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 create and then call methods that determine if the password read from the file satisfies certain requirements. For this task you will use if/else statements and possibly boolean logic operators such as “&&” or “||”. Finally, you will continue to learn more about creating and using Java classes and objects as you create a password checker.

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 4 and Sections 5.1 and 5.4 in the textbook.

Implementing and Evaluating a Password Checker

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 invites you to create a program that will check an input password. In one of the classes for this assignment, PasswordChecker.java, you will write methods that, for the file’s input from PasswordCheckerMain.java, determine if the chosen password:

- Contains ten or more characters.

- The first character is an upper-case letter. Please note that you should look up a wrapper class for char called Character in the Java API’s java.lang package. Specifically, the method called isUpperCase(char ch) maybe useful for this step.

- The last character is a number. The method called isDigit() in the Character class maybe useful for this step.

Figure 1 contains the output from running a program like the one you must implement. You should study the comments in the src/main/java/practicaleight/PasswordCheckerMain.java to see each step that you have to implement. You should also look at the src/main/java/practicaleight/PasswordChecker.java to see the methods that will ultimately contain the conditional logic. After
I will read in a password from a file.
Okay, I read in the password "computerscience2017".

Is the password "computerscience2018" a valid password? No

Thank you for using the PasswordChecker.

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

finishing the both of these files, you should repeatedly test your program to make sure that it is creating the correct textual output. This will involve you editing the input file and then building and running the program and checking the output to ensure that it produces different values and that the checks are correct. Don’t forget that this assignment requires you to understand and edit two different files called src/main/java/practicaleight/PasswordCheckerMain.java and src/main/java/practicaleight/PasswordChecker.java. This means that you must have correct formatting and documentation in both of these files; check the README.md file for a statement of other checks. You should also be able to draw a diagram explaining the relationship between the two files.

When you are testing your src/main/java/practicaleight/PasswordCheckerMain.java, please make sure that you try both valid and invalid passwords. For instance, the output in Figure 1 shows how the program should work when you input an invalid password. What does the program’s output look like when you input a valid password? Finally, you should notice that there are several ways in which you could enhance this password checker. For example, while the program currently accepts a single password, it would be more useful if it could read and check a list of passwords. Although it is not a requirement of this assignment, you should think about how to add this feature. Moreover, the output in Figure 1 shows that the program currently does not explain why the password is invalid. Again, even though it is not a requirement for this assignment, you should think about how to enhance the PasswordCheckerMain so that it can report precisely why the password is not valid. Please see the instructor with questions about how to add these extra features.

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/practicaleight/PasswordCheckerMain.java file has been changed and is ready for transfer to GitHub you would first type “git commit src/main/java/practicaleight/PasswordCheckerMain.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. Yet, if the green ✓ appears on the last commit in your GitHub repository, then you satisfied all of the main checks. 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.

Since this is another challenging practical assignment and you are still learning how to use the Java classes and objects, 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. 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/practicaleight/PasswordCheckerMain.java` that meets all of the established source code requirements and produces the desired text-based output.

2. A correct version of `src/main/java/practicaleight/PasswordChecker.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 it correctly, as evidenced by a red ✗ in your commit listing, 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.