Objectives
In addition to enhancing the skills that you have learned in the past laboratory assignments, the purpose of this assignment is to explore the ideas of a “class” and an “object” in the Java programming language. Specifically, you will learn how to use the methods provided by the java.lang.String class to inspect and manipulate a String object. You will then apply your knowledge of Strings to the application domain of steganography, or the practice of “hiding” messages inside of non-secret content. Finally, you will read two articles about computational thinking and present your reflection on the topics raised in these articles.

Reading Assignment
To review what you have already learned about about variables, expressions, and user input, please read Sections 2.1–2.6 in your textbook; pay close attention to the Scanner methods in Figure 2.7 and the program in Listing 2.8. To learn more about Java classes and objects and, in particular, the methods provided by the java.lang.String class, please study Sections 3.1–3.2 in the textbook.

You are also required to read two articles, including Learning to Think Like a Computer by New York Times and Computational Thinking, 10 years later blog by Dr. Wing. You might find it useful to also read Dr. Wing’s original article referenced in the blog.

You should also study all of the slides that we discussed during class sessions. Finally, don’t forget to examine the “GitHub Guides” if you have questions about how to create and use a Git repository. Please see the instructor if you have any questions about these readings.

Accessing the Laboratory Assignment on GitHub
Since this is your first team-based assignment we will be using a group assignment functionality of GitHub Classroom. For group assignments only one person will be creating the team while the other team members will join that team. Please form a team consisting of two or three members, assign one person to be the designated team manager.

The selected team manager 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 laboratory assignment and create a new team with a unique and descriptive team name (under “Or Create a new team”).

Now the other members of the team can click on the assignment link in the #labs channel and select their team from the list under “Join an Existing Team”. When other team members join their group in GitHub Classroom, a team is created in our GitHub organization. Teams have pretty cool functionality, including threaded comments and emoji support. Every team member will be able to push and pull to their team’s repository. Your team’s project manager should be the one to resolve any conflicts or merge pull requests.

Unless you provide the instructor with documentation of the extenuating circumstances that you are facing, not working in a team and not accepting the assignment means that you automatically receive a failing grade for it.
Implementing and Evaluating a Steganography Program

Within the field of computer security, there are many sub-fields that develop strategies for sending, receiving, and storing secret messages. Involving the hiding of a secret “in plain sight”, steganography is one way in which you can create and send a secret message. For this laboratory assignment, you will implement a program, called `MessageHide.java`, that will perform these operations:

1. Read in the user’s message from the terminal. You can assume that it is exactly 20 characters long. For now, `MessageHide` should allow the user to enter any type of “padding letter” before or after a message that is less than twenty characters. Your program should not work for the messages that are longer than 20 characters.

2. Since the `MessageHide` program must output all of its characters in a capitalized form, you should transform the message provided by the user so that it only contains upper-case letters.

3. Finally, `MessageHide` should output a 20 × 20 “grid” of letters that contain the user’s message “hidden” inside of it. All of the letters in this grid should be capitalized. Your team should brainstorm and prototype different techniques for effectively hiding the user’s message in the grid of letters. As you implement your program, you must make decisions about the following matters: (i) what letters will you add to the grid to best hide the user’s message? (ii) where and how will you place the user’s message in the grid? (iii) what features of the Java programming language will you use to ensure the grid is formatted properly in the terminal window? As you answer these questions and finish the implementation and testing of `MessageHide.java`, it may help to consider the fact that the user’s message might be better hidden if the grid contains both some randomly chosen letters and some letters found in the user’s input. You should devote a considerable amount of time to brainstorming, prototyping, and evaluating (e.g., by asking a friend to try your program) different ways in which you can effectively hide messages.

In summary, this assignment asks you to write a program, `src/main/java/labfour/MessageHide.java`, that will complete a message-hiding task. In particular, it must meet the following requirements:

1. Contain at least eight single-line comments and two multi-line comments.
2. Include calls to at least 12 “println” statements.
3. Include at least one call to the “toUpperCase”, “substring” and “charAt” methods.
4. Declare at least 5 “String” variables for the lines of output.
5. Produce a total of 24 lines of output.
6. Display the contiguous Message hidden in a 20 × 20 grid.
7. Separate the three blocks of output with a blank horizontal line.

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 atom 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 `MessageHide` correctly, then please talk with a teaching assistant 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 the graphical output. Make
sure there are spaces separating Messages in your output (did you forget to put a space inside the
quotation marks after your last name?). 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. Further details about
the writing aspect of the assignment is given in the section titled “Reflection on Computational
Thinking and MessageHide.”

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:

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 MessageHide 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**

As verified by Checkstyle, the code for the src/main/java/labfour/MessageHide.java 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 these well-established standards for correctness. Please see the instructor if you have questions about GatorGrader.

To get 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. 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/labfour/MessageHide.java file has been changed and is ready for transfer to GitHub you would first type “git commit src/main/java/labfour/MessageHide.java -m “Your descriptive commit message” in your terminal, followed by typing “git push”, and then checking to see that the transfer to GitHub is successful. Remember, to correctly complete this assignment you need to commit all code and writing files to GitHub. If you notice that the network communication with GitHub did not work, then please try to determine why, asking a teaching assistant or the course instructor for additional assistance.

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 main 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. In conclusion, here are some points to remember for creating programs that performs steganography:

1. You should think carefully about how the 20 × 20 grid can be displayed using variables.
2. As in past assignments, your program only needs to have one main method in one file.
3. See Figure 3.1 for a listing of some common “String” methods for use in your program.
4. Your program will alternate between creating and displaying textual output—this is okay!
5. Don’t forget to review the assignment sheets from the previous laboratory and practical assignments as they contain insights that will support your completion of this assignment.

**Reflection on Computational Thinking and MessageHide**

As you develop algorithmic solutions to the message hiding problem, your team is also invited to read two articles: 1) *Learning to Think Like a Computer* by New York Times and 2) *Computational...*
Thinking, 10 years later blog by Dr. Wing. You might find it useful to also read Dr. Wing’s original article referenced in the blog. These articles should provide you with the insight into the important new skills you are developing in this course. Then, your team should discuss these articles and develop persuasive points for the questions below:

1. What are the two-three new ideas you have learned from the articles? It should be something you did not know or did not think of before.

2. How did you utilize computational thinking during this or previous lab assignments?

3. In your opinion, what is the future of computational thinking and its use in society (education, research, every day life, etc.)? Please brainstorm both positive and negative aspects.

During your next lab session on October 3, your team will present answers to these questions in an informal format to the technical leaders and the instructor. There is no need to make slides.

Finally, you are invited to write a comprehensive reflection document stored as a Markdown file. The first paragraph of your document should describe the specific commands you used in your program and any technical challenges that you encountered when using them. For every challenge that you encountered, please explain your solution for it. The second paragraph should describe your team’s strategy for hiding a message in the 20 x 20 character grid and include a details surrounding your team work (e.g., did you work on all parts of the lab together, did you distribute the work, who did what, did you meet outside of the lab time, etc.). Finally, the third paragraph should provide the answers to the questions raised in bullet points above.

Summary of the Required Deliverables
This assignment invites you to submit, using GitHub, the following deliverables.

1. An informal presentation to the technical leaders and the instructor of your reflection on the assigned articles. You should prepare to talk for approximately five minutes and provide feedback based on the questions outlined in the previous section.

2. Stored in writing/reflection.md, a three-paragraph Markdown-based reflection document fulfilling the requirements outlined in the previous section.

3. A complete and correct version of src/main/java/labfour/MessageHide.java that both meets all of the established requirements and produces the desired textual output in the terminal.

Evaluation of Your Laboratory Assignment
The grade that a student receives through Sakai on this assignment will have the following components:

- **Percentage of Correct GatorGrader Checks [up to 75%]:** Students are encouraged to repeatedly try to implement a Java program that passes all of GatorGrader’s checks by, for instance, creating a program that produces the correct output. Students should also repeatedly revise their technical writing to ensure that it also passes all of GatorGrader’s checks about, for instance, the length of its content and its appropriate use of Markdown.

- **Travis CI Build Status [5%]:** Since additional checks on the source code and/or technical writing may be encoded in Travis CI’s actions and, moreover, all of the GatorGrader checks are also run in Travis CI, a portion of the students’ lab grade depends on whether their last before-the-deadline build passes and a green checkmark; appears in their GitHub commit log
instead of a red mark. As with the previous grading component, students are encouraged to repeatedly revise their source code and technical writing in an attempt to get their Travis CI build to pass.

- **Mastery of Technical Writing [up to 10%]:** Students will also a portion of the lab grade when the responses to the technical writing questions presented in the ‘writing/reflection.md’ reveal a mastery of both writing skills and technical knowledge. To receive this portion of the grade, the submitted writing should have correct spelling, grammar, and punctuation in addition to following the rules of Markdown and providing technically accurate answers. Students are encouraged to ask the course instructor or a student technical leader to use the GitHub issue tracker to provide feedback on their mastery of technical writing skills.

- **Mastery of Technical Knowledge and Skills [up to 10%]:** Students will receive a portion of their assignment grade when their GitHub repository reveals that they have mastered all of the technical knowledge and skills developed during the completion of this project. As a part of this grade, the instructor will assess aspects of the project including, but not limited to, the use of effective source code comments and Git commit messages. Students are encouraged to ask the course instructor or a student technical leader to use the GitHub issue tracker to provide feedback on how well their work demonstrates mastery of the assignment’s technical knowledge and skills.