LEARNING PROFILE FOR ASSIGNMENT#0, QUESTION#1

CelsiusToFahrenheit.java

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Name* | *:* | *Tyler Lucas* | *Due Date* | *:* | *N/A* |
| *Student ID* | *:* | *3305203* | *Submission Date* | *:* | *2017/04/18* |

# 1. Problem Statement

Write a java program that allows user to input a temperature in Celsius (integer) and then the program prints the result in Fahrenheit (integer).

# 2. Description of the Code

Outputs temperature in Fahrenheit given an input temperature in Celsius.

# **3.** Errors and Warnings

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | CelsiusToFahrenheit class wasn’t found in CelsiusToFahrenheit project. | [v. 1.0] I had set the main class as "private". | I changed the class and main method from “private” to “public”. |
| 2 | CelsiusToFahrenheit class wasn’t found in CelsiusToFahrenheit project. | [v. 1.0] I made a type with "String[]". | I corrected the misspelled “Stringp[]” to “String[]”. |
| 3 | error: variable degC is already defined in method main(String[])  int degC = sc.nextInt(); | [v. 1.2] I accidently wrote "int" again, in front of degC = … | I deleted "int ". |
| 4 | Exception in thread "main" java.util.InputMismatchException | [v. 1.2] Scanner integer input fetcher returned this error when given a float. | This was left as is / as designed. |
|  |  |  |  |

# 4. Sample Input and Output

[Version 1.2-1.3, input “0”]

Enter temperature in integer degrees Celsius: 0

0 degrees Celsius is equivalent to 32 degrees Fahrenheit.

[Version 1.2-1.3, input “60.5” (float type input)]

Enter temperature in integer degrees Celsius: 60.5

Exception in thread "main" java.util.InputMismatchException

at java.util.Scanner.throwFor(Scanner.java:864)

at java.util.Scanner.next(Scanner.java:1485)

at java.util.Scanner.nextInt(Scanner.java:2117)

at java.util.Scanner.nextInt(Scanner.java:2076)

at CelsiusToFahrenheit.main(CelsiusToFahrenheit.java:34)

C:\Users\tyblu\Documents\repos\comp268-random\CelsiusToFahrenheit\nbproject\build-impl.xml:1040: The following error occurred while executing this line:

C:\Users\tyblu\Documents\repos\comp268-random\CelsiusToFahrenheit\nbproject\build-impl.xml:805: Java returned: 1

# 5. Discussion

The first error, where a class couldn’t be found in the project, was caused by setting either or both the class and the main method to private. I first ran into this error when attempting the *HelloWorld* sample program[[1]](#footnote-1). Having read the class Style Guide in which it says “Create private fields with getters/setters rather than leaving fields public,” as well as Controlling Access to Members of a Class[[2]](#footnote-2) in which it says “Use private unless you have a good reason not to,” I mistakenly thought this applied to the main class and method as well, as I’m still not sure what the differences are between a class, method, and object. Searching online didn’t reveal a solution right away, as few experienced programmers would think that something this simple could go awry, but I eventually found the answer here: <https://goo.gl/P2OdMJ>. Of course, the next page in the textbook had the answer as well:

The word “public” in the first line of main() means that this routine can be called from outside the program. This is essential because the main() routine is called by the Java interpreter, which is something external to the program itself.[[3]](#footnote-3)

I’ll have to keep an eye out to see how to implement other access levels at these top levels.

The fourth error was left as it is, returning an error, essentially ‘breaking’ the program if the user input the wrong type (non-integer). Though not ‘graceful’, it is an effective way of dealing with errors: let the JVM and OS handle them with their default methods. The alternative is to write your own error handling algorithm, attempting to predict what unwanted states the system may end up in or what unwanted inputs the user may enter. It could be quite simple; for example: checking to see if the input is an integer, and, if not, exiting with an explanation or repeating the request for an integer input. It could be quite complex; for example: having the program double-check and attempt to correct the states of pertinent variables for memory errors such as overflow or corruption. In the case of a user inputting an incorrect data type, like a float instead of an integer, leaving errors to the JVM and OS does not give them the specific feedback they may need to correct their behaviour, but it does leave the topic of error detection and handling to the tried-and-true algorithms developed by teams of professional programmers. Generally, I will keep error handling to a bare minimum, restricted to cases when I am confident in the use of the pertinent methods.

1. (Eck, 2014, p. 21) [↑](#footnote-ref-1)
2. (Oracle, 2015) [↑](#footnote-ref-2)
3. (Eck, 2014, p. 22) [↑](#footnote-ref-3)