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 use development tools to complete a real-world programming task, using sites like GitHub to support your completion of the laboratory assignment.

Reading Assignment

To review what you have already learned 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 `WordHide.java`, that will perform these operations:

1. Read in the user’s word from the text file (“wordhide_inputs”) that is exactly ten characters in length. For now, `WordHide` should allow the user to enter any type of “padding character” before or after a word that is less than ten characters. If the word is longer than ten characters, then your program should simply discard all of the characters after the tenth one (you may also implement better strategies).

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

3. Finally, `WordHide` should output a $20 \times 20$ “grid” of letters that contain the user’s word “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 word 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 word? (ii) where and how will you place the user’s word 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 `WordHide.java`, it may help to consider the fact that the user’s word 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 words.

In summary, this assignment asks you to write a program, `src/main/java/labfour/WordHide.java`, that will complete a word-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 10 “String” variables for the lines of output.
5. Produce a total of 24 lines of output.
6. Display the contiguous word hidden in a $20 \times 20$ 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 `WordHide` 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 words in your output (did you forget to put a space inside the quotation marks after your last name?). If not, then repair the program and re-build and re-run it. 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. To complete this aspect of the assignment, you should write one high-quality paragraph that reports on your experiences with the various commands and Java code segments. You should also write an additional paragraph that describes your strategy for hiding the word in the grid.

Checking the Correctness of Your Program and Writing

As verified by Checkstyle, the code for the src/main/java/labfour/WordHide.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/WordHide.java file has been changed and is ready for transfer to GitHub you would first type “git commit src/main/java/labfour/WordHide.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 WordHide**

Your team is 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.

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.)?

During your next lab session on October 4, your team will provide an information presentation in a poster style format to the teaching assistants and the instructor.

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 confronted 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 word in the 20 × 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 teaching assistants 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/WordHide.java that both meets all of the established requirements and produces the desired textual output in the terminal.

**Evaluation of Your Laboratory Assignment**

Using a comment that the instructor shares with you through the commit log in GitHub, your team will privately receive a 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 of your Java source code, the instructor will also evaluate the accuracy of both your technical writing and the comments in your source code. If your submission receives a red ✗, the instructor will reduce your grade for the assignment while still considering the regularity with which you committed to your GitHub repository and the overall quality of your partially completed work. Please see the instructor if you have questions about the evaluation of this laboratory assignment.

Due: 4 October, 2018, 2018 at 2:30 pm