

Overview

In this first CS1002 practical, I was asked to write a program that converts a time duration inputted in hours into weeks, days, and hours. The output of my code also had to follow a certain format, and I had to use conditional statements to make sure that I was displaying the time properly. Finally, I had to run different tests, some mandatory, others optional, to check that my program was fully functional.

I'm proud to say that I've achieved the previously established goals:

- I've converted the time into weeks, days, and hours
- The output is in accordance with the practical's specifications
- All tests are successful (see **Testing** for more details)

Design

The project is made up of two Java programs:

- On one hand, the Converter.java file contains all the code for performing the conversion and displaying the result.
- On the other hand, the W05Practical.java file deals with the user interface, and uses the methods defined in Converter.java to convert the input and display the converted time.

I will now describe both programs in more detail.

The Converter.java file contains two methods: *convert* and *display*. Both are *void* methods, as they don't return anything. *convert* takes an integer *h* as input (which corresponds to the inputted time in hours), and converts this time in weeks, days, and hours. *display* prints out the converted time, following the practical's format. This method doesn't take any argument or parameter.

Before entering the *convert* method body, I've created 4 public variables: *userInput*, *weeks*, *days*, and *hours*. All four are integers and are initialised to 0, using the *public int userInput = 0;* syntax. What these four variables store is relatively straightforward: *userInput* stores the time that the user inputs (in hours), and *weeks*, *days* and *hours* store the converted time. I've made these variables public as they are used throughout Converter.java, in both the *convert* and *display* methods.

In the *convert* method body, I've used two *while* loops to convert the time. The first *while* loop updates the *week* variable: while the number of hours is greater or equal to 168 (as 1 week = 168 hours), add one to the number of weeks, and subtract 168 to the number of hours. The second *while* loop is very similar to the first one, except that it updates the *days* variable. After executing this method, the *userInput* has been updated and contains the inputted number of hours, and *weeks*, *days*, and *hours* now contain the converted time.

```

C:\Users\antoi>OneDrive\Desktop>ANTOINE> J Converter.java
1 public class Converter {
2     // 4 public variables that are used throughout the different methods
3     public int userInput = 0;
4     public int weeks = 0;
5     public int days = 0;
6     public int hours = 0;
7
8     // The first method: convert
9     // Takes a number of hours as input and stores in the public variables the corresponding time in week(s), day(s) and hour(s)
10    public void convert(int h) {
11        userInput = h;
12        hours = h;
13
14        // Convert the hours into weeks (1 week = 168 hours)
15        while (hours >= 168) {
16            weeks += 1;
17            hours -= 168;
18        }
19
20        // Convert the remaining hours into days (1 day = 24 hours)
21        while (168 > hours && hours >= 24) {
22            days += 1;
23            hours -= 24;
24        }
25    }

```

Figure 1: The four *public* variables and the *convert* method

I could've used Euclidian division to compute *weeks*, *days*, and *hours* without using *while* loops. The integer division of the total number of hours by 168 gives the number of weeks, using either $weeks = h/168$; or $weeks = (int) (h/168)$; The number of days is equal to the remainder of the previous Euclidian division, divided by 24: $days = (h\%168)/24$; The remaining hours are directly computed as the remainder of the Euclidian division of the inputted hours by 24 using the $hours = h\%24$; syntax. I chose to stick with the *while* loops as I find them more intuitive, and they make the code more readable.

In the *display* method, I've used a myriad of conditional statements to display the converted time properly. To get a better understanding of this method, let us look at an example of the required format:

For example, if the input is **169**, the program should run as follows:

```

Enter total number of hours
169
169 hours in weeks, days and hours is:
1 week and 1 hour

```

(not "**1 week, 0 days and 1 hour**" or "**1 weeks and 1 hours**")

The line containing only the value **169** shows the input value as typed by the user.

Figure 2: Example of the required format for the output (source: [Studres](#))

We can see that every output begins with either "[The inputted hours] hour in weeks, days and hours is:" if the user inputted 1 hour, or "[The inputted hours] hours in weeks, days and hours is:" if the user inputted more than 1 hour. To make sure that the output follows the correct plurality, I've used an *if ... else ...* statements, as followed:

```

27 // The second method: display
28 // Prints out/displays the weeks/days/hours time in the required format
29 public void display() {
30     // Checking if the user inputed one or more hours
31     if (userInput == 1) {
32         System.out.println(userInput + " hour in weeks, days and hours is:");
33     }
34     else {
35         System.out.println(userInput + " hours in weeks, days and hours is:");
36     }

```

Figure 3: The first *if ... else ...* statement of the *display* method

I've then created three variables (empty strings), one for each part of the final display: *weeksString*, *daysString*, and *hoursString*. The next step was updating these variables in accordance with the content of *weeks*, *days*, and *hours* respectively. The logic I've used is the same for *weeksString*, *daysString*, and *hoursString* (the only difference is the variable name), so I'll only explain what I did for *weeksString*. To check if there are one or more weeks, I've used an *if ... else if ...* statement: if the number of weeks is exactly one, set *weeksString* to "1 week". Else, if the number of weeks is greater than one, set *weeksString* to the number of weeks, concatenated to " weeks". Notice that my conditional statement doesn't use the traditional *else ...* statement, as we've already covered that part before: if the number of weeks is greater or equal to 0, the computer would've "gone into" the *else* statement only if the number of weeks is equal to zero, to set *weeksString* to an empty string. But as *weeksString* has been initialised as an empty string (and not *null*) at the beginning of the method, the *else* statement here would be redundant.

```
38 // 3 variables (empty strings), one for each part of the final display
39 String weeksString = "";
40 String daysString = "";
41 String hoursString = "";
42
43 // Checking if there is one or more weeks
44 if (weeks == 1) {
45     weeksString = "1 week";
46 }
47 else if (weeks > 1) {
48     weeksString = weeks + " weeks";
49 }
```

Figure 4: Creating and updating the *weeksString* variable

The *display* method ends with a long, tedious yet inevitable *if ... else ...* statement, to display the time properly. The first *if* statement is to check if the number of weeks is greater than 0. If this is the case, then the computer performs another check, using an *if ... else if ... else ...* statement nested in the previous *if* statement:

- If **both** the number of days and the number of hours are greater than 0, then the computer prints out *weeksString*, concatenated to a comma (", "), concatenated to *daysString*, concatenated to " and ", and concatenated to *hoursString*. [We do not have to worry about the plurality anymore, as we've already covered that previously; the only thing that matters here is if there are/aren't weeks (or days or hours) to add the comma and the "and" properly.]
- Else, if the number of days is equal to 0, then we must check if the number of hours is also equal to 0, by using yet another nested *if ... else ...* statement.
 - o If the number of days is also equal to 0, then the computer only prints *weeksString*.
 - o Else, it prints *weeksString* concatenated to " and ", concatenated to *hoursString*.
- Else, the computer prints *weeksString* concatenated to " and ", concatenated to *daysString*. Here, the algorithm knows that the number of hours must be equal to 0: if it isn't the case, then we would've been in the first case (which has already been covered)

After every output, I've used the *System.exit(1);* command to exit the system so that the program stops running and doesn't perform unnecessary checks.

```

67      // Making sure that we are displaying in the correct format:
68      // 1. Not printing out "0 weeks", "0 days" or "0 hours"
69      // 2. Adding "," and "and" when necessary
70      if (weeks > 0) {
71          if (days > 0 && hours > 0) {
72              System.out.println(weeksString + ", " + daysString + " and " + hoursString);
73              System.exit(1); // Exiting the system so that the program stops running
74          }
75          else if (days == 0) {
76              if (hours == 0) {
77                  System.out.println(weeksString);
78                  System.exit(1);
79              }
80              else {
81                  System.out.println(weeksString + " and " + hoursString);
82                  System.exit(1);
83              }
84          }
85          else {
86              System.out.println(weeksString + " and " + daysString);
87              System.exit(1);
88          }
89      }

```

Figure 5.1: The nested conditional statements, if the number of weeks is greater than 0

If the number of weeks isn't greater than 0, we are now in the *else* statement. We are doing here the same checks as in the *if* part of the statement (see previous paragraph and Figure 5.1), so I won't be covering them in detail. However, the outputs differ, as we are not printing *weeksString*; therefore, I will refer to the line numbers in Figure 5.1 and in Figure 5.2 to show the difference between the *System.out.println()*-s in the *if* statement and in the *else* statement:

- Line 92 (as opposed to Line 72): the computer prints out *daysString* concatenated to " and ", concatenated to *hoursString*.
- Line 97 (as opposed to Line 77): the computer prints out "0 hours".
- Line 101 (as opposed to Line 81): the computer prints out *hoursString* only.
- Line 106 (as opposed to Line 86): the computer prints out *daysString* only.

```

90      }
91      else {
92          if (days > 0 && hours > 0) {
93              System.out.println(daysString + " and " + hoursString);
94              System.exit(1);
95          }
96          else if (days == 0) {
97              if (hours == 0) {
98                  System.out.println("0 hours"); // Prints out "0 hours" only if the user inputs 0
99                  System.exit(1);
100              }
101              else {
102                  System.out.println(hoursString);
103                  System.exit(1);
104              }
105          }
106          else {
107              System.out.println(daysString);
108              System.exit(1);
109          }
110      }
111  }

```

Figure 5.2: The end of the *display* method, and the end of the Converter.java file

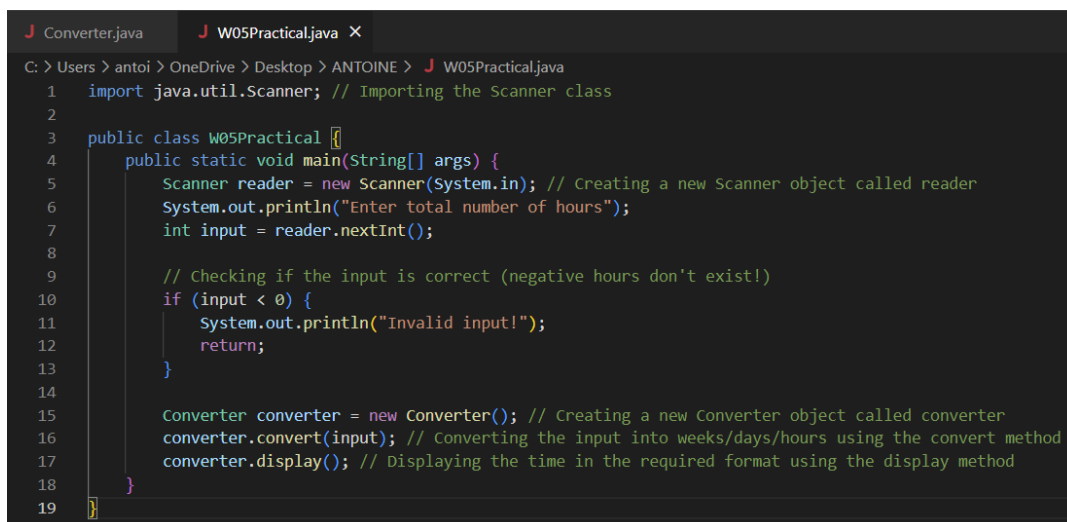
P.S. Updating the *weeksString*, *daysString*, and *hoursString* variables beforehand clearly reduces the amount of *if* statements used throughout the *display* method. The "naïve" algorithm would check if the number of weeks is either equal to 0, equal to 1, or greater than 1. For each case, it would perform the same checks but for the number of days, and for each sub-case, it would perform another 3 checks for the number of hours. This gives us a total of

$3 \times 3 \times 3 = 27$ statements. Meanwhile, my algorithm only has 18 statements (as you can see in Figures 4, 5.1 and 5.2).

W05Practical.java only contains one *main* method. To ask the user the time he/she wants to convert, I've used a *Scanner* object from the *java.util.Scanner* class. To do that, we must import it with the *import java.util.Scanner;* command. We can then create a new *Scanner* object called *reader* in the *main* method, using the *Scanner reader = new Scanner(System.in);* syntax. We can now ask the user to enter the total number of hours with *System.out.println("Enter total number of hours");*. To store the user's input, I've created a variable (integer) called *input*, and I've initialised it to the input, using a predefined method called *nextInt* from the *Scanner* class thanks to the *reader* object. The syntax is the following: *int input = reader.nextInt();*

The only conditional statement used in the W05Practical.java file is to check if the input is correct: therefore, I've used a single *if* statement to check if *input* is smaller than 0. If so, the computer prints out "Invalid input!" and stops the program using the *return* keyword.

Finally, to perform the conversion, I've created a new *Converter* object (from the Converter.java file), called *converter*, with *Converter converter = new Converter();*. I've then used the *convert* method of the *converter* object to convert the inputted time, by using *input* as parameter to the method, and the *display* method of the *converter* object to display the converted time.



```
1 import java.util.Scanner; // Importing the Scanner class
2
3 public class W05Practical {
4     public static void main(String[] args) {
5         Scanner reader = new Scanner(System.in); // Creating a new Scanner object called reader
6         System.out.println("Enter total number of hours");
7         int input = reader.nextInt();
8
9         // Checking if the input is correct (negative hours don't exist!)
10        if (input < 0) {
11            System.out.println("Invalid input!");
12            return;
13        }
14
15        Converter converter = new Converter(); // Creating a new Converter object called converter
16        converter.convert(input); // Converting the input into weeks/days/hours using the convert method
17        converter.display(); // Displaying the time in the required format using the display method
18    }
19 }
```

Figure 6: W05Practical.java

Testing

As briefly explained in the Overview, I've performed two types of tests to make sure that my programs worked. The first type of checks was mandatory: those are the *stacscheck* tests. To run these tests, I had to go to the W05Practical directory and type in the *sctacscheck StacscheckTests* command in the Linux terminal. As you can see in the Figure 7, my programs have passed all 6 *stacscheck* tests:

```
am692@trenco:~/Documents/CS1002/W05Practical/W05Practical
am692@trenco:~/Documents/CS1002/W05Practical/W05Practical $ stacscheck StacscheckTests
Testing CS1002 W05 Practical
- Looking for submission in a directory called 'source': found in current directory
* BUILD TEST - build-all : pass
* COMPARISON TEST - 1_public/prog-run-input_-11.out : pass
* COMPARISON TEST - 1_public/prog-run-input_1.out : pass
* COMPARISON TEST - 1_public/prog-run-input_185.out : pass
* COMPARISON TEST - 1_public/prog-run-input_625.out : pass
* COMPARISON TEST - 1_public/prog-run-input_72.out : pass
6 out of 6 tests passed
am692@trenco:~/Documents/CS1002/W05Practical/W05Practical $
```

Figure 7: The *stacscheck* tests

Additionally, I've manually tested my code by running it multiple times in the VS Code terminal, with different values:

- The first values were those given as examples in the practical's instructions: 169 hours, 185 hours, 625 hours, and 72 hours (see Figure 8.1).

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\antoi> cd OneDrive\Desktop\ANTOINE
PS C:\Users\antoi\OneDrive\Desktop\ANTOINE> javac *.java
PS C:\Users\antoi\OneDrive\Desktop\ANTOINE> java W05Practical
Enter total number of hours
169
169 hours in weeks, days and hours is:
1 week and 1 hour
PS C:\Users\antoi\OneDrive\Desktop\ANTOINE> java W05Practical
Enter total number of hours
185
185 hours in weeks, days and hours is:
1 week and 17 hours
PS C:\Users\antoi\OneDrive\Desktop\ANTOINE> java W05Practical
Enter total number of hours
625
625 hours in weeks, days and hours is:
3 weeks, 5 days and 1 hour
PS C:\Users\antoi\OneDrive\Desktop\ANTOINE> java W05Practical
Enter total number of hours
72
72 hours in weeks, days and hours is:
3 days
PS C:\Users\antoi\OneDrive\Desktop\ANTOINE>
```

Figure 8.1: Tests for 169, 185, 625 and 72

- I've also tested for two specific values: I wanted to check if my program returns "0 hours" if I input 0 (the special case), "1 hour" if I input 1 (the plurality), and "Invalid Input" if I input a negative number (for example -5).

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\antoi\OneDrive\Desktop\ANTOINE> java W05Practical
Enter total number of hours
0
0 hours in weeks, days and hours is:
0 hours
PS C:\Users\antoi\OneDrive\Desktop\ANTOINE> java W05Practical
Enter total number of hours
1
1 hour in weeks, days and hours is:
1 hour
PS C:\Users\antoi\OneDrive\Desktop\ANTOINE> java W05Practical
Enter total number of hours
-5
Invalid input!
PS C:\Users\antoi\OneDrive\Desktop\ANTOINE>
```

Figure 8.2: Tests for 0, 1 and -5

Evaluation

I am proud to say that my program works: it can convert an inputted time in hours into weeks, days, and hours. I've tested my program successfully, with various examples. I am aware that my code isn't the most efficient, and there are probably some less time-consuming ways of completing the required task, but this CS1002 Practical being my first major Java assignment, I believe I do not have the knowledge yet to make a more simplified code.

Conclusion

All in all, I have found this assignment a bit challenging at first, as I didn't know what to do at first. But by taking my time and being methodical, I figured a way to complete my first CS1002 practical.

If I had some extra time, I would've loved to push the challenge even further, by making the user input a decimal number of hours, and converting it in weeks, days, hours, minutes and even seconds. I can imagine that the logic behind the code would be quite similar, but the *if* statements in the *display* method would be too long and complicated to code, so I would have to find a more optimal way to check whether there are 0, 1 or more weeks/days/hours.

I believe there's a way to deal with similar problems using lists and arrays. For example, I can guess that storing the time in a list (the first element is the number of weeks, the second the number of days, ...), then going through the list and performing identical checks for every entry might drastically reduce the number of conditional statements used in my code.