**Criterion C: Development**

**Program Description:**

This program (using Java) will be a calculator that takes the inputs of a scout’s age, weight, and height, and the location’s total distance, elevation, and time period of a backpacking/hiking trip to end up with a calculated result of total calorie intake along with a macronutrient ratio of fats, proteins, and carbohydrates needed for a backpacking/hiking trip in units of calories per day.

**Input:**

* Keyboard input: User inputs data.
* File input: Data from keyboard input is then written into a text file.

Method name: Input()

----------------[MAIN MENU]----------------

[1]New entry

[2]Load entries

[3]Edit an entry

[4]Delete an entry

[X]Exit Program

Please enter a menu option number/letter:

Figure 1: Main menu

Method name: Input()

Please enter a menu option number/letter: 1

Enter the location name (Do not include spaces): OldRagMountain

Enter a distance (miles): 9

Enter an elevation gain (ft): 2355

Enter a time (hours): 8

Enter a gender (F or M):M

Enter an age (years): 20

Enter a height (inches): 68

Enter a weight (lbs.): 147

Figure 2: Main menu 🡪 [1]New Entry.

Method name: Load()

Please enter a menu option number/letter: 2

[0]Load all entries

[1]Search for an entry

Please enter the number of your desired action: 1

~~~Search for the name of your entry~~~:

[Please enter the name of the location of your desired entry (No spaces)]

Search: OldRagMountain

Figure 3: Main menu 🡪 [2]Load entries.

Method name: Edit()

Please enter a menu option number/letter: 3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Entry #1:

Location: OldRagMountain

Distance: 9

Elevation: 2355

Duration: 8 hours

Gender: M

Age: 20 years

Height: 68 in.

Weight: 147 lbs.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

~~~Which entry would you like to edit?~~~

Please enter the number of the desired entry (Indicated by ‘Entry #X’): 1

~~~What do you want to update?~~~

[0]Location: OldRagMountain

[1]Distance: 9 miles

[2]Elevation: 2355 ft.

[3]Duration: 8 hours

[4]Gender: M

[5]Age: 20 years

[6]Height: 68 in.

[7]Weight: 147 lbs.

Please enter the number of the attribute you wish to update (Indicated by [X]): 7

~~~What would you like to replace it with?~~~

Please enter a new value: 150

~~~Entry edited~~~

Figure 4: Main menu 🡪 [3]Edit an entry

Method name: Delete()

Please enter a menu option number/letter: 4

Delete one entry or delete all?

[1]Delete one entry

[2]Delete all entries

Please enter the number of your desired action: 1

Entry #1:

Location: test1

Duration: 1 hours

Entry #2:

Location: test2

Duration: 2 hours

~~~Which entry would you like to delete?~~~

Please enter the number of your desired entry: 2

~~~Entry deleted~~~

Figure 5 : Main menu 🡪 [4]Delete an Entry

**Processing:**

|  |  |  |
| --- | --- | --- |
| Algorithm | Method Name | Purpose |
| Main menu | Main() | Initializes a file “list.txt,” its corresponding PrintWriter, file Scanner, keyboard Scanner, and object ArrayList, all of which are static components. Creates a main menu for the user to be able to navigate to the different methods: Input(), Load(), Edit(), and Delete(). |
| (#2)Import file | Input() | Inputted data creates Entries objects and adds each object into an object ArrayList. Then each object is written into the text file with spaces in between each data entry. |
| Read the file | ReadObject() | The file is read and updated into the ArrayList e every time the program returns to the main menu. This keeps the data updated and consistent. |
| (#4)Search for an entry/Load all the entries | Load() | The user can either search for an entry using the location name of the entry or load all entries. |
| Print only one entry using index of ArrayList e | PrintAnEntry(int) | Using the index number of ArrayList e, this method takes the index of uses e.get(int) as an input for the called method, PrintRestuls(Entries) |
| Print results for an entry | PrintResults(Entries) | Takes data from the objects and formats it for the user to view. Calls methods: findBMR(), findTDEE(), MultiplyAL(), and findRatio() to carry out the calculations and present them to the user along with the other data. |
| (#5)Find the BMR of the person | findBMR(Entries) | Takes the person’s height, weight, age, and gender to calculate a basal metabolic rate for each entry. |
| (#5)Find the activity multiplier from the backpacking/hiking trip | findAM(Entries, ArrayList<Double>) | Takes distance and elevation gain to determine the activity multiplier. This is then multiplied with the BMR of the person. |
| Multiply values of two ArrayLists | findTotal(ArrayList<Double>, ArrayList<Double>) | Finds the total amount of calories by combining the double ArrayLists of findBMR, and findAM. |
| (#6)Apply the ratio to total calorie intake | findRatio(ArrayList<Double>) | Applies a specific ratio for distribute the total calorie intake into the three macronutrients: fats, proteins, and carbohydrates. |
| Each line in the file is put into a string ArrayList | ReadToString() | These strings can be easily managed and manipulated in methods Edit() when editing and Load() when searching. |
| Edits data in an entry | Edit() | User is able to choose to edit the name, distance, elevation, time, gender, age, height, or weight of an entry. |
| Deletes an entry/Clears database | Delete() | An entry is deleted by choice of user. All entries can also be cleared at once. |

**Output:**

1 OldRagMountain 9 2355 8 M 20 68 147

Figure 6: Sample of a line in the text file: “list.txt”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Entry #1:

Location: OldRagMountain

Distance: 9 miles

Elevation: 2355 ft.

Duration: 8 hours

Gender: M

Age: 20 years

Height: 68 in.

Weight: 147 lbs.

TOTAL CALORIES BURNED: 3247.88 cal/day

~~~Macronutritients Ratio~~~

Protiens (30%): 974.36 cal/day

Carbs (40%): 1299.15 cal/day

Fats (30%): 974.36 cal/day

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Figure 7: Sample output using Load() method

**Existing Libraries Used:**

* java.util.ArrayList
* java.util.Scanner
* java.io.File
* java.text.DecimalFormat
* java.io.PrintWriter
* java.io.FileWriter

**Programming Techniques and Complexity:**

1. ArrayLists: Keeps a list of entries for the file, “list.txt” Also stores doubles additional calculations.
2. File I/O: Creates a file for reading, writing, and updating of the list of entries in a text file.
3. If/Else Statements: Used for male/female calculations and their differences in BMR calculations. Also used within ‘for’ loops to select one element in an ArrayList.
4. Complex selection: Switch statements are used to prompt the user to select an option from main menu or other menus of other methods.
5. Looping: Loops will be used for the ‘for’ loops to select or make calculations for each element in the object ArrayList.
6. Searching: User inputs string to search for the name of the location of the entry.
7. User-defined methods: The program is organized into different methods which the user is allowed to call upon.
8. User-defined objects: Entries objects are created by the user.
9. Use of additional libraries: ArrayLists, Scanners, File, DecimalFormat, PrintWriter, and FileWriter.
10. Use of sentinels: Sentinels are used in a while loop in method Input() to prompt user if they want to add additional entries.

**Organization:**

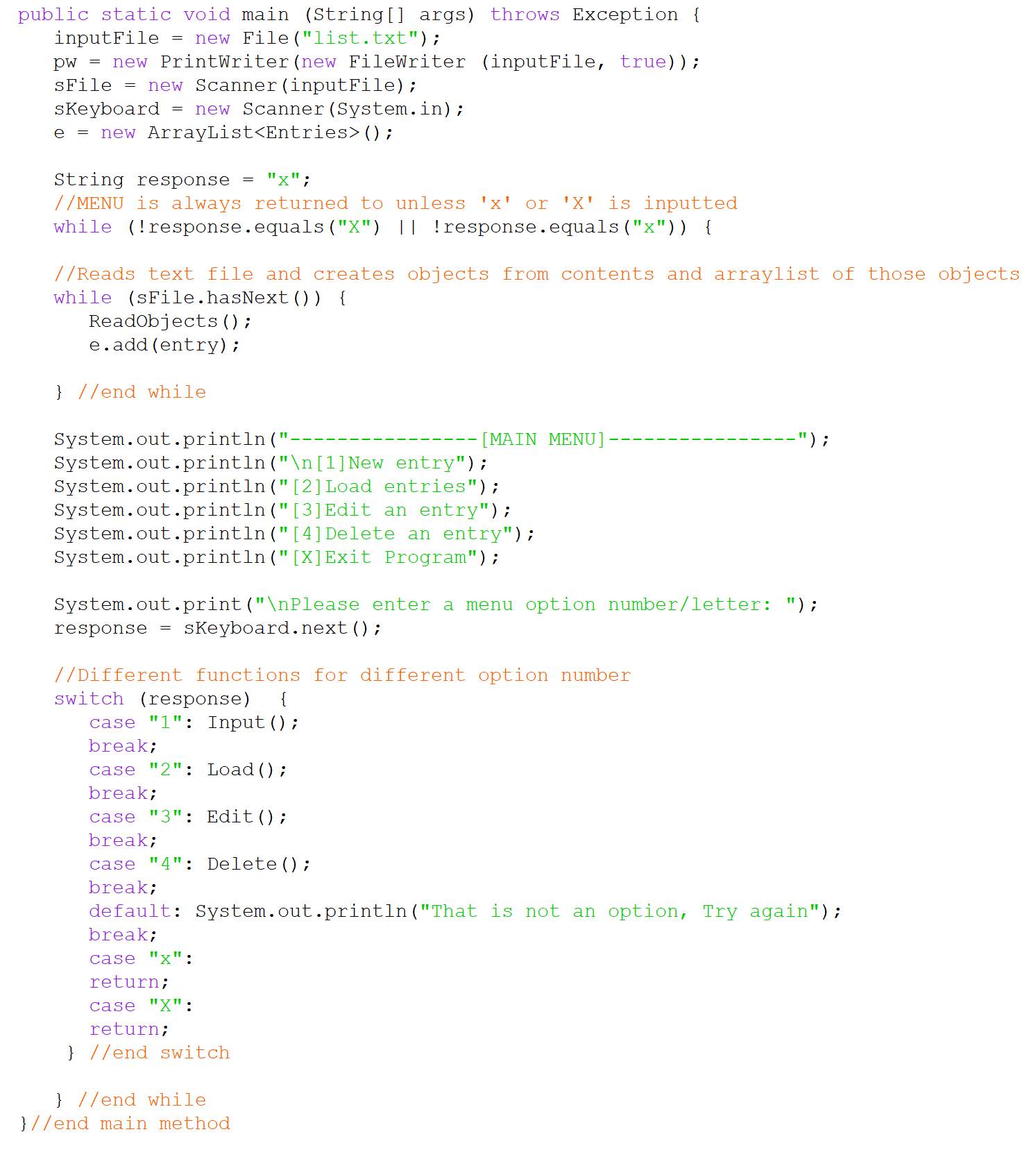
The code is fairly organized into many different methods. The main functions that the user interacts with are identified as public methods, and the calculations and other methods are identified as private methods to be used within the public methods.

Advantage: Each private method has a distinct function, so if another programmer were to change a calculation, it would be easy to access that private method. Additionally, some variables such as PrintWriters, Scanners, etc. are declared static before the main method. This allows for the programmer to access those variables from any where in the different program methods.

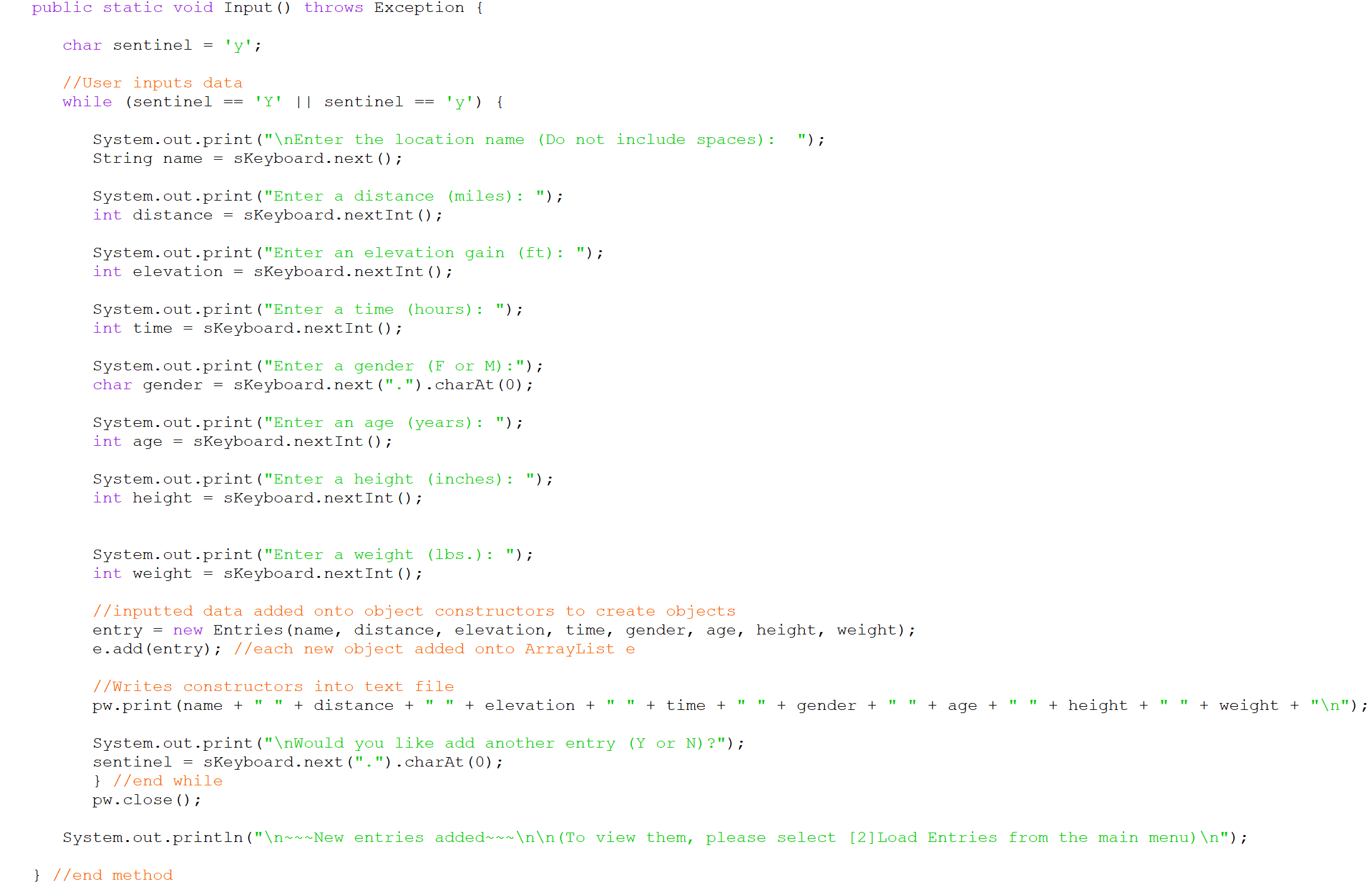
Disadvantage: The program lacks a GUI, so there will have to be a large quantity of print statements in the code.

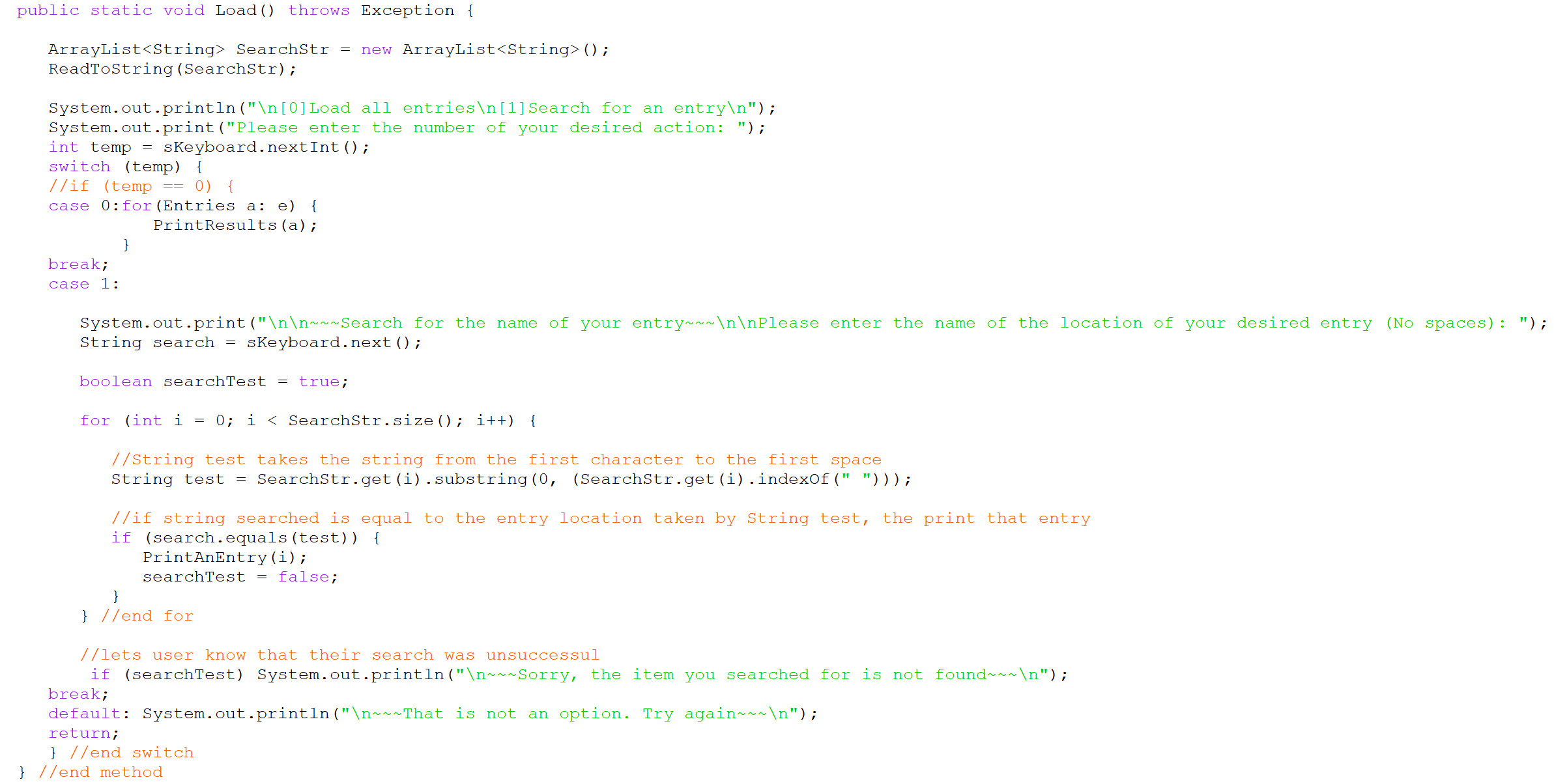
**Screenshots of the main methods of the program:**

main (String[])

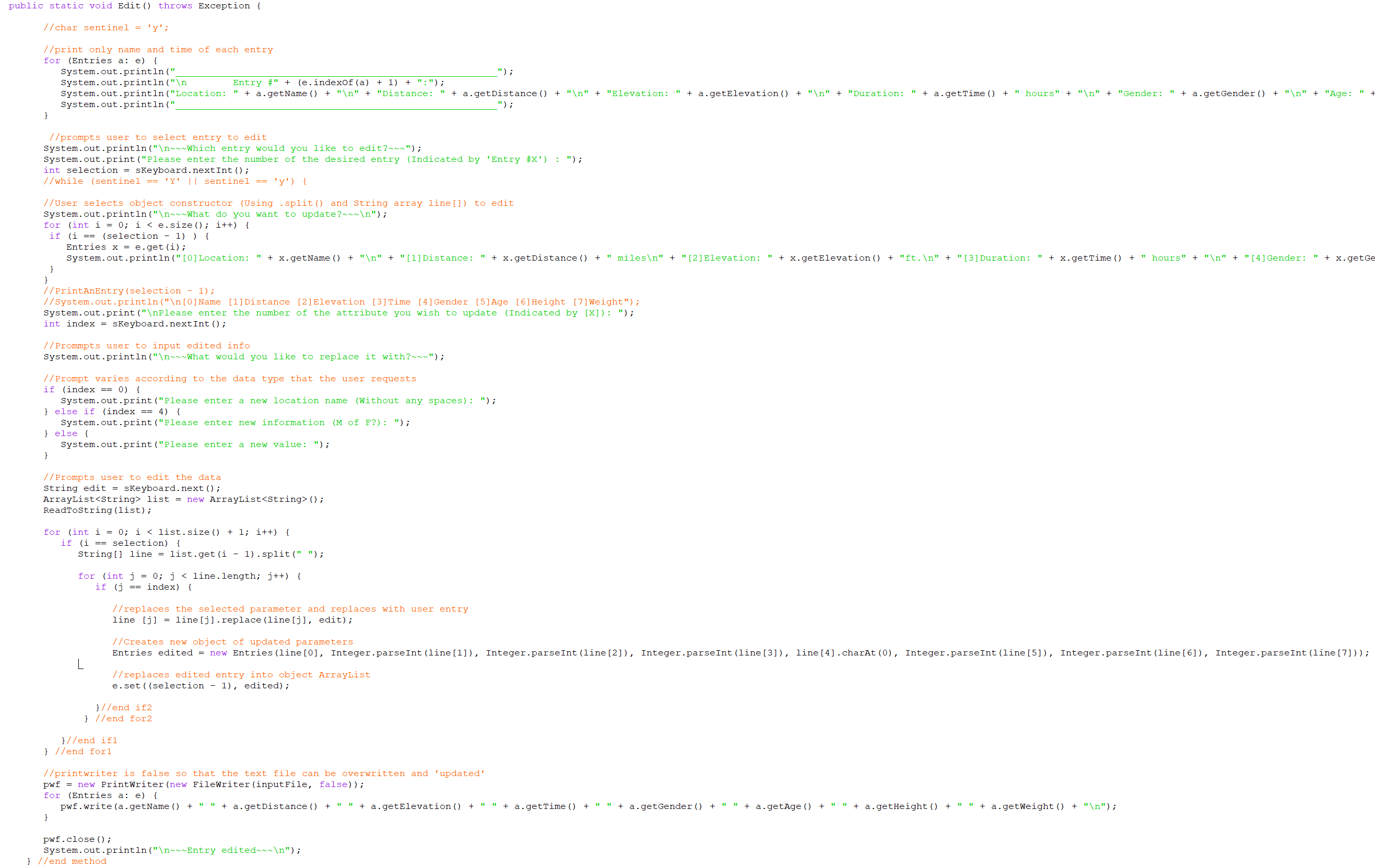


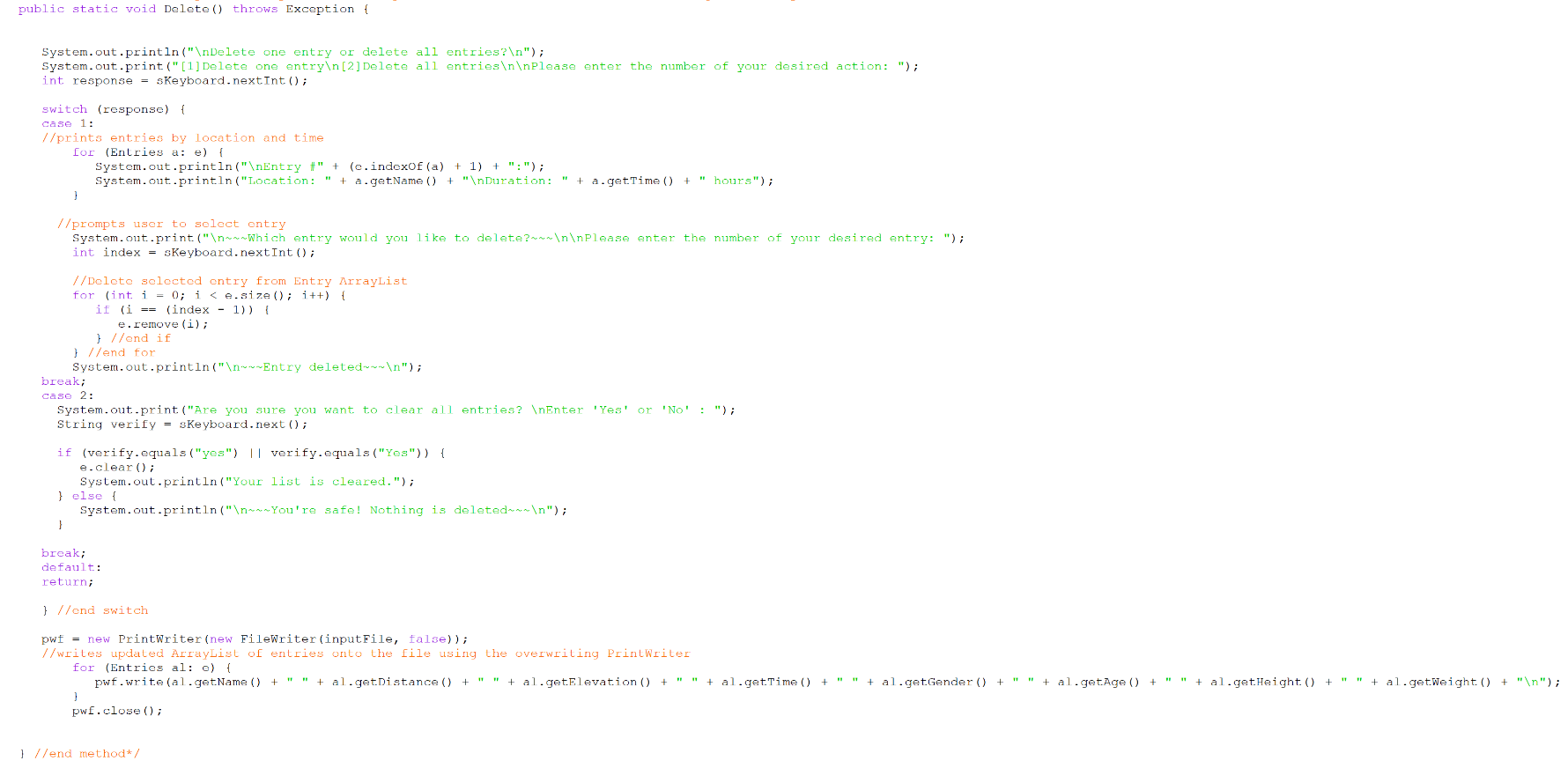
Input():



Load():

Edit():



Delete():

Word Count: 859

**Works Cited (Values for Calculations of BMR and Activity Multiplier)**

“How to Determine Hiking Difficulty.” National Parks Service, U.S. Department of the Interior,

5 Dec. 2017, www.nps.gov/shen/planyourvisit/how-to-determine-hiking-difficulty.htm.

“Harris-Benedict Formula.” HealthFIT, HealthFIT, 2018,

www.healthfitonline.com/resources/harris\_benedict.php.