Objectives

To continue practicing the use of GitHub to access the files for a practical assignment. Additionally, to practice using the Ubuntu operating system and software development programs such as a “terminal window” and the “Atom text editor”. You will continue to practice using Slack to support communication with the teaching assistants and the course instructor. Next, you will write a program to determine if certain events occurred during the year input from the file. For this task you will use if/else statements and boolean logic operators such as “&&” or “||”.

Reading Assignment

If you have not done so already, please read all of the relevant “GitHub Guides”, available at https://guides.github.com/, that explain how to use many of the GitHub’s features. In particular, please make sure that you have read guides such as “Mastering Markdown” and “Documenting Your Projects on GitHub”; each of them will help you to understand how to use both GitHub and GitHub Classroom. Focusing on the content about creating and using Java objects and writing conditional logic, you should review Chapters 1 through 3 and Sections 5.1 and 5.3 in the textbook.

Using Conditional Logic in a Java Program

To access the practical assignment, you should go into the #practicals channel in our Slack team and find the announcement that provides a link for it. You should accept the practical assignment and see that GitHub Classroom created a new GitHub repository for you to access the assignment’s starting materials and to store the completed version of your assignment. Unless you provide the instructor with documentation of the extenuating circumstances that you are facing, not accepting the assignment means that you automatically receive a failing grade for it.

This practical assignment is inspired by a quotation is from Allegheny College’s Alma Mater. In the class YearChecker.java, you will write statements that for the input year stored in the year_input file, determine which of the following events occurs that year:

- it is designated as a leap year
- the emergence of the 17-year cicadas (more specifically, Brood II)
- it is predicted to be a peak year of sunspot activity

A year is a leap year if it is divisible by 4, unless it is a century year. If it is a century year, it is a leap year if it is divisible by 400. For instance, 1968 and 1972 are leap years since they are divisible by 4; 1967 and 1970 are not. The year 2000 is a leap year because it is divisible by 400; however, 1900 is not (even though 1900 is divisible by 4—century years are treated differently).

The 17-year cicadas emerge from underground every 17 years. There are several “broods”—the one in which we have interest emerged most recently in 2013. That is, any year that differs from 2013 by a multiple of 17 is also an emergence year for Brood II (e.g., 2040, 1996, 1928, and 3713). Sunspot activity usually peaks every 11 years. The year 2013 was supposed to be such a “solar max” year. This means that any year that differs from 2013 by a multiple of 11 should also be a solar max year. For instance, 2002, 2024, and 1793 are all years predicted to exhibit peak activity.
I will read in a year between 1000 and 3000.
Okay, I read in the year 1452.

Is the year 1452 a leap year? Yes
Is the year 1452 a cicada emergence year? Yes
Is the year 1452 a year of peak sunspot activity? Yes

Thank you for using the YearChecker.

Figure 1: Sample “YearChecker” output featuring output from conditional logic checks.

Checking the Correctness of Your Program and Writing

As in the past assignments, you are provided with an automated tool for checking the quality of your source code. Please note that the practical assignments do not require you to produce a writing document as you do in the laboratory assignments. However, to check your Java source code you can started with the use of GatorGrader, type the command "./gatorgrader.sh --start" in your terminal window. Once this step completes you can type "./gatorgrader.sh --check". If your work does not meet all of the assignment’s requirements, then you will see the following output in your terminal: “Overall, are there any mistakes in the assignment? Yes”. If you do have mistakes in your assignment, then you will need to review GatorGrader’s output, find the mistake, and try to fix it. Specifically, don’t forget to add in the required comments! If you are having trouble running GatorGrader locally, don’t forget to ensure that you still transfer all of your source code to GitHub. Please see the course instructor if you have questions about this step.

Once your program is building correctly, fulfilling at least some of the assignment’s requirements, you should transfer your files to GitHub using the “git commit” and “git push” commands. For example, if you want to signal that the src/main/java/practicalseven/YearChecker.java file has been changed and is ready for transfer to GitHub you would first type “git commit src/main/java/practicalseven/YearChecker.java -m "Your descriptive commit message"” in your terminal, followed by typing “git push” and checking to see that the transfer to GitHub is successful. If you notice that transferring your code to GitHub did not work correctly, then please try to determine why, asking a teaching assistant or the course instructor for help, if necessary.

After the course instructor enables “continuous integration” with a system called Travis CI, when you use the “git push” command to transfer your source code to your GitHub repository, Travis CI will initialize a “build” of your assignment, checking to see if it meets all of the requirements. If both your source code and writing meet all of the established requirements, then you will see a green ✓ in the listing of commits in GitHub after awhile. If your submission does not meet the requirements, a red ✗ will appear instead. If the green ✓ appears on the last commit in your GitHub repository, then you satisfied all of the main checks.
This assignment invites you to submit, using GitHub, the following deliverables. Because this is a practical assignment, you are not required to complete any technical writing.

1. A correct version of `src/main/java/practicalseven/YearChecker.java` that meets all of the established source code requirements and produces the desired text-based output.

**Evaluation of Your Practical Assignment**

Practical assignments are graded on a completion — or “checkmark” — basis. If your GitHub repository has a ✔️ for the last commit before the deadline then you will receive the highest possible grade for the assignment. However, you will fail the assignment if you do not complete any part of the assignment by the set deadline for completing the project. Please see the course instructor if you do not understand how practical assignments are graded or you do not know how to complete one of the specific tasks in this assignment.

**Adhering to the Honor Code**

In adherence to the Honor Code, students should complete this practical assignment on an individual basis. While it is appropriate for students in this class to have high-level conversations about the assignment, it is necessary to distinguish carefully between the student who discusses the principles underlying a problem with others and the student who produces assignments that are identical to, or merely variations on, someone else’s work. Deliverables (e.g., the Java source code) that are nearly identical to the work of others will be taken as evidence of violating the Honor Code. Please see the course instructor if you have questions about this policy.