Objectives

To continue practicing the use of GitHub to access the files for a practical assignment. You will continue to practice using Slack to support communication with the teaching assistants and the course instructor. Next, you will learn more about file input and output, 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 ways in which they are correctly combined.

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. Focusing on the content about declaring and variables and reading from the terminal, you should review Chapters 1 and 2 and Sections 3.1 and 3.2 in the textbook.

Implementing a “Mad Libs” 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. 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”. 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.

Figure 1 contains the output from running a program like the one you must implement. First, you must read in user’s input in the “input” directory, save them into appropriate variables and display them in the terminal. Then you have to come up with your own Mad Libs story while using each of the words obtained from the user via terminal. To gain some inspiration and to learn more about Mad Libs and how your program should behave, please visit the site http://www.madlibs.com/. After finishing the src/main/java/practicalfour/MadLibs.java file, you should repeatedly test your program to make sure that it is creating the correct textual output. This will involve you typing different user inputs so that it contains different words and numbers in it and then running your program to determine if it correctly displays the story.

Testing your Program

When you have written you are ready to use tools that build and run your program! If you are using Docker Desktop, you can use the following “docker run” command to start “gradle grade”
Third-Grade Word Problem

If you own 32 verboten noggins, and you purchase 42 glitzy noggins, how many more noggins do you need to snooze 5.43 noggins?

Answer: You need -68.57 more noggins.

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:

```bash
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:

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

To enter into an “interactive terminal” in the Docker container, you can instead use the following command.
docker run -it --rm --name dockagator \
  -v "/$(pwd)":/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 atom 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”. If you cannot build MadLibs correctly, then please talk with a technical leader or the instructor.

When all of the errors are eliminated, you can run your program by typing “gradle run” in the terminal window—this is the “execute” step that will run your program and produce the designated output.

Checking the Correctness of Your Program and Writing

To check your Java source code you can started with the use of GatorGrader, type the command “gradle grade” in your terminal window. 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/practicalfour/MadLibs.java file has been changed and is ready for transfer to GitHub you would first type “git commit src/main/java/practicalfour/MadLibs.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.

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 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 correct version of src/main/java/practicalfour/MadLibs.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. 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.

**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.