            }
```

```
                        } catch (Exception e) {
                            System.out.println("Please enter a
valid input. It should be a number between -20 and 50.");
                            keyboard.next();
                            }
                    }
                    //Humidity validation
                    while (true) {
                        try {
                            System.out.println("Humidity
Percentage>>");
                            humidity = keyboard.nextInt();
                            if (humidity < 0 || humidity > 100){
                                System.out.println("Please enter
a valid input. It should be a number between 0 and 100.");
                                keyboard.next();
                            } else {
                                break;
                            }
                        } catch (Exception e) {
                            System.out.println("Please enter a
valid input. It should be a number between 0 and 100.");
                            keyboard.next();
                            }
                    }
                    //Wind Speed validation
                    while (true) {
                        try {
                            System.out.println("Wind Speed in
m/s>>");
                            windSpeed = keyboard.nextInt();
                            if (windSpeed < 0 || windSpeed >
50){
                                System.out.println("Please enter
a valid input. It should be a number between 0 and 50.");
                                keyboard.next();
                            } else {
                                break;
                            }
                        } catch (Exception e) {
                            System.out.println("Please enter a
valid input. It should be a number between 0 and 50.");
                            keyboard.next();
                            }
                    }
                    //UV validation
                    while (true) {
                        try {
                            System.out.println("UV Index>>");
                            uVIndex = keyboard.nextInt();
                            if (uVIndex < 0 || uVIndex > 11){
                                System.out.println("Please enter
a valid input. It should be a number between 0 and 11.");
                                keyboard.next();
                            } else {
                                break;
```

```
                            }
                        } catch (Exception e) {
                            System.out.println("Please enter a
valid input. It should be a number between 0 and 11.");
                            keyboard.next();
                            }
                    }
                    //Rain validation
                    while (true) {
                        try {
                            System.out.println("Is it raining
heavily or stormy weather (yes/no)?");
                            rain = keyboard.next();
                            if (rain.equals("yes") == false &&
rain.equals("no") == false){
                                System.out.println("Please enter
a valid input. You should type 'yes' or 'no'.");
                                keyboard.next();
                            } else {
                                break;
                            }
                        } catch (Exception e) {
                            System.out.println("Please enter a
valid input. You should type 'yes' or 'no'.");
                            keyboard.next();
                            }
                    }
                    //Weight validation
                    while (true) {
                        try {
                            System.out.println("Your Weight in
Kg>>");
                            weight = keyboard.nextDouble();
                            if (weight < 30 || weight > 200){
                                System.out.println("Please enter
a valid input. It should be a number between 30 and 200.");
                                keyboard.next();
                            } else {
                                break;
                            }
                        } catch (Exception e) {
                            System.out.println("Please enter a
valid input. It should be a number between 30 and 200.");
                            keyboard.next();
                            }
                    }
                    //Create Fitness object and set details
                    myFitness = new Fitness();
                    myFitness.setWeatherDetails(temperature,
humidity, windSpeed, uVIndex, rain, weight);
                    System.out.println("Data has been
succesfully saved.");

                //User selection: 2. Weather Advice (temperature
and rain conditions)
                } else if (userSelection == 2){
```

```
                    if (myFitness != null){

myFitness.computeTemperatureConditions();
                    } else {
                        System.out.println("Please enter weather
details.");
                    }
                //User selection: 3. Calculate Heat Index
                } else if (userSelection == 3){
                    if (myFitness != null){
                        myFitness.computeHeatIndex();
                    } else {
                        System.out.println("Please enter weather
details.");
                    }
                //User Selection: 4. Check Wind Speed conditions
                }else if (userSelection == 4){
                    if (myFitness != null){
                        myFitness.computeCheckWindSpeed();
                    } else {
                        System.out.println("Please enter weather
details.");
                    }
                //User selection: 5. UV Index precautions
                } else if (userSelection == 5){
                    if (myFitness != null){
                        myFitness.computeUVIndexPrecautions();
                    } else {
                        System.out.println("Please enter weather
details.");
                    }
                //User selection: 6. Calculate Calories Burned
                } else if (userSelection == 6){
                    System.out.println(activitiesMenu);
                    activity = keyboard.nextInt();
                    System.out.println("Duration of exercise in
hours>>");
                    duration = keyboard.nextDouble();
                    if (myFitness == null){
                        if (myFitnessJustWeight == null){
                            myFitnessJustWeight = new Fitness();
                            System.out.println("Your weight in
Kg>>");
                            weight = keyboard.nextDouble();

myFitnessJustWeight.setWeight(weight);

myFitnessJustWeight.setActivityDetails(activity, duration);

myFitnessJustWeight.computeCaloriesBurned();
                        } else {

myFitnessJustWeight.setActivityDetails(activity, duration);

myFitnessJustWeight.computeCaloriesBurned();
                        }
```

```
                    } else {
                        myFitness.setActivityDetails(activity,
duration);
                        myFitness.computeCaloriesBurned();
                    }
                //User selection: 7. Reset Application
                } else if (userSelection == 7){
                    myFitness = null;
                    myFitnessJustWeight = null;
                    System.out.println("Application reset
successfully. All data has been cleared.");
                //User selection: 8. Exit Application
                } else if (userSelection == 8){
                    System.out.println("Would you like to check
another set of conditions?");
                    checkAnotherSet = keyboard.next();
                    if (checkAnotherSet.equals("yes")){
                        myFitness = null;
                        myFitnessJustWeight = null;
                        System.out.println("Application reset
successfully. All data has been cleared.");
                    } else {
                        break;
                    }
                      //User selection is valid data type but
not an option between 1-8
                } else {
                    System.out.println("Selection not valid.
Please enter a valid choice.");
                }

            //User input is not an int
            } catch (Exception e) {
                System.out.println("Selection not valid. Please
enter a valid choice.");
                keyboard.next();
                }

        }//end first while (true)
        System.out.println("Thank You for using
FitnessApp.\nGood Bye.");
    }//end public static void main(String args[])
}//end puclic class FitnessApp
```

Fitness.java

```
/*
 * @author: Adrian Tarin
 * @date: 28/10/2024
 * @file: Fitness.java
 */

public class Fitness{
    //data members
```

```
    private int temperature, humidity, windSpeed, uVIndex,
activity, mET;
    private double heatIndex, duration, weight, caloriesBurned;
    private String rain;



    //constructor
    public Fitness(){
    }

    //methods
    //set methods
    public void setWeatherDetails(int temperature, int humidity,
int windSpeed, int uVIndex, String rain, double weight){
        this.temperature = temperature;
        this.humidity = humidity;
        this.windSpeed = windSpeed;
        this.uVIndex = uVIndex;
        this.rain = rain;
        this.weight = weight;
    }

    public void setActivityDetails(int activity, double
duration){
        this.activity = activity;
        this.duration = duration;
    }

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


    //computation methods
    public void computeTemperatureConditions(){
        System.out.println("Weather Advice:");
        if (temperature < 0 || rain.equals("yes")){
            System.out.println("Recommended indoor activities
like yoga or strength training.");
        } else if (temperature >= 0 && temperature <= 10) {
            System.out.println("Recommended outdoor activities
like jogging or hiking, and advise dressing warmly.");
        } else if (temperature > 10 && temperature <= 25){
            System.out.println("Recommended outdoor activities
such as running, cycling, or tennis.");
        } else {
            System.out.println("Recommended swimming or water
sports, and suggest doing outdoor activities in the early
morning or evening to avoid the heat.");
        }
    }

    public void computeHeatIndex(){
        //heatIndex =  temperature + (0.33 * humidity) - (0.7 *
windSpeed) -4;
```

```
        if (temperature >= 20 && humidity <= 40){
            heatIndex =  Math.round(temperature + (0.33 *
humidity) - (0.7 * windSpeed) -4);
            System.out.println("Heat Index: " + heatIndex +
"°C");
            //System.out.printf("Heat Index: %.1f°C%n",
heatIndex);
            if (heatIndex > 30){
                //System.out.println("Heat Index: " + heatIndex
);
                System.out.println("Avoid strenuous outdoor
activities. Consider alternatives like walking or swimming.");
            } else {
            System.out.println("There is no specific guidance
based on that temperature.");
            }
        } else {
            System.out.println("There is no specific guidance
based on that temperature.");
        }
    }
    //1m/s is equal 3.6km/h
    public void computeCheckWindSpeed(){
        System.out.println("Wind Advice: ");
        if (windSpeed > (40/3.6)){
            System.out.println("Avoid activities like cycling or
running.");
        } else if (windSpeed >= (20/3.6) && windSpeed <=
(40/3.6)){
            System.out.println("Recommended activities like
jogging or hiking in sheltered areas.");
        } else {
            System.out.println("All outdoor activities are safe
to perform.");
        }
    }

    public void computeUVIndexPrecautions(){
        System.out.println("UV Protection Advice: ");
        if (uVIndex > 7){
            System.out.println("Apply sunscreen, avoid direct
sunlight during peak hours (10 AM - 4 PM), and wear protective
clothing, a hat, and sunglasses.");
        } else if (uVIndex >= 5 && uVIndex <=7){
            System.out.println("It is advised to use sunscreen
and wear a hat.");
        } else {
            System.out.println("No specific sun protection is
required, though using sunscreen is still encouraged.");
        }
    }

    public void computeCaloriesBurned(){
        if (activity == 1){
            mET = 7;
        } else if (activity == 2){
```

```
            mET = 8;
        } else if (activity == 3){
            mET = 6;
        } else if (activity == 4){
            mET = 1;
        } else if (activity == 5){
            mET = 9;
        } else if (activity == 6){
            mET = 10;
        } else if (activity == 7){
            mET = 5;
        } else if (activity == 8){
            mET = 4;
        } else if (activity == 9){
            mET = 11;
        }

        caloriesBurned = mET * weight * duration;
        System.out.println("Estimated Calories Burned: " +
caloriesBurned + "Kcal");
    }

}
```