**Kwantlen Polytechnic University**

**INFO 3235 S50 Spring 2022**

**Project**

**Enoch Chew 100390133**

**Josiah Tolentino 100304547**

**Ramandeep Vandal 100377660**

**April 6th, 2022**

**Overview of our Prototype Design Approach vs Existing Products**

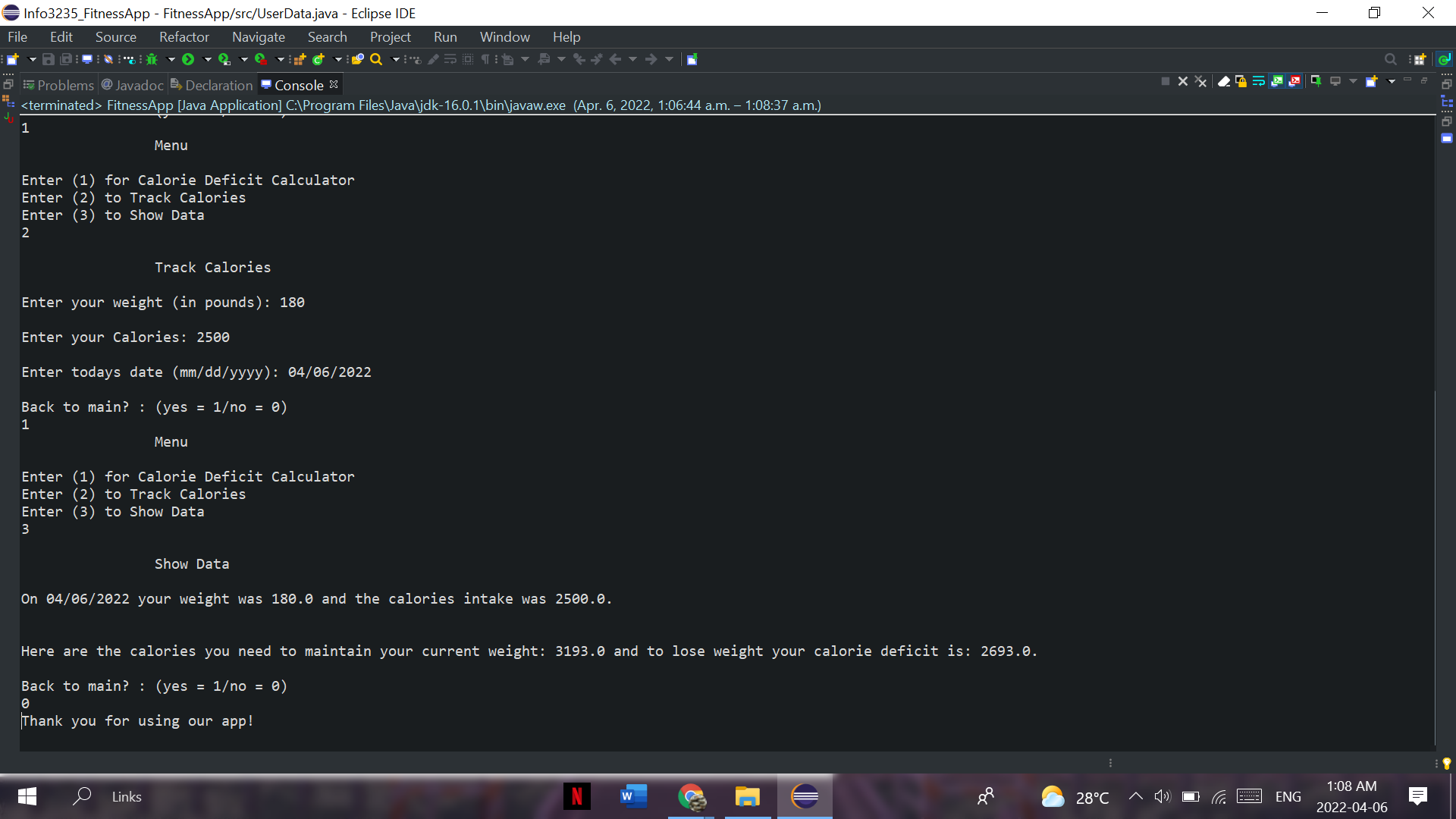
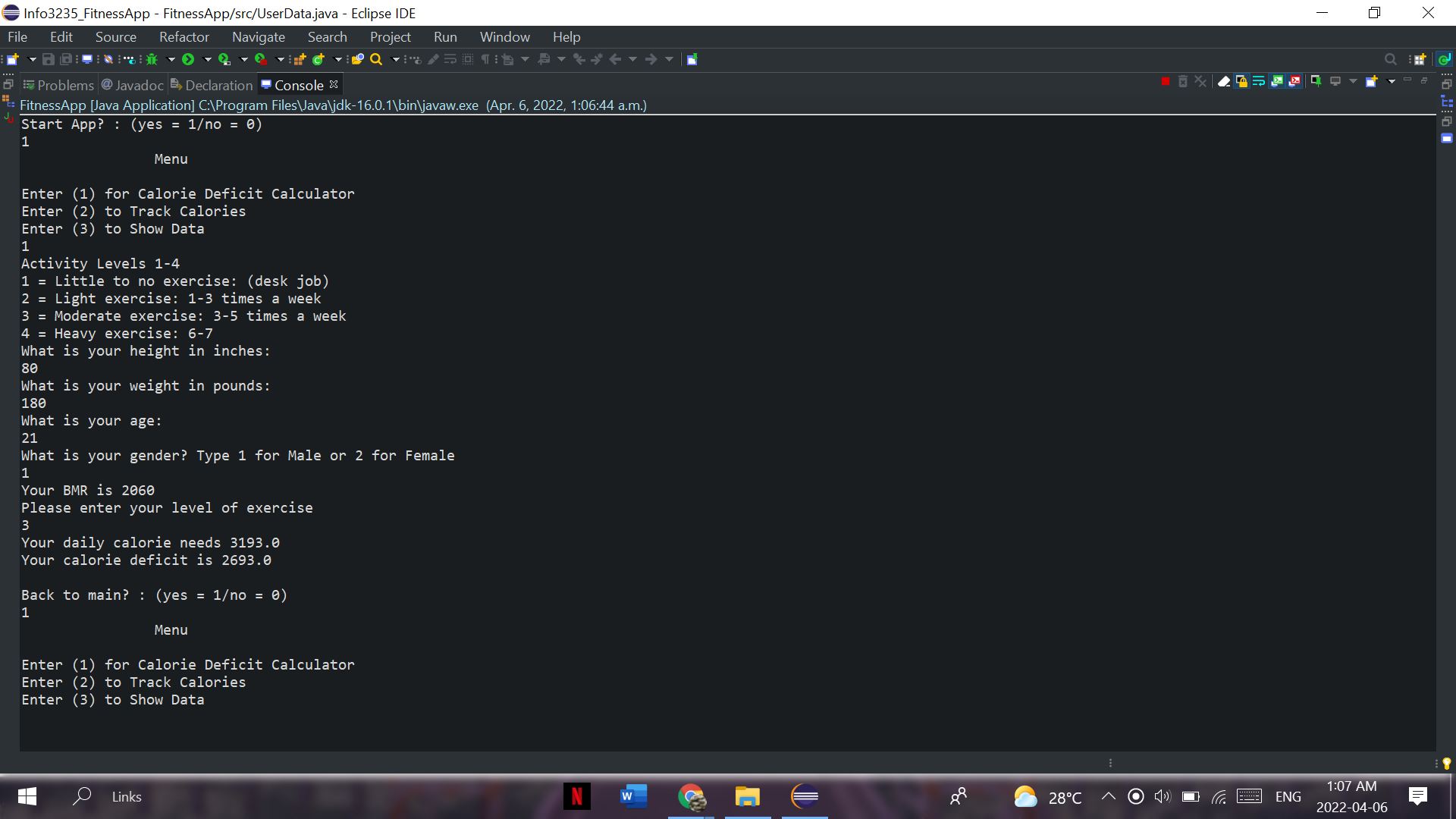
**Existing Products: Lose It! and FatCalc**

“Lose It!” is a calorie tracker app that allows users to scan products to count calories. This app also provides the user with their BMR (daily calorie needs). However, this amount is only calculated when the user first creates their account; to use the calculator again the user must navigate through pages of settings.

Moreover, “FatCalc” is a website that has a calorie deficit calculator, it allows users to calculate the number of calories needed per day to lose weight. Even though, the calculator can be used infinitely the website has no other use.

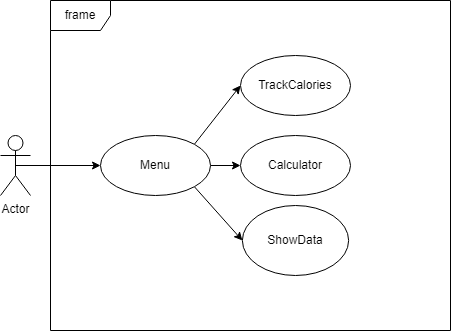
Our approach is to combine the features of “LoseIt!” and “FatCalc”. Our app will track calories and allow users to calculate their BMR or calorie deficit. Unlike “LoseIt!” and “FatCalc”, our app will have more features and will show all the calories tracked by the user.

**Our Prototype Design:**

The images above show the features and UI (user interface) of our app. The image on the left shows the calorie deficit calculator screen; which calculates the BMR based on user input and allows the user to return to the menu screen or close the app. The image on the right shows the track calories and show data screens; the data entered on the track calories screen will be displayed on the show data screen. Also, the menu screen, shown in both images allows the user to easily use the features and navigate the app. 

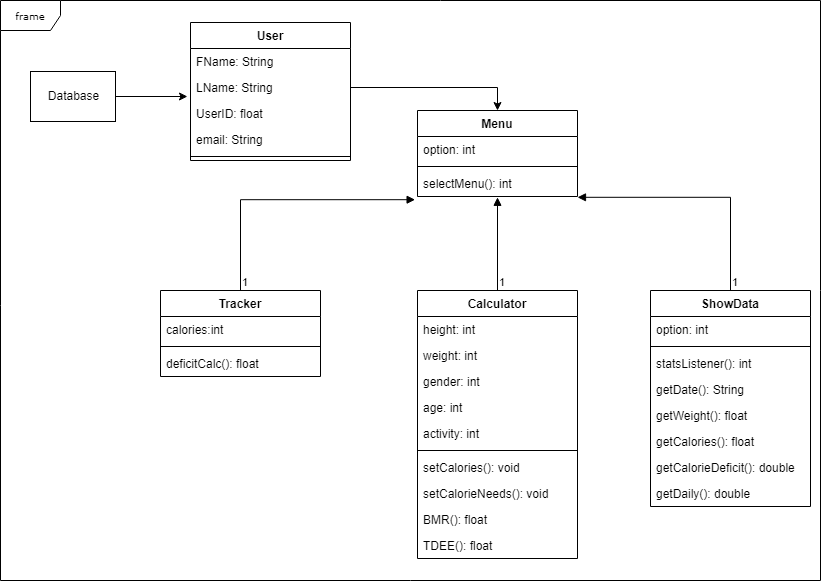
**UML diagrams**

These diagrams help depict the structure of our program and how users would interact with the app.

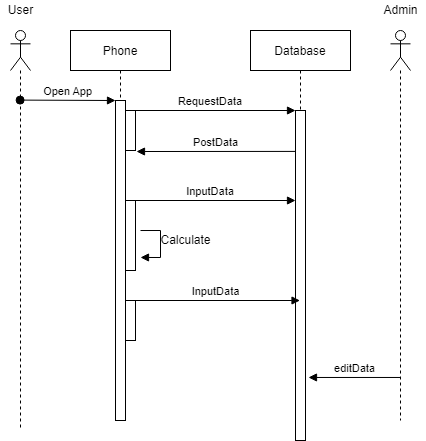
**Use Case Diagram**

Depicted here are the different options that the user will be presented with upon opening the app.

Class Diagram

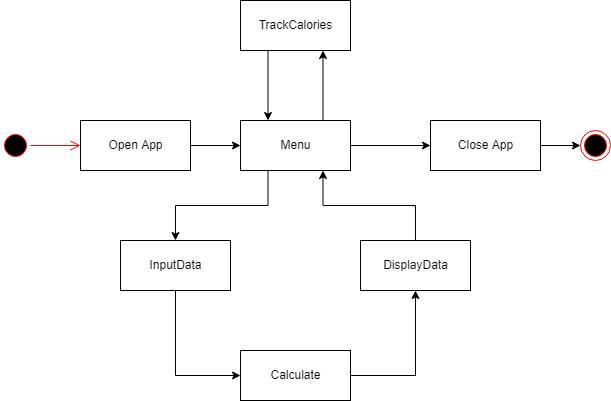
This diagram shows how each class has different attributes and how they would be used.Sequence Diagram

This diagram shows how the flow of data and functions would flow depending on what function is done and how users interact with the app.



State Diagram

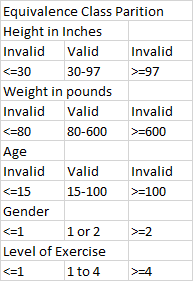
This state diagram shows how different functions transition between different states in order to function.



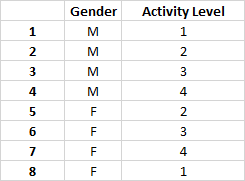
**Test Cases**

**Black-box testing**

For black-box testing, it was a case of making sure user inputs are correct and constrained to realistic proportions of human beings.

****

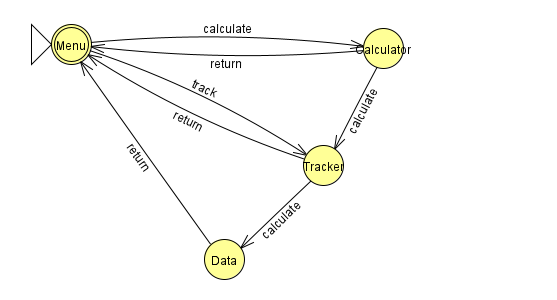
The pairwise tests were to check the variation of user inputs to the constrained values of Gender and Activity Level. The reason for this is in the calculations for BMR and TDEE. BMR has separate calculations for Male and Female, while TDEE uses BMR in their calorie deficit calculator as described in the code.

****

**White-box testing**

**Finite State Machine**

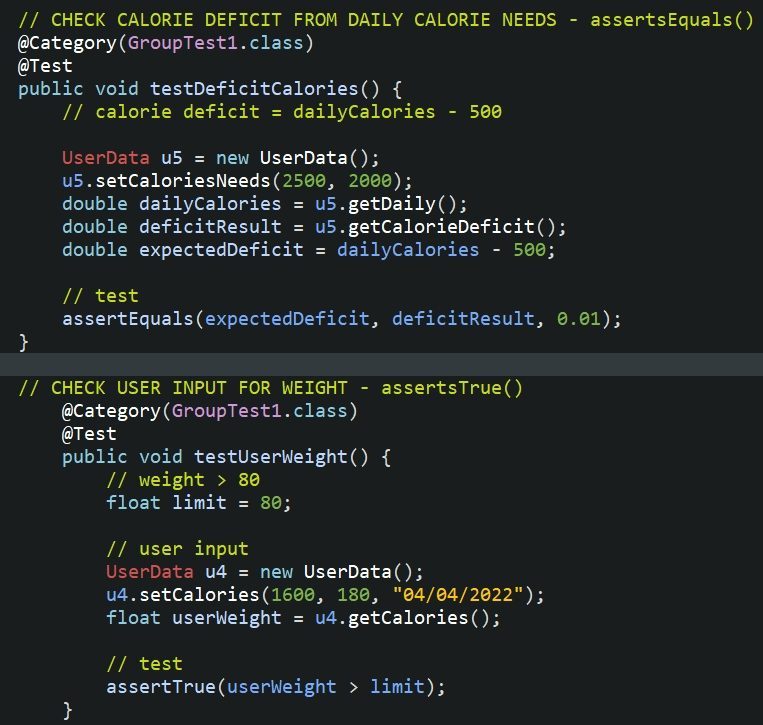
When constructing our program, we made sure that user options after inputs and analysis of user variables can return to the menu and end the program. Unless a user force quits the code, it will only end at the initial point of the menu. Calculate as a transition is the core user transition from the initial calculator, to user tracking and overall data. Having menu to track be a separate variable in this case is for returning users.

****

**JUnit testing**

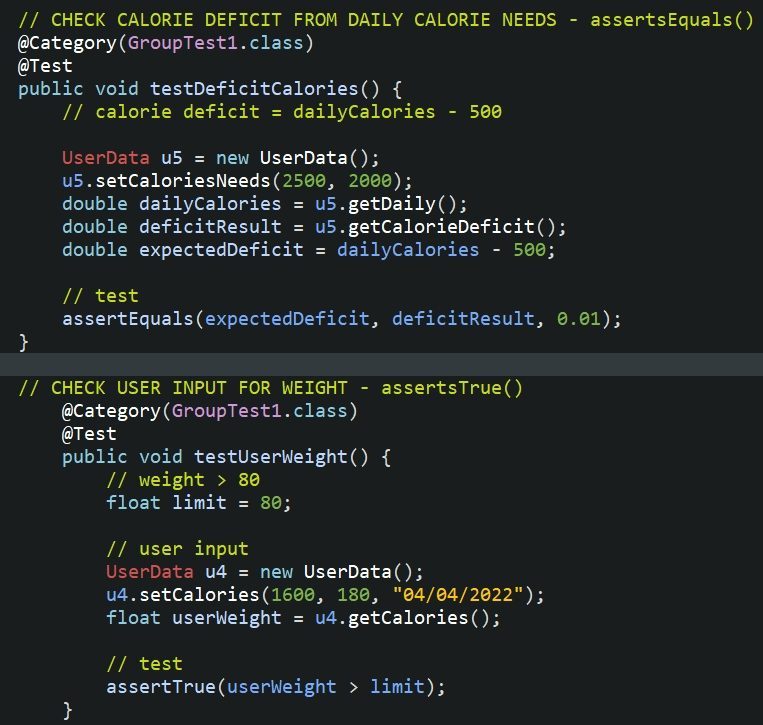
**Use of assertEquals() and assertTrue():**

**assertEquals():**

****

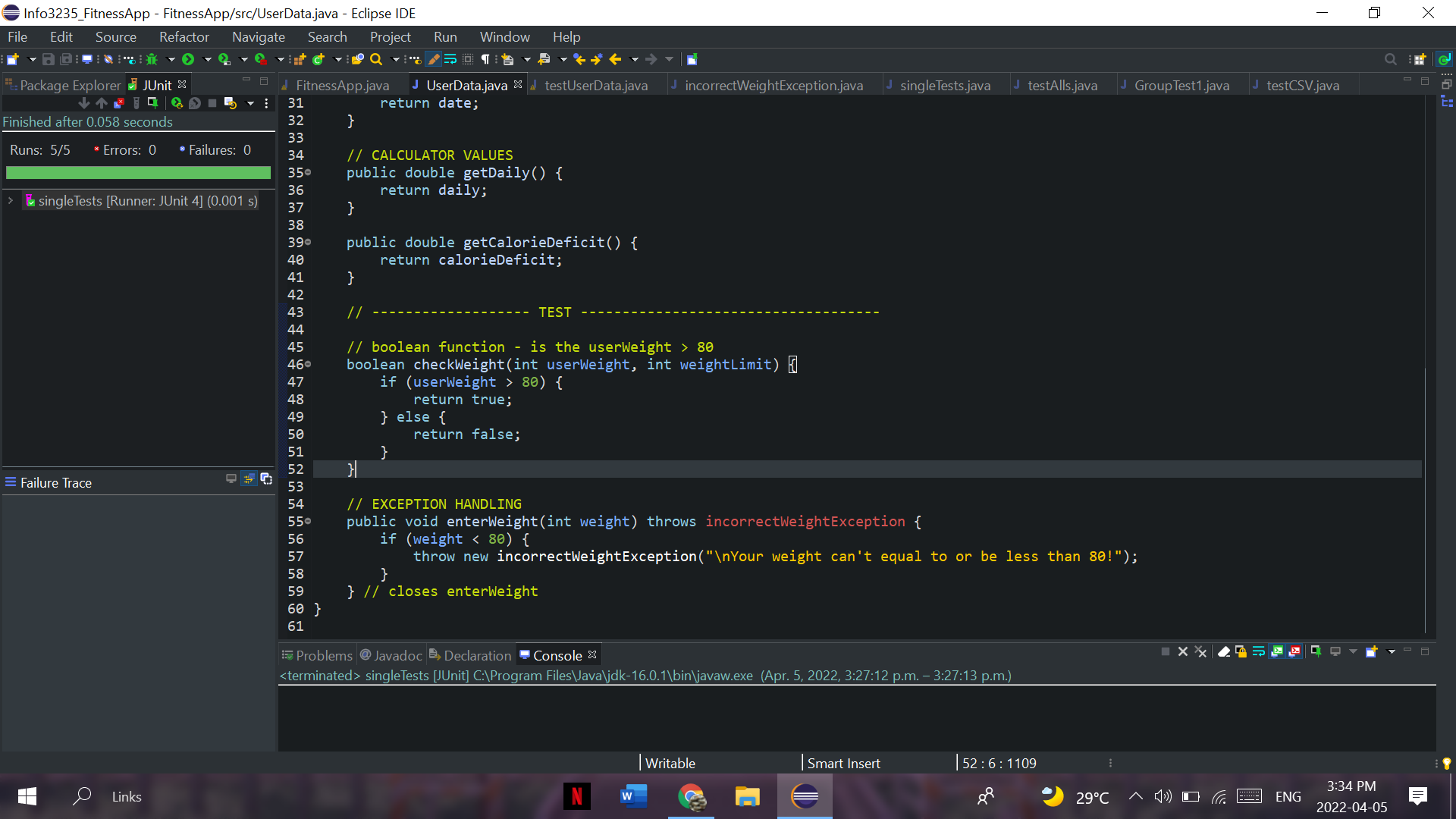
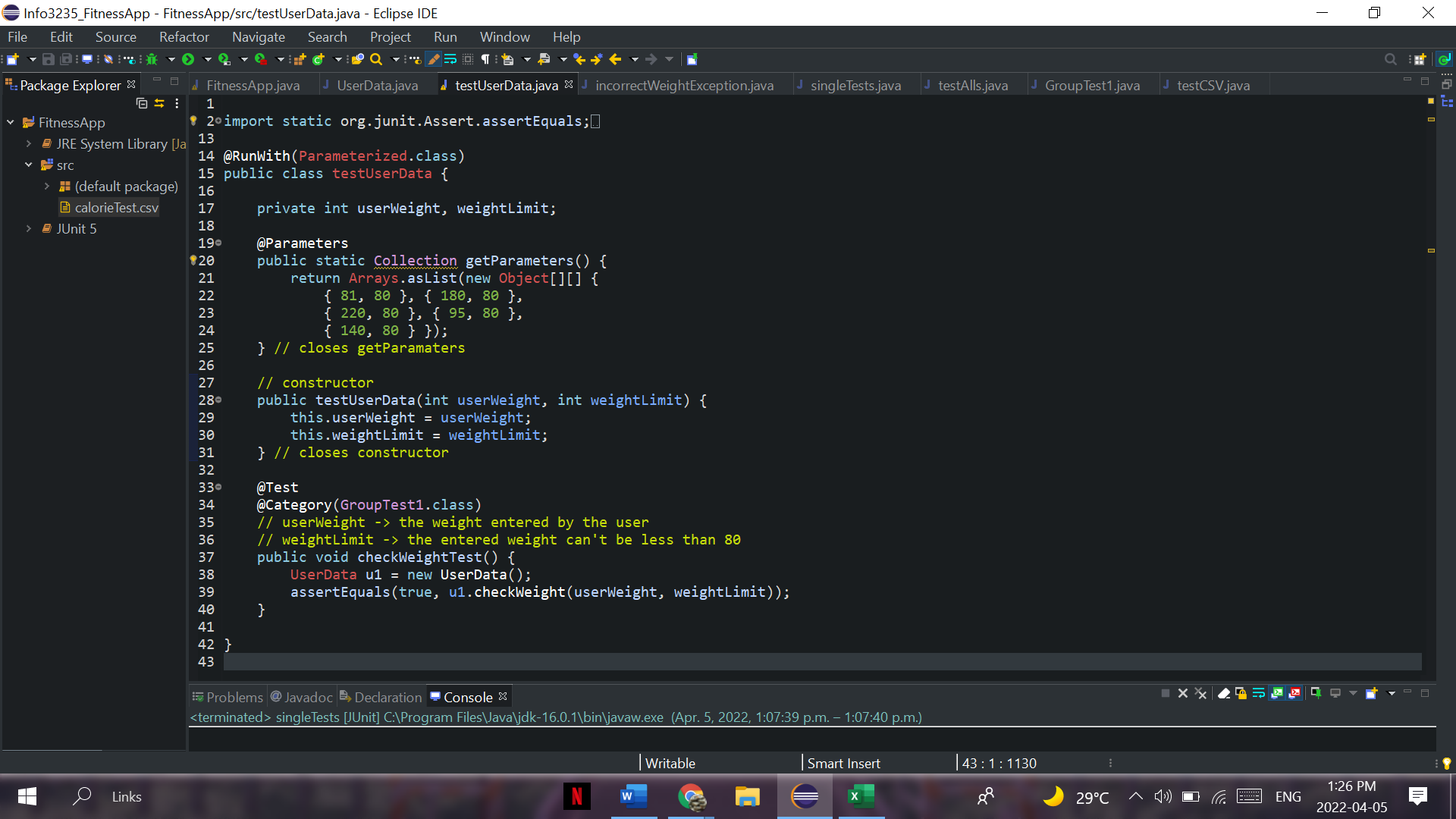
The code above tests the calorie deficit calculator. After the user adds the details for the calculator, the BMR (daily calorie needs) and the calorie deficit (BMR - 500) are stored in objects; which the user fetches when using the show data screen. In the code, we tested the calorie deficit calculation, the “deficitResult” had to be 500 less than the “dailyCalories” for the test to be successful. Due to this, we used assertEquals to check if the “deficitResult” was equal to the “expectedResult” (dailyCalories-500).

**assertTrue():**

****

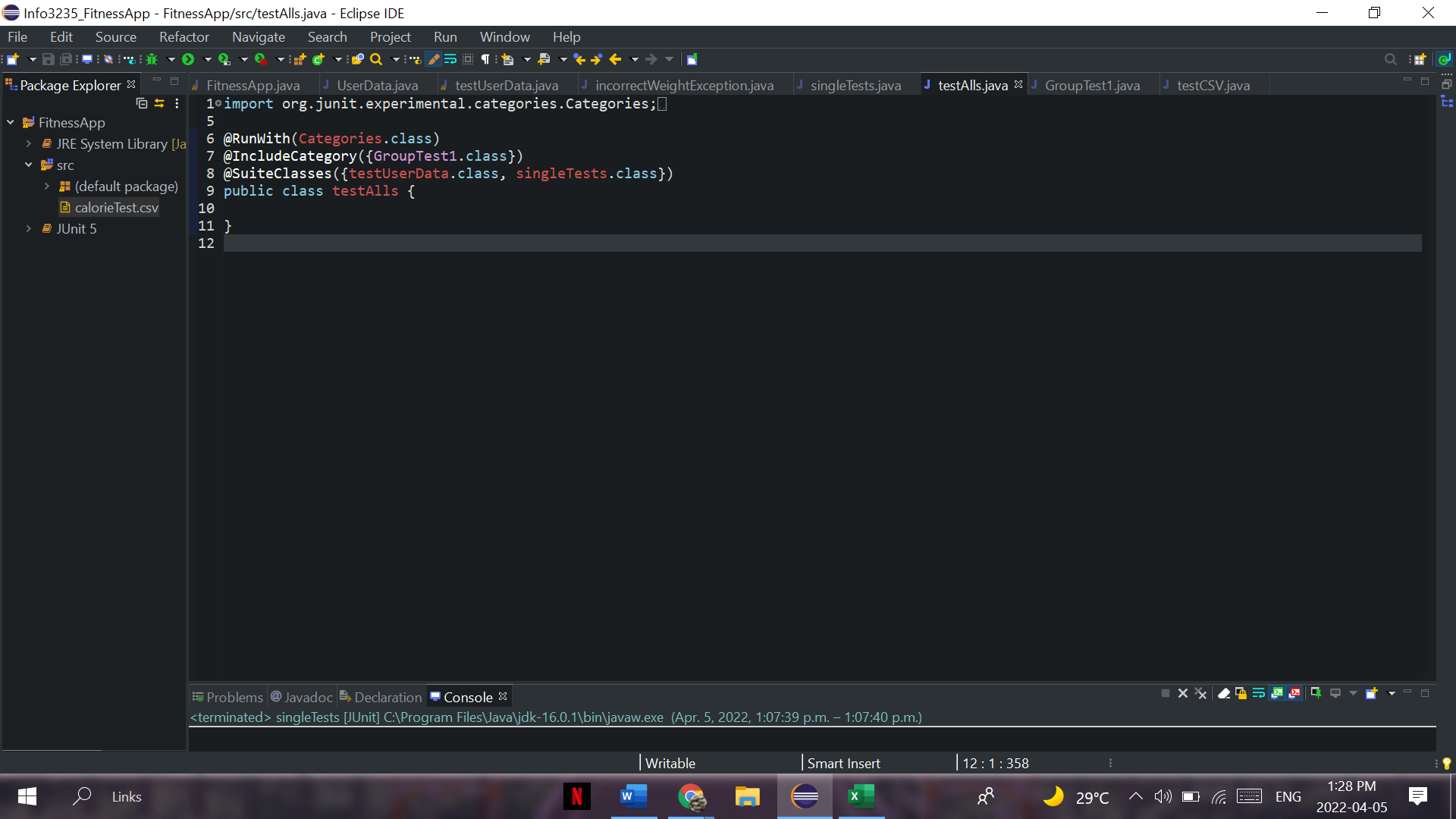
The code above tests the user input for weight on the track calories screen. The minimum amount of weight that can be entered for our app is 81 pounds; anything less than that is invalid. Due to this, we used assertTrue() to check if the user input (“userWeight”) was greater than the weight limit (“limit”).

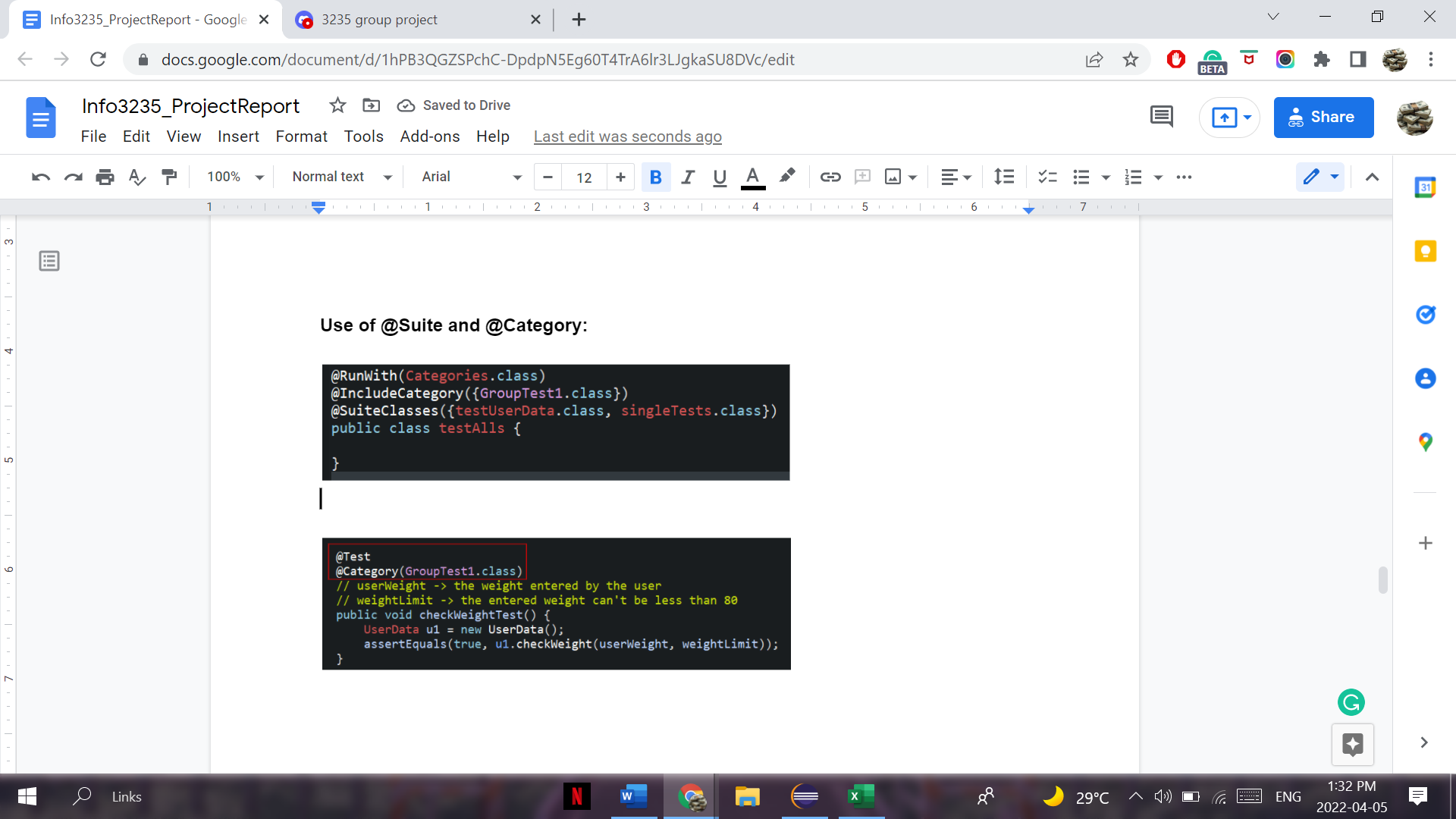
**Multiple Data (array) Input through @Parameters:**

****

We used @Parameters to test the user input for weight using an array of data inputs. In the code above, the “weightLimit” variable is equal to 80, and the “userWeight” variable represents the user input. The assertEquals uses the “checkWeight” function on the right to check if the input is greater than the limit and then compares the boolean value with the expected value (true).

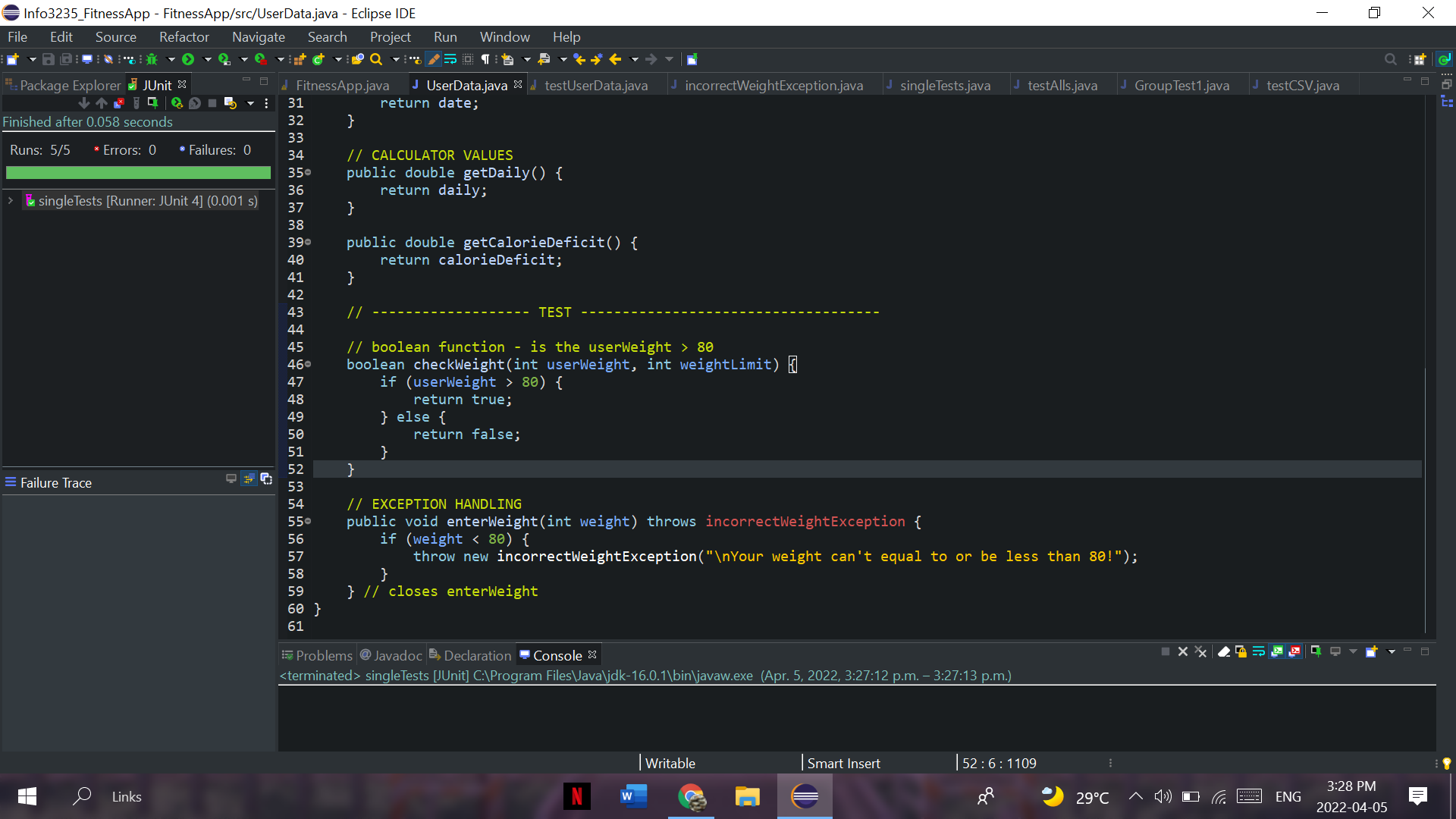
**Use of @Suite and @Category:**

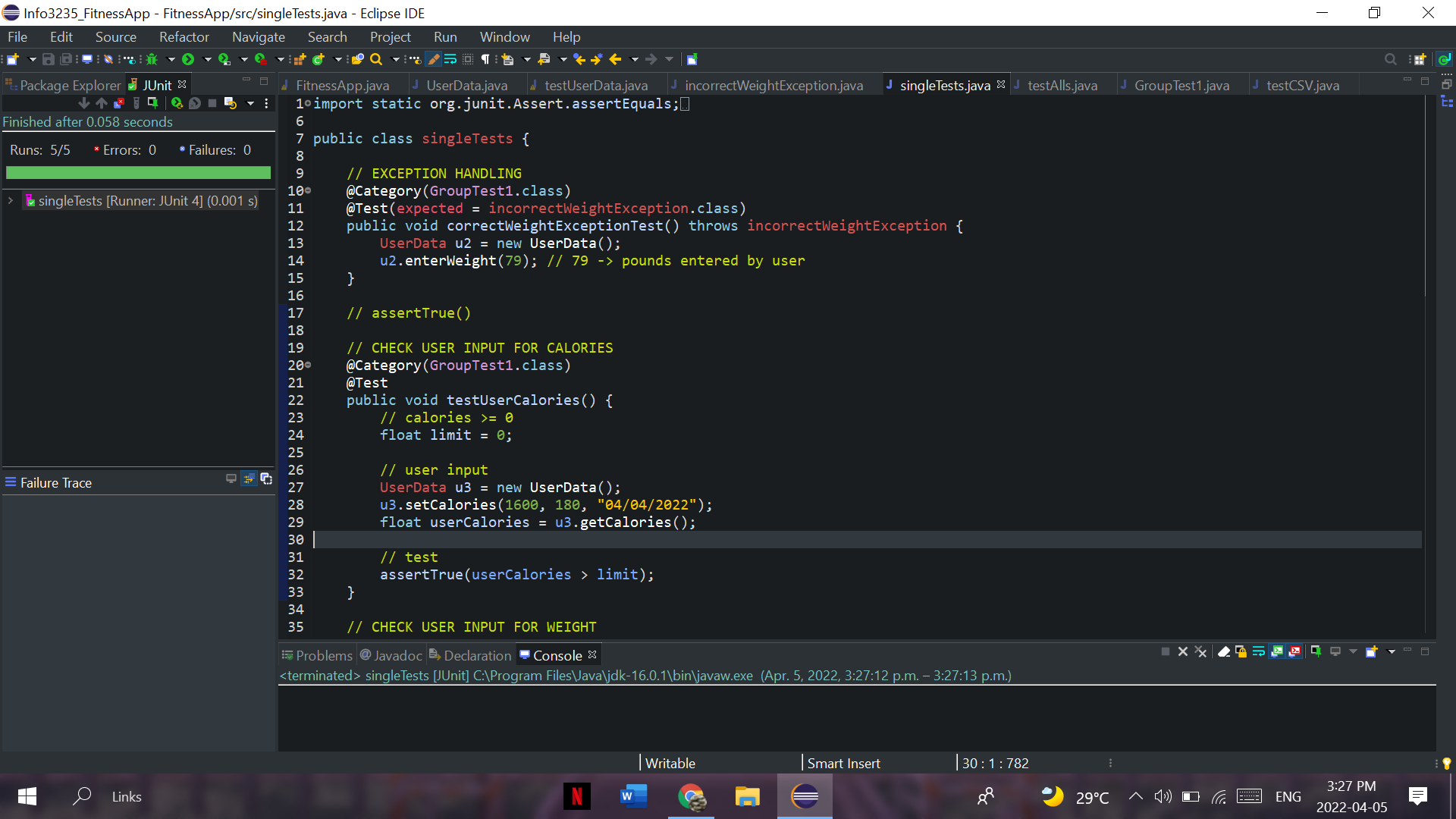
****

****

We used @Suite and @Category to group together the array data input, exception handling, assertEquals(), and assertTrue() test cases to run all test cases in the “testAlls” class. This allowed for running multiple test cases at once, instead of running each test case individually.

**Exception Handling:**

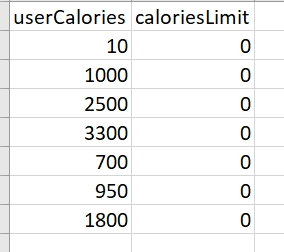
****

****

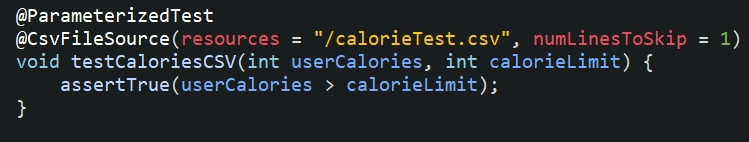
In the code above, we tested the user input for weight on the track calories screen. The correctWeightExceptionTest() function throws an exception if the weight entered by the user was less than or equal to 80. This ensures that the user can’t enter a weight input that is less than the minimum weight limit (80).

**Use of @CsvFileSource:**

calorieTest.csv file:



JUnit code:



We used assertEquals() with the @CsvFileSource JUnit technique to test the user input for calories on the track calories screen. The minimum amount of calories that can be entered for our app is 1; anything less than that is invalid. In the code above, we checked if the values in the “userCalories” column are greater than the values in the “caloriesLimit” column; if they were greater than the limit then the test was successful. Also, we used “numLinesToSkip=1” to skip the column titles in the “.csv file”.