Objectives
To continue practicing the use of GitHub to access the files for a practical assignment. You will also continue to practice using Slack to support communication with the technical leaders and the course instructor. Next, you will learn more about conditional and repetition statements and the use of if/else statements, including if/else statements, and while loops. Finally, you will further explore how the course’s automated grading tool assesses your progress towards correctly completing the project.

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 creating and using Java objects, you should review Chapters 5.1 through 5.3 in the textbook.

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. Please follow the steps from the previous assignments to accept the assignment, create an assignment repository, and to download it locally.

In this practical you are invited to develop a program that will validate user’s password given certain requirements. Specifically, this program should satisfy the following requirements:

- Ask the user to input the password.
- Using a sequence of if/else statements, check the password requirements stated below.
- If the requirements are not satisfied, the program should re-prompt the user for a different password and re-validate the password. This should be done until the user enters a valid password.

A template program with the initial Java statements to get you started is given in the lab repository. Study the given program, paying close attention to the while loop used in the template program. Then compile and run it before you proceed!

Password requirements:
1. A password can not contain any spaces. You can use contains() method of the String class to check if the user’s password contains a space (“ ”). If it does not an appropriate boolean variable should be assigned a value of true. Assuming password is the String name for the user’s password, you can accomplish this by using the following lines of code:
if (!password.contains(" ")) {
    isvalidNoSpace = true;
    System.out.println("Pass: Your password does not contain a space!");
} else {
    System.out.println("Fail: Your password contains a space");
}

2. **A password must have at least eight characters.** You can use length() method of the String class to check if the number of characters in the user's password is greater or equal to the PASSWORD_MINIMUM_LENGTH variable, which is declared and assigned a value of 8 on line 14.

3. **A password must contain at least one numeric digit.** You can use contains() method of the String class and || boolean operator to check if the user's password contains each of the ten possible digits.

4. **A password must have at least one uppercase letter.** You can use equals and toLowerCase methods of the String class and the boolean operator ! to check if the password String given by the user is equal to the same string converted to all lowercase letters. If it is not equal, that means the original String was already in all lowercase letters and did not contain any uppercase letters. Assuming password is the String name for the user's password, you can accomplish this check by:
   ```java
   if(!password.equals(password.toLowerCase())) {
   ...
   }
   ```

5. If you have extra time think of additional checks to perform on the password and implement them!

Below are two sample outputs from running the program.

```
jjumadinova@aldenv113:~/practical06$ gradle -q --console plain run

Please enter a password.
OnetwoThree4
Pass: Your password does not contain a space!
Pass: Your password meets the length requirement!
Pass: Your password contains a digit!
Pass: Your password contains an upper case letter!
Your password satisfies all requirements.

jjumadinova@aldenv113:~/practical06$ gradle -q --console plain run
Janyl Jumadinova Thu Oct 10 22:32:00 EDT 2019

Please enter a password.
onetwo
Pass: Your password does not contain a space!
Fail: Your password does not meet the length requirement!
Fail: Your password does not contain a digit!
```
Fail: Your password does not contain an upper case letter!
Your password does not satisfy all requirements.
Please enter another password
one two three
Fail: Your password contains a space
Pass: Your password meets the length requirement!
Fail: Your password does not contain a digit!
Fail: Your password does not contain an upper case letter!
Your password does not satisfy all requirements.
Please enter another password
Onetwothree4
Pass: Your password does not contain a space!
Pass: Your password meets the length requirement!
Pass: Your password contains a digit!
Pass: Your password contains an upper case letter!
Your password satisfies all requirements.

In addition to regularly building (compiling) your program, you should thoroughly test it with various user password inputs as well!

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 RollDice 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/practicalsix/PasswordChecker.java` file has been changed and is ready for transfer to GitHub you would first type “`git commit src /main/java/practicalsix/PasswordChecker.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 technical leader 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/practicalsix/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 commit and push your program, 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.

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 technical leader 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.