Objectives

To practice using GitHub to access the files for a lab assignment. Additionally, to practice using the software development program such as a “terminal window”. You will also continue to use Slack to support communication with the teaching assistants and the course instructor. Next, you will learn how to implement a Java program and also discover how to use the course’s automated grading tool to assess your progress towards correctly completing the project. Finally, you will explore how to perform character output and to use escaped characters.

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, paying particularly close attention to Sections 1.5 and 2.1. Please see the instructor or one of the teaching assistants if you have questions these reading assignments.

Display Character-Based Artwork

To access the lab assignment, you should go into the #labs 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 lab 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.

Study the documentation in the provided source code to understand the type of output that your program should produce. Note the constraints on the character symbols that should appear in the output and the number of possible println statements you can use to produce the output. When you are finished typing text in your text editor, make sure to save your program. Leaving the text editor’s window open, go back to your terminal window.

Testing your Program

Now, you are ready to use tools that build and run your program and check its correctness! 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:
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:

```
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 "$(pwd)":/project \
  -v "$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”. If you cannot build DisplayArtwork 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. You should see your name, today’s date and your character art. Once the program runs, please reflect on this process. What step did you find to be the most challenging? Why? You should write your reflections in a file, called writing/reflection.md, that uses the Markdown writing language. You can learn more about Markdown by viewing the aforementioned GitHub guide. To complete this aspect of the assignment, you should write one high-quality paragraph that reports on your experiences.

Since this is our second laboratory assignment and you are still learning how to use the appropriate hardware and software, 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 technical leaders, and the course instructor.
Checking the Correctness of Your Program and Writing

**Study style guides:** As verified by Checkstyle tool, the source code for the DisplayArtwork file must adhere to all of the requirements in the Google Java Style Guide available at https://google.github.io/styleguide/javaguide.html. The Markdown file that contains your reflection must adhere to the standards described in the Markdown Syntax Guide https://guides.github.com/features/mastering-markdown/. Instead of requiring you to manually check that your deliverables adhere to these industry-accepted standards, the GatorGrader tool that you will use in this laboratory assignment makes it easy for you to automatically check if your submission meets the standards for correctness. For instance, GatorGrader will check to ensure that DisplayArtwork produces exactly four lines of output and that you use the “new Date()” construct in the Java source code to cause the program to display today’s date.

**Write useful commits:** To get started with the use of GatorGrader, type the command “gradle grade” in your terminal window. If your laboratory work does not meet all of the assignment’s requirements, then you will see a summary of the failing checks along with a statement giving the percentage of checks that are currently passing. 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. 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/labtwo/DisplayArtwork.java file has been changed and is ready for transfer to GitHub you would first type in your terminal:

```
$ git commit src/main/java/labtwo/DisplayArtwork.java -m "Your commit message"
```

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 read the errors in your terminal window and please try to determine why, asking a technical leader or the course instructor for assistance, if necessary.

**Verify all checks:** 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. The instructor will reduce a student’s grade for this assignment if the red ✗ appears on the last commit in GitHub immediately before the assignment’s due date. Yet, if the green ✓ appears on the last commit in your GitHub repository, then you satisfied all of the baseline checks, thereby allowing the course instructor to evaluate other aspects of your source code and writing, as further described in the “Evaluation” section of this assignment sheet. Unless you provide the instructor with documentation of the extenuating circumstances that you are facing, no late work will be considered towards your grade for this laboratory assignment.

Summary of the Required Deliverables

This assignment invites you to submit, using GitHub, the following deliverables.

1. A properly documented, well-formatted, and correct version of src/main/java/labtwo/DisplayArtwork.java that both meets all of the established requirements and produces the desired output.

2. Stored in writing/reflection.md, a reflection on the commands that you typed in the terminal window and the Java source code that you wrote in your text editor. This Markdown-based
document should explain the input, output, and behavior of each command (e.g., “gradle build”) and the challenges that you confronted when using it. For every challenge that you encountered, please explain your solution for it.

Evaluation of Your Laboratory Assignment

Using a report that the instructor shares with you through the commit log in GitHub, you will privately receive a grade on this assignment and feedback on your submitted deliverables. Your grade for the assignment will be a function of the whether or not it was submitted in a timely fashion and if your program received a green ✔ indicating that it met all of the requirements. Other factors will also influence your final grade on the assignment. In addition to studying the efficiency and effectiveness and documentation of your Java source code, the instructor will also evaluate the correctness of your technical writing. If your submission receives a red ✗, the instructor will reduce your grade for the assignment. Finally, please remember to read your GitHub repository’s README.md file for a description of the four grades that you will receive for this laboratory assignment.

Adhering to the Honor Code

In adherence to the Honor Code, students should complete this 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., Java source code or Markdown-based technical writing) 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.