Objectives

To continue practicing the use of GitHub to access the files for a practical assignment. Additionally, to practice using software development program such as a “terminal window”. You will continue to practice using Slack to support communication with the technical leaders and the course instructor. Next, you will learn how to fix a Java program, further discovering how the course’s automated grading tool assesses your progress towards correctly completing the project. Finally, you will continue to learn more about variables and data types and the errors that may arise when variables have the wrong types.

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 features that GitHub provides. 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. To do well on this assignment, you should also read Chapters 1 and 2 in the course textbook, ensuring that you fully understand all of the concepts that we discussed during class and investigated during prior practical and laboratory sessions. Please see the instructor or one of the technical leaders if you have questions these reading assignments.

Finding and Fixing Defects in Programs

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. Copy this link and paste it into your web browser. Now, 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. Specifically, to access your new GitHub repository for this assignment, please click the green “Accept” button and then click the link that is prefaced with the label “Your assignment has been created here”. If you accepted the assignment and correctly followed these steps, you should have created your own practical 2 GitHub repository. 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. Please see the instructor if you are stuck on getting started.

Now, study the documentation in the provided source code to understand the type of output that your program should produce. Note that the provided source code contains defects in it! You are responsible for finding and fixing all of these defects. Once you locate and resolve one of these issues, please put a comment into the code to explain the problem that you found and the way in which you decided to handle it. As you complete this practical assignment, make sure that you regularly commit your code to GitHub and use descriptive messages that say what you fixed.
Testing your Program

After you have made some modifications to the program you can build and run your program and check its correctness! If you have all needed software installed on your machine (Windows Home users) you can type “gradle grade” to use the automated tool for checking the quality of your source code, or you can use the commands “gradle build” and “gradle run” to compile and execute your program.

If you are using Docker Desktop, you can use the following “docker run” command to start “gradle grade” as a containerized application, using the “DockaGator” Docker image available on DockerHub. You can run the following command to run the “gradle grade” on your project:

```sh
docker run --rm --name dockagator \
  -v "$(pwd)":/project \
  -v "${HOME}/dockagator":/root/.local/share \
  gatoreducator/dockagator
```

The aforementioned command will use "$(pwd)" (i.e., the current directory) as the project directory and "${HOME}/dockagator" as the cached GatorGrader directory. Please note that both of these directories must exist, although only the project directory must contain some content. Generally, the project directory should contain the source code and technical writing for this assignment, as provided to you through GitHub during the completion of a previous step. Additionally, the cache directory should not contain anything other than directories and programs created by DockaGator, thus ensuring that they are not otherwise overwritten during the completion of the assignment. To ensure that the previous command will work correctly, you should create the cache directory by running the command “mkdir $HOME/.dockagator”; you will only need to do this once. If the above “docker run” command does not work correctly on the Windows operating system, then you may need to instead run the following command to work around limitations in the terminal window:

```sh
docker run --rm --name dockagator \
  -v "$(pwd)":/project \
  -v "${HOME}/dockagator":/root/.local/share \
  gatoreducator/dockagator /bin/bash
```

To enter into an “interactive terminal” in the Docker container, you can instead use the following command

```sh
docker run -it --rm --name dockagator \
  -v "$(pwd)":/project \
  -v "${HOME}/dockagator":/root/.local/share \
  gatoreducator/dockagator /bin/bash
```

Now, if you want to “build” your program you can type the command “gradle build” in your terminal, thereby causing the Java compiler to check your program for errors and get it ready to run. If you get any error messages, go back into your text editor and try to figure out what you mis-typed and fix it. Once you have solved the problem, make a note of the error and the solution for resolving it. Re-save your program and then build it again by re-running the “gradle build”.

Submission to GitHub

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/practicaltwo/ComputeMoonDistance.java file
has been changed and is ready for transfer to GitHub you would first type:
“git commit src/main/java/practicaltwo/ComputeMoonDistance.java -m "your 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 or writing to GitHub did not work correctly, then please try to determine why, asking a teaching assistant or the course instructor for help, if necessary.

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 your source code meets 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. You should aim to finish practical assignments on the day that they are assigned; please see the instructor if you do not understand this policy.

Summary of the Required Deliverables
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 properly documented, well-formatted, and correct version of src/main/java/practicaltwo/ComputeMoonDistance.java that both meets all of the established requirements and produces the desired output. In addition to your Java source code no longer containing any defects that “break the build”, it must feature comments that explain each of the mistakes that you found and a clear statement of how you fixed them.

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 have completed the assignment to the highest possible level. In order to receive a “checkmark” for this assignment you need to commit and push your work (even if incomplete) before the stated deadline. 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.

Additional Success Strategies for the Practical Sessions
Since you are still learning how to use the Java programming language, don’t become frustrated if you make a mistake. Instead, use your mistakes as an opportunity for learning both about the necessary technology and the background and expertise of the other students in the class, the teaching assistants, and the course instructor.

- **Experiment.** Practical sessions are for learning by doing without the pressure of grades or “right/wrong” answers. So try things!
- **Practice Key Laboratory Skills.** As you are completing this assignment, practice using the terminal and git until you can easily use their most important features.
- **Help One Another!** If your neighbor is struggling and you know what to do, offer your help. Don’t “do the work” for them, but advise them on what to type or how to handle things. If you are stuck on a part of this practical assignment and you could not find any insights in either your textbook or online sources, formulate a question to ask your neighbor, a technical leader, or a course instructor. Try to strike the right balance between asking for help when you cannot solve a problem and working independently to find a solution.