Objectives

To continue practicing the use of GitHub to access the files for a practical assignment and the use of a “terminal window” to compile and run Java programs. To analyze various Java programs and modify them to fix the errors and to implement small new features to the program. To experiment with static methods, various loop structures.

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 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. To learn more about the concepts associated with writing static methods, using different types of loops, and utilizing switch statements and arrays, please study the content in Chapters 6, 7 and 8.

Analyzing and Modifying Given Programs

After you have accepted the practical assignment and cloned your repository for it, locate three Java programs inside the “src” directory in your practical’s repository. Study each program and make sure you understand what each program is attempting to do. Pay a close attention to the programs, where the methods are written in and called from the same class. For example, printOne method in the CallingMethodsInSameClass class is called from CallingMethodsInSameClass class. printOne is a static method, which allows us to call it without creating an instance of the object first. To call the methods from within the same class, you need to specify the method name and parameters if any (as is done with the printOne method); if you call static methods from another class, you need to specify the class name then use a dot operator followed by the method name (e.g., Math.pow(4,8)).

All of the given programs contain one or more errors, which may be compile-time errors found when you compile the program or run-time errors found when you run the program. Locate all of the errors and fix them to get all of the programs to compile and run.

Next, find // TODO: comment lines in each program and add Java statements to satisfy the tasks specified in the // TODO:. Compile and run each modified program and save the output.

You should test each program one at a time by running the Gradle’s run command. It is currently configured to run the CallingMethodsInSameClass class. To change the Gradle set up to run a different class, you should open build.gradle file and uncomment the appropriate line that starts with “//mainClassName = 'practicalten.'”.
Testing your Program

Once you have written parts of your program, 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” 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:

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:

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 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 InvestigateExceptions 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

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 start 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 method!

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.

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. Since this is another challenging practical assignment and you are continuing to learn how to repeat actions with while loops, 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.

Summary of Required Deliverable

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

1. A properly documented and completed version of the CallingMethodsInSameClass program.
2. A properly documented and completed version of the LoopStyles program.
3. A properly documented and completed version of the ArrayDay program.

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. Finally, remember that, in adherence to the Honor Code, students should complete this practical assignment on an individual basis. Please see the course instructor if you have any questions about this course policy.