Objectives

To further improve your expertise with designing, implementing, and enhancing Java methods, including the completion of tasks such as creating and calling methods that use boolean expressions, complex conditional logic, and iteration constructs. Additionally, to practice using Java methods that read input from a text file with the `java.util.Scanner`. Finally, to fully design and implement a real-world program that provides useful features. Since you will complete this assignment with a partner, the last objective for this project is to further hone your team-work skills and better your ability to use a shared GitHub repository.

Reading Assignment

To continue to learn more about “if/else-if/else” statements and boolean expressions, please again review Sections 5.1–5.3. Since this assignment will also require you to continue to use Java classes and methods, you should once again review Sections 4.1–4.5. To best prepare for the content in this laboratory assignment, you should also study Sections 5.4–5.6, paying particularly close attention to the material about `while` loops, `break` and `continue` statements, text file input, `Iterators`, and the `java.util.ArrayList` class. Students who are not familiar with text-based todo list management tools are encouraged to review the web site http://todotxt.com/.

Reviewing the Class Programs

Your first task is to review the class programs on analyzing the crime data file (`CrimeFileReader`, `CrimeAnalyzer`, `CrimeMain`). You can find the completed versions of these programs in the in-class repository under the “instructor” directory. Before you move on to the lab programs make sure you understand how the interaction between classes and methods in this example are created. Then, write one high-quality paragraph (at least 150 words) in the reflection document explaining these interactions.

Accessing the Laboratory Assignment on GitHub

This is an individual assignment. To access the laboratory assignment, you should go into the `#labs` channel in our Slack team and find the announcement that provides a link for it. Now, select a leader of your team and this person should accept the laboratory assignment and see that GitHub Classroom created a new GitHub repository for your team to access the assignment’s starting materials and to store the completed version of your assignment. Note that the team leader will have to add other members of the team as collaborators after the repository for the lab has been created. Finally, each team should fill out the Google Form associated with this lab, which is available in the `#labs` channel. Please see the course instructor with any questions.

Before you move to the next step of this assignment, please make sure that you read all of the content on the web site for your new GitHub repository, paying close attention to the technical details about the commands that you will type and the output that your program must produce. Now you are ready to download the starting materials to your laboratory computer by cloning the repository.
Welcome to the Todo List Manager.
What operation would you like to perform?
Available options: read, priority-search, category-search, done, list, quit
read
list
0, A, Understand, Draw diagram(s) to explain classes, done? false
1, A, Understand, Use the LJV to see TodoList, done? false
2, B, Explain, Add comments to all of the Todo classes, done? false
priority-search
What is the priority?
A
0, A, Understand, Draw diagram(s) to explain classes, done? false
1, A, Understand, Use the LJV to see TodoList, done? false
category-search
What is the category?
Explain
2, B, Explain, Add comments to all of the Todo classes, done? false
done
What is the identifier of the task?
0
list
0, A, Understand, Draw diagram(s) to explain classes, done? true
1, A, Understand, Use the LJV to see TodoList, done? false
2, B, Explain, Add comments to all of the Todo classes, done? false
quit
Thank you for using the Todo List Manager.

Figure 1: Sample “TodoListManager” output featuring user input and tasks read from a file.

Implementing a List Management Tool

You and your group members should explore your repository by using a text editor to study the
source code of the TodoListManager.java, TodoListItem.java, and TodoList.java files. What
methods do these classes provide? How do they work? Can you draw a picture that depicts the
relationship between these three Java classes? Before you and your group members move onto the
next step of this assignment, make sure that you understand what parts of the system are already
implemented and what you have to extend and add in order to complete the assignment.

You will notice that this assignment organizes the methods into three separate classes, as you
have seen in past in-class exercises. In particular, the TodoListManager provides the user interface
for our program and the TodoList furnishes the methods that perform the required computations
and the TodoListItem represents an entry inside of a task list. If you want to make changes to
the way in which the program accepts input or processes variables to produce output, then you
will need to modify the TodoListManager. Otherwise, if you want to modify the way in which
the program performs a computation, or add a new computation, then you must make changes to
either the TodoList or the TodoListItem. Although more complex than this example, these three
Java classes complete their work by following a pattern similar to that which is in Figures 4.7 and
4.8. Please see the instructor if you have questions about this approach.

The current implementation of the TodoListManager reads from a file called “todo.txt”, an
example of which is included below this paragraph. An individual line in the “todo.txt” file always adheres to the format “Priority, Category, Task” where “A” is the most important priority (with “B” being the next level, and so on), “Understand” being an example of a category, and “Use the LJV to see the TodoList” is a task. Following this format for a task, you should consider adding in all of items that you and your partner must complete in order to successfully finish this assignment. That is, you can actually use your TodoListManager to assist you as you complete both this project, later practical assignments, and the final project!

A, Understand, Draw diagram(s) to explain classes
A, Understand, Use the LJV to see TodoList
B, Explain, Add comments to all of the Todo classes

As shown in Figure 1, the finished version of the TodoListManager should provide features to read the todo list, search for specific tasks according to both priority and category, mark a task as done, and list all of the current tasks. Currently, the system does not include the source code to implement the priority-search and category-search features. Yet, you can see from the provided source code that the TodoListManager can already read the todo list from the file, mark a task as done, list the existing tasks, and stop running the program. For this assignment, you must run the program by typing “gradle -q --console plain run”. Can you also build and run this program?

To complete this assignment, your team is responsible for adding all of the source code that is needed to implement the priority-search and category-search features. This means that you will first have to add code that can determine when the user has input the word “priority-search” or “category-search”—what file should contain this code? Please notice that you will need to finish implementing the methods that perform these operations! Both of these operations will involve you using a java.util.Iterator to iterate through all of the instances of the TodoItem class.

When you are performing a priority-search, you will need to collect and return all of the TodoItems that match the provided priority level. For instance, using the example todo list above on this page, a request for the “A” priority tasks would return those with id values of zero and one. Similarly, the use of the category-search operation will require you to iterate through all of the TodoItems managed by a TodoList as you find those that match the requested category. For the aforementioned list and a search for the “Explain” category, the search would find only one task. The markTaskAsDone method, as shown below, gives a concrete example of how to iterate through the todoItems and use conditional logic to check if a specific todoItem has the requested toMarkId. You can use this method as an inspiration for those methods that you must implement.

```java
public void markTaskAsDone(int toMarkId) {
    Iterator<TodoListItem> iterator = todoItems.iterator();
    while (iterator.hasNext()) {
        TodoListItem todoItem = iterator.next();
        if (todoItem.getId() == toMarkId) {
            todoItem.markDone();
        }
    }
}
```

Checking the Correctness of Your Program and Writing

As verified by Checkstyle and GatorGrader, the source code for all three of your files must adhere to all of the requirements in the Google Java Style Guide available at https://google.github.io/styleguide/javaguide.html and in the README.md. 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.

You will notice that some of the provided source code does not contain all of the required comments. Moreover, the provided source code is missing many of the lines that are needed to pass the GatorGrader checks. Make sure that you review the requirements for these Java source code files, as outlined in this section and the previous one. You can also study the source code of all three of the Java files to learn more about what you need to add to them. Don’t forget to look in your team’s GitHub repository to learn about GatorGrader’s checks!

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/labeight/TodoListManager.java file has been changed and is ready for transfer to GitHub you would first type “git commit src/main/labeeight/labnine/TodoListManager.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. Also, all of the team members should regularly practice interacting with 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 X will appear instead. The instructor will reduce a student’s grade for this assignment if the red X 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.

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

1. Stored in writing/reflection.md, a three-paragraph Markdown-based document containing the reflection. The first paragraph of the reflection should explain how the class programs (CrimeFileReader, CrimeAnalyzer, CrimeMain) that you studied at the beginning of the lab session interact with each other. The next paragraph should concentrate on the lab program and explain the input, output, and behavior of each piece of code that you typed and the challenges that you confronted when using it. For every challenge that you encountered, please explain your solution for it. In the last paragraph of the reflection you should address your understanding of the multiple class and method interactions. Do you find this content challenging, what is still not clear, do you understand how to make classes and methods connect and interact with each other?
2. A complete and correct version of `src/main/java/labeight/TodoListManager.java` that meets all of the set requirements and produces the desired textual output in the terminal.

3. A complete and correct version of `src/main/java/labeight/TodoList.java` that meets all of the set requirements and supports the desired textual output in the terminal.

4. A complete and correct version of `src/main/java/labeight/TodoListItem.java` that meets all of the set requirements and supports the desired textual output in the terminal.

Evaluation of Your Laboratory Assignment

Using a report that the instructor shares with you through the commit log in GitHub, you 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.

All team members will receive the same baseline grade for the laboratory assignment. If there are extenuating circumstances in which one or more of the team members do not effectively collaborate to complete this assignment, then the course instructor will adjust the grade of specific team members so that it is higher or lower than the baseline grade, as is fair and necessary. Please see the instructor if you do not understand how he assigns grades for collaborative assignments. Finally, in adherence to the Honor Code, students should only complete this assignment with their team members. Deliverables (e.g., Java source code or Markdown-based technical writing) that are nearly identical to the work of outsiders of your team will be taken as evidence of violating the Honor Code.

Suggestions for Success

- **Follow each step carefully.** Carefully read each sentence in the assignment sheet, making sure that you precisely follow each instruction. Take notes about each step that you attempt, recording your questions and ideas and the challenges that you faced.

- **Regularly ask and answer questions.** If you have a question about one of the steps in an assignment, then you can post it to the designated labs Slack channel. Or, you can ask a student sitting next to you or talk with a teaching assistant or the course instructor.

- **Store your files in GitHub.** As in the past laboratory assignments, you will be responsible for storing all of your files (e.g., Java source code and Markdown-based writing) in a Git repository using GitHub Classroom. Please verify that you have saved your source code in your Git repository by using “`git status`” to ensure that everything is updated.

- **Keep all of your files.** Don’t delete your programs, output files, and written reports after you submit them through GitHub; you will need them again when you study for the quizzes and examinations and work on the other laboratory, practical, and final project assignments.

- **Hone your technical writing skills.** Take extra care to ensure that your writing is interesting and both grammatically and technically correct, remembering that computer scientists must effectively communicate and collaborate with their team members and the tutors, teaching assistants, and course instructor.