LEARNING PROFILE FOR ASSIGNMENT#1, QUESTION#7

HailstoneSequence.java

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Name* | *:* | *Tyler Lucas* | *Due Date* | *:* | *N/A* |
| *Student ID* | *:* | *3305203* | *Submission Date* | *:* | *2017/05/09* |

# 1. Problem Statement

Douglas Hofstadter’s Pulitzer-prize-winning book *Gödel, Escher, Bach* contains many interesting mathematical puzzles. In Chapter XII, Hofstadter mentions a wonderful problem that is well within the scope of the control statements in Java. The problem can be expressed as follows: Pick some positive integer and call it *n*. If *n* is even, divide it by two. If *n* is odd, multiply it by three and add one. Continue this process until *n* is equal to 1. Hofstadter illustrates this process with the following example, starting with the number *n* = 15:

15 is odd, so I make 3n+1: 46

46 is even, so I take half: 23

23 is odd, so I make 3n+1: 70

70 is even, so I take half: 35

35 is odd, so I make 3n+1: 106

106 is even, so I take half: 53

53 is odd, so I make 3n+1: 160

160 is even, so I take half: 80

80 is even, so I take half: 40

40 is even, so I take half: 20

20 is even, so I take half: 10

10 is even, so I take half: 5

5 is odd, so I make 3n+1: 16

16 is even, so I take half: 8

8 is even, so I take half: 4

4 is even, so I take half: 2

2 is even, so I take half: 1

As you can see from this example, the numbers go up and down, but eventually—at least for all numbers that have ever been tried—come down to end in 1. In some respects, this process is reminiscent of the formation of hailstones, which get carried upward by the winds over and over again before they finally descend to the ground. Because of this analogy, this sequence of numbers is usually called the *Hailstone sequence*, although it goes by many other names as well.

Write a program that reads in a number from the user and then displays the Hailstone sequence for that number, followed by a line showing the number of steps taken to reach 1.

# 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”. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 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

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