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. You will learn how to use the methods provided by the java.lang.String class to inspect and manipulate a String object. Specifically, you will write a Java program that manipulates Strings of Deoxyribonucleic acid (DNA) by appropriately using methods from the java.util.String and java.util.Random classes. Additionally, you will investigate ways to use Java classes and methods to organize a complete solution to an interdisciplinary problem. Finally, you will experience software development process in a team setting.

Reading Assignment

To learn more about Java strings and random numbers, review Sections 3.1–3.5 in your textbook. You should also study all of the slides that we discussed during class. 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

This is a team-based assignment. You need to form teams of 2-3 members before beginning to work on this assignment. Since this is a team assignment, once you have formed a team, you should create a unique team name. Then, one of the team members should go into the #labs channel in our Slack team and find the announcement that provides a link for the assignment. Next, choose an option to form a new team using your unique team name and accept the laboratory assignment to 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. Once your team is successfully created, the other member(s) of the team can join your unique team after clicking on the link. Unless you provide the instructor with documentation of the extenuating circumstances that you are facing, not completing the assignment in a team and not accepting the assignment means that you automatically receive a failing grade for it.

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 should clone the repository, ensuring that you have selected “Clone with SSH” option. For instance, if the course instructor ran the “git clone” command in the terminal, it would look like:

```
git clone git@github.com:Allegheny-Computer-Science-111-S2018/computer-science-111-
spring-2018-lab-5-team1.git
```

After this command finishes, you can use “cd” to change into the newly created repository.
Creating a DNA Manipulation Program

Overview

Bioinformatics is the study of biological phenomena by the use of biology, mathematics, and computer science. One of the most important study areas in bioinformatics concerns DNA. Deoxyribonucleic acid is a molecule that encodes the genetic instructions (genes) which are used by all known living organisms and many viruses to build the proteins required to sustain existence. The genes of DNA are written in the nucleotides guanine (G), adenine (A), thymine (T), and cytosine (C), (chemical compounds) which serve as the alphabet of the genetic language. Essentially, a DNA string is a string consisting of only the letters A, C, G, and T, for instance, “CAATGTAC”. These strings encode various genetic traits such as hair color, eye color, and many others.

Each DNA string has a complement formed by replacing each code letter by its complementary code. A and T are complements; so are G and C. Thus, the complement to the string “CAATGTAC” is “GTTACAGT”. DNA sometimes undergoes a mutation. There are three types of mutation: insertion of a new letter somewhere in the string; removal of a letter from the string; and replacement of one letter by another. The following table shows the examples of the replacement of letters and the complement of the given sequence.

<table>
<thead>
<tr>
<th>Strand</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>ACGTGCTCTTGGTAC</td>
</tr>
<tr>
<td>A → T</td>
<td>TCGTGCTCTTGGTTC</td>
</tr>
<tr>
<td>T → A</td>
<td>TCGAGCACAAGGATC</td>
</tr>
<tr>
<td>C → G</td>
<td>TGGAGGGAAGGATG</td>
</tr>
<tr>
<td>G → C</td>
<td>TGCACGGGAAACCATG</td>
</tr>
</tbody>
</table>

| $S_{complementary}$ | TGCACGGGAAACCATG |

Program Requirements

You should design and implement a Java program that does the following. Note that all changes are made to the original input string — that is, they are not cumulative.

1. Read in a string of DNA from a text file named “dna_inputs.txt”. You will need to declare a variable and save the DNA string from the text file into that variable. Also, you should output a message that includes the user’s DNA string in the terminal.

2. Find the complement of the given DNA string. You will need to declare a variable that stores the complement and then display the result in the terminal.

To find the complement you can use several applications of the String class’s “replace” method. For example, if I want to replace all characters ‘A’ with characters ‘T’ in the String variable called dna, then I will say dna.replace(‘A’, ‘T’). However, how do you keep track of which ‘T’s were in the original DNA sequence and which ones are the replaced one? You will need to devise a strategy to overcome this problem!

You can check your complementary sequence from the website (using the Complement option after you enter your sequence): http://arep.med.harvard.edu/labgc/adnan/projects/Utilities/revcomp.html.

NOTE: The next three parts of your program will also use Java’s java.util.Random class. You may want to read the textbook’s description of the inputs to the methods of the Random class.
3. Perform a random mutation consisting of inserting a randomly-chosen extra letter into a random position in the user’s DNA string; it must be one of the four “allowed” letters for DNA. Store the insertion mutation into a new variable and print it, ensuring that it is appropriately labeled and identifying the position of the insertion and the letter inserted.

4. Perform a random mutation consisting of removing a letter from a randomly-chosen position in the \texttt{dna}. Once again, store the result on the deletion mutation in the new variable and display it in the terminal window, making sure that it is appropriately labeled and identifying the position of the insertion and the letter removed. You should also declare and use other variables as needed to complete this task. Also, note that you can not re-use the same random position from the previous step here, you should generate a new random number.

5. Perform a random mutation consisting of altering a single letter from a randomly-chosen position in \texttt{dna}; it must be changed to a randomly-chosen letter from the set of allowed letters for DNA. Declare a variable to store your result, print it, appropriately labeled and identifying the position of the replacement, the new letter, and the letter it replaces. Also, note that you can not re-use the same random position or the same random letter from the previous step here, you should generate a new random number and a new random letter.

6. Print an exit message.

The following gives two sample runs of a previously implemented version of the program. Your output should be different! Please see the course instructor if you do not understand this program's output.

Janyl Jumadinova Mon Feb 19 11:10:23 EDT 2018
I am going to manipulate the DNA string "actg".

Complement of ACTG is TGAC
Inserting T at position 0 gives TACTG
Deleting from position 1 gives ATG
Changing position 2 gives ACGG

Thanks for using the ManipulateDna program.

Janyl Jumadinova Mon Feb 19 11:11:34 EDT 2018
I am going to manipulate the DNA string "actg".

Enter a string containing only C, G, T, and A: actg
Complement of ACTG is TGAC
Inserting G at position 0 gives GACTG
Deleting from position 0 gives CTG
Changing position 0 gives ACTG

Thanks for using the ManipulateDna program.

In the second example, nothing was changed in the last line—the \texttt{ManipulateDNA} program randomly replaced the letter “A” with the letter “A”! This behavior is acceptable.
Testing your program

Your should build and run your program after you write every few lines of code. For example, you can build and run your program after you complete implementation of each step.

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 gvim 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 ManipulateDna 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.

Technical Writing

After you have completed the programming tasks for the assignment, please reflect on the 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 three high-quality paragraphs that contain no writing errors. The first paragraph should report on your experiences with the various commands and Java code segments, and it should also describe your strategy for manipulating the DNA String. The second paragraph should outline all of the challenges you faced with this lab and your solution for overcoming these challenges. Finally, a third paragraph should describe each team member’s contribution. Please explain what strategy you took as a team to complete this assignment (e.g., whether you used pairwise programming technique, if you divided the work in a certain way, how frequently you met outside of the lab session, etc.).

Checking the Correctness of Your Program and Writing

As verified by Checkstyle, the code for the src/main/java/labfive/ManipulateDna.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 “./gatorgrader.sh --start” in your terminal window. Once this step completes you can type “./gatorgrader.sh --check”. If your work does not meet 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. 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/labfive/ManipulateDna.java file has been changed and is ready for transfer to GitHub you would first type “git commit src/main/java/labfive/ManipulateDna.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.

Due: Section 1: 1 March, Section 2: 28 February, 2018 at 2:30 pm
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.

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 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. Unless you provide the instructor with documentation of the extenuating circumstances that you are facing, no late work will be considered towards your grade for this laboratory assignment.

Summary of the Required Deliverables

This assignment invites you to submit, using GitHub, the following deliverables. Both your reflection document and your src/main/java/labfive/ManipulateDna.java needs to contain all of your team members’ names.


2. A complete and correct version of src/main/java/labfive/ManipulateDna.java that both meets all of the established requirements and produces the desired textual output in the terminal. See Section named “Program Requirements” in this document for details of this requirement.

Evaluation of Your Laboratory Assignment

Using a comment that the instructor shares with you through the commit log in GitHub, you will privately receive a feedback on this assignment. Your grade for the assignment will be a function of the whether or not it was submitted in a timely fashion, if your program received a green ✓ indicating that it met all of the requirements, and whether your program correctly implemented the steps outlined in this 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.

Suggestions for Success

• Use the laboratory computers. The computers in this laboratory feature specialized software for completing this course’s laboratory and practical assignments. If it is necessary for you to work on a different machine, be sure to regularly transfer your work to a laboratory machine so that you can check its correctness. If you cannot use a laboratory computer and you need help with the configuration of your own laptop, then please carefully explain its setup to a teaching assistant or the course instructor when you are asking questions.
• **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. If you are stuck, then please tell a teaching assistant or instructor what assignment step you recently completed.

• **Regularly ask and answer questions.** Please log into Slack at the start of a laboratory or practical session and then join the appropriate channel. If you have a question about one of the steps in an assignment, then you can post it to the designated 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. You can see if your assignment submission meets the established correctness requirements by using the provided checking tools on your local computer and by checking the commits in GitHub.

• **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.

• **Back up your files regularly.** All of your files are regularly backed-up to the servers in the Department of Computer Science and, if you commit your files regularly, stored on GitHub. However, you may want to use a flash drive, Google Drive, or your favorite backup method to keep an extra copy of your files on reserve. In the event of any type of system failure, you are responsible for ensuring that you have access to a recent backup copy of all your files.

• **Explore teamwork and technologies.** While certain aspects of the laboratory assignments will be challenging for you, each part is designed to give you the opportunity to learn both fundamental concepts in the field of computer science and explore advanced technologies that are commonly employed at a wide variety of companies. To explore and develop new ideas, you should regularly communicate with your team and/or the teaching assistants and tutors.

• **Hone your technical writing skills.** Computer science assignments require to you write technical documentation and descriptions of your experiences when completing each task. 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.

• **Review the Honor Code on the syllabus.** While you may discuss your assignments with others, copying source code or writing is a violation of Allegheny College’s Honor Code.