Project 6: Data Science Consulting

High Level Description:

The purpose of this project is to practice using arrays and file IO to process information stored in a text file. For this project, you will be role-playing as software consultants. Your firm was just hired to assist in a project on bioinformatics; an interdisciplinary field of science that develops software tools for understanding large sets of biological data. Your first task will be to understand the requirements that the client has given you, which means doing a bit of background research into their field. Then, you'll use your software skills to analyze and dissect the client's dataset to generate a detailed report. Finally, you will have the opportunity extend your knowledge of data science in Java by completing an extension task of your choice.

Instructions:

Part 1: Problem Definition & Gathering Information

As a software consultant, a major part of your job involves bridging the gap between your client and the software. For the purposes of this project, your client is a biological researcher who has been gathering data on different DNA sequences. Your first task will be to make sure you fully understand what the client requires of you, and how you can best help them accomplish their goal. Consulting involves a LOT of learning, so it is imperative that you understand what your client is doing and what they need from you to deliver a quality product. Here is the preliminary information they have sent to you:

> Dear Tesla STEM Consulting Firm,
>
> I am seeking your help in analyzing a series of nucleotide sequences that I have collected. The data is stored in a file called DNA.txt that I have attached to this email. Please write a program that will process the file and generate a series of summary statistics. It would be even better if you could make this program flexible enough to work with other data files that I want to analyze. Here is some more information on how I want the data to be processed:

For each nucleotide sequence, I need you to count the occurrences of each of the four nucleotides, calculate the mass percentage occupied by each nucleotide type (rounded to two decimal places), report the codons present in each sequence, and predict whether the sequence encodes a protein using a series of heuristics that I will describe below. Once you have analyzed the data, I need you to generate a report that includes all the information described above in a neat format that will be printed to the console. I will submit this output to my managers. Some information in this file is highly confidential, so I would appreciate your discretion as you work through this project.

To determine if a nucleotide sequence encodes a protein, it must meet all the following constraints:

1. It must begin with a valid start codon (ATG).
2. It must end with a valid stop codon (TAA, TAG, or TGA).
3. It must contain at least 4 codons, including its start and stop codons.
4. At least 30% of the sequence's mass must be Cytosine and Guanine (C & G).

To compute the mass percentages, please use the following values as the mass of each nucleotide:

- Adenine (A): 135.128 g/mol
- Cytosine (C): 111.103 g/mol
- Guanine (G): 151.128 g/mol
- Thymine (T): 127.107 g/mol

I believe that is everything. If anything else comes up, I will let you know immediately. Please feel free to speak to my contact, Mr. Scheffel, if you have any questions or concerns. I need this information by the end of this month, so please do not delay!

Thank you for your help.

All the Best,

Dr. Rosalind Franklin

Instructions for part 1:

1. Set up your project.
    a. Open Eclipse and navigate to your Package Explorer.
    b. In your eclipse-workspace, under your APCSA_Work project, within your src folder, create a new package called: project6_DataScienceConsulting_FirstnameLastname.

   c. Within this package create two new files:
      i. A new file called "README.md".
      ii. A new Java Class called "DNA.java"
2. A README.md file is an essential component of many software projects. Part 1 of this project will be setting up our README files with all the necessary information we need.
   a. Read this article from GeeksForGeeks on what a README file is and how to write an effective one.

   https://www.geeksforgeeks.org/what-is-readme-md-file/

3. Within your README.md file, include the following:
   a. Your name, class period, and the date you created this project.
   b. The title of this project (DNA, Data Science Consulting, or any other appropriate name for this project).
   c. A description of what this program will do and how it will be used.
      i. Your client gave you 4 different tasks that your program must complete for each nucleotide sequence. Be sure to include these 4 tasks and any other relevant details that were included in the email she sent.
   d. Definitions of any words that are new, or unfamiliar to you or anyone who might be looking at your program. As a consultant, you are responsible for knowing everything about your project, both on the software side, and the client side. At the very least, you must include definitions for the following:
      i. Bioinformatics
      ii. Nucleotide
      iii. Codon
   e. Both the relative and the absolute file path of where DNA.txt must be stored in order for the program to access it.
   f. If you work together with another student, or get advice from someone outside of this class, include those people in the credits. (Remember the plagiarism rules of this class. I

take them very seriously. You may chat with your fellow programmers for advice and ideas, but you may NOT copy anyone else's work).

g. Include anything else that you think would be important for someone to know while using your program.

4. You may continue to edit your README.md file as you work through this project. However, before moving on to the next step, submit your README.md to Teams to be checked for completion, neatness, and accuracy. Your client will want to be sure that you fully understand what is required of you.

Part 2: Design and Implementation

Now that you have gathered enough information and fully defined the problem, it's time to start designing your program.

1.) Where to start:

Within DNA.java, you will start with two methods: `main`, and `startAnalysis`. Your main method should only be responsible for calling `startAnalysis` with the appropriate file name. The `startAnalysis` method should take a String parameter that is the name of the file it will be analyzing. If you would like, you may prompt a user for the name of the file in `main`, but your scanner and all input handling should take place in the `startAnalysis` method. There is a starter code file in the Teams assignment for this project if you would like to use that or see an example of how DNA.java should be set up.

2.) Design and Pseudocode:

Using the information you gathered in your README.md file, start by writing out how you will accomplish this task in pseudocode. You have a series of instructions that can easily be converted into a list of programmatic steps. This is an extremely important part of building a program, so take your time and make sure you consider all the requirements. You will submit a pseudocode.txt file to Teams as part of this project's requirements.

3.) Build Your Program:

You should now be able to start writing your code. Here are some specific requirements that you will need to adhere to in order to receive full credit on this assignment:

- You must have a minimum of 4 class constant variables to make your program more readable and modifiable. You must use the following names exactly for each of your constants. See this link for more information on class constants: https://www.geeksforgeeks.org/java-symbolic-constants/
  1.) The minimum number of codons a valid protein must have, as an integer.
    - Default: 4
    - Name: `MINIMUM_LENGTH`
  2.) The percentage of mass from C and G required for a protein to be valid, as an integer.
    - Default: 30
    - Name: `CG_PERCENTAGE`
  3.) The number of unique nucleotides.
    - Default: 4 (Representing A, C, G, and T)
    - Name: `NUM_NUCLEOTIDES`
  4.) The number of nucleotides per codon.
    - Default: 3
    - Name: `CODON_LENGTH`
- You must use the following arrays. These arrays must not be declared as class variables, they must be local to your `startAnalysis` method.
  1.) Nucleotide counts
  2.) Mass percentages
  3.) Codons
- You must use methods in this program to provide structure and avoid redundancy. You must have at least four non-trivial methods other than main and startAnalysis. You should decide for yourself which methods to write, but your methods should obey at least the following constraints:
  1.) No method should be overly long.
  2.) Each method should accomplish a single distinct task.

3.) Each significant array in your program should be filled with data in its own individual method.
4.) Looking solely at the `startAnalysis` method should present a clear idea of all the major tasks your program is doing.
5.) Your method names should be clear and descriptive.
6.) Your methods should accept arrays as parameters or return arrays as appropriate.

- You may not use any material past chapter 7.4 in our textbook. There is no need to use multidimensional arrays in this project. There are certainly other ways of accomplishing this task, but for the purpose of this class you must stick to the content presented in this course.

More Information on Inputs and Outputs:

The input file, "DNA.txt" contains data in the following format:

```
Name of the first DNA sequence being analyzed
Nucleotide sequenece
Name of the second DNA sequence being analyzed
Nucleotide sequence
```

Your program should do the following:

1.) Introduce the user to the program:
```
Welcome to the DNA Scanner!  Here are the results for DNA.txt:
```
2.) (Optional) You may use a scanner in the introduction to ask the user for the name of the file that is to be scanned.
3.) Print the output in the following format to the console:
```
Name: Name Of the First DNA Sequence
Nucleotides: The string of nucleotides in the sequence.
Nucleotide Counts: [Num A, Num C, Num G, Num T]
Mass Percentages: [% A, % C, % G, % T]
Codons: [List of all codons separated by commas]
Encodes a Protein: Yes or No
```

Here is an example of what your code should print.

Input:

```
cure for cancer protein
ATGCCACTATGGTAG
```

Output:

```
Welcome to the DNA Scanner!  Here are the results for
DNA.txt:

Name: Cure for Cancer Protein
Nucleotides: ATGCCACTATGGTAG
Nucleotide Counts: [4, 3, 4, 4]
Mass Percentages: [27.32, 16.84, 30.55, 25.29]
Codons: [ATG, CCA, CTA, TGG, TAG]
Encodes a protein: Yes
```

You will also receive a second set of data in a file called Ecoli.txt. After you ensure that your code works with DNA.txt, you must also test to make sure that your code works with the new file as well. This was one of your client's specifications. Your program must be able to handle other data sets that use the same input format.

Grading:

Your final grade on this project will be a culmination of the work you submit throughout the project's timeline. There are 40 points total to be earned by completing each part of this project. Work will lose credit it if is submitted late or is incomplete. Here are the specific requirements for each section of this project:

Part 1 – Problem Definition and Gathering Information – 6 Points

You must submit a README.md file to Teams by Wednesday, February 12th, with the following requirements:

- ❑ The file contains the creator's name, class period, and the date the project was created.
- ❑ The file contains an appropriate title for the program you submitted.

- ❑ The file contains an accurate description of what the program will accomplish.
- ❑ The file contains definitions of at least the following words (bioinformatics, nucleotide, codon). You may include more definitions as necessary.
- ❑ The file contains both the relative and absolute file paths of where DNA.txt must be saved for the program to work as intended. You may also include file paths for Ecoli.txt and any other data sets you may end up analyzing.
- ❑ The file must be neat, organized, and use standard formatting conventions (see the link or look online for information on how to format your README file).

6 Points Total – README.md Due at Midnight on February 12th, 2025.

Part 2a – Pseudocode.txt – 3 Points

You must submit a .txt file with the pseudocode design for this project by Wednesday, February 19th, 2025 with the following requirements:

- ❑ Your pseudocode must include a rough outline of how your program will accomplish this task.
- ❑ Your pseudocode should include all the different methods you might write to accomplish this task, in addition to main and startAnalysis.
- ❑ Your pseudocode should be neat, organized and easy to follow. Other programmers should be able to design and implement your code using the pseudocode you submit.

3 Points Total – PsuedoCode.txt Due at Midnight on February 19th, 2025.

Part 2b – DNA.java – 27 Points

You must submit your complete DNA.java file to Teams by Wednesday, February 26th with the following requirements:

External Correctness (10 Points):

- ❑ List of nucleotides (2 Points):
  - ❑ Attempts to print all proteins, may have small errors.

    - ❑ Perfectly prints the nucleotides with no errors.
- ❑ Mass percentages (2 Points):
    - ❑ Attempts to print the mass percentages, may have small errors in rounding, formatting, or calculations.
    - ❑ Perfectly prints the exact mass percentages for all nucleotides.
- ❑ Codons (2 Points):
    - ❑ Attempts to print the list of codons for each protein.
    - ❑ Perfectly prints all codons with no errors.
- ❑ Protein Determination (2 Points):
    - ❑ Attempts to check if sequence encodes a protein.
    - ❑ Perfectly determines whether the sequence encodes a protein.
- ❑ Constants: Program still works when C/G percentage and minimum codon constants are modified.
- ❑ Otherwise externally correct (no errors, output is properly formatted, correct files are used, etc.)

Internal Correctness (8 Points):

- ❑ All four constants properly declared and used throughout code.
- ❑ Methods (2 Points):
    - ❑ No single method is too long, main calls startAnalysis, startAnalysis is a concise summary of the code. No chaining!
    - ❑ Different methods for counting nucleotides, mass percentage and protein determination.
- ❑ Uses arrays properly for each step of computation.
- ❑ Uses for loops to process elements of array (no loop unrolling!).
- ❑ Each section of the computation is properly based on the previous part.
- ❑ Avoids redundancy (repeated sections of code are put into methods to reduce repetition).
- ❑ Comments! Class header comment, JavaDoc comments before each method, comments to explain unique or complex code.

Programming Style (9 Points):

- ❑ Uses conventional indentation and whitespace.
- ❑ No lines over 100 characters.
- ❑ All methods and variables have meaningful names.
- ❑ All methods and variables follow conventional naming standards.
- ❑ All methods are preceded by a blank line and a JavaDoc comment.

- ❑ Header comment at the top of file with name, class period, and a description of what your code will do.
- ❑ Meaningful comments for any "tricky" parts of your code. Comments should explain **why** you chose to do something in a certain way, not explain **what** your code is doing.
- ❑ Opening curly braces are either at the end of the line beginning the block or on their own new line.
- ❑ Closing curly braces must be on their own line unless followed by an else or else if statement.

27 Points total – DNA.java Due at midnight on Wednesday, February 26th, 2025.

Part 3 – Extensions – 4 Points

In order to receive full credit for this assignment, you must also do something to extend your knowledge of the topic and challenge yourself beyond what we have already done. Along with your extension, you must respond to the project reflection form that will be posted at the end of the project. Within this form, you must explain what you did for your extension and how Here is how your extension will be graded:

- ❑ Extension has been attempted but may throw errors or is otherwise incomplete.
- ❑ Extension is complete and works as intended.
- ❑ Reflection form has been submitted and accurately describes what your extension is.
- ❑ Reflection form has been submitted and accurately describes how you challenged yourself or learned something new by doing this extension.

You are free to come up with your own ideas for your extensions, if you do, please check with Mr. Scheffel to make sure it is appropriate for this class. All extensions may go beyond what we have covered in this class (you do not need to stay in chapter 7.4 of our book). Here are a few ideas for extensions to get you started (I used DeepSeek to generate some of these ideas. Feel free to do the same if you want more!):

- **Random Sequence Generator**: Create a random sequence generator that will create new DNA sequences and unique names for them. Then analyze your sequences with your DNA.java file.
- **Generate an Output File**: Instead of printing your output to the console, have your program create and write to an external file that the client can then download.
- **Visualize the Data**: Use different Java libraries to create a visual representation of the DNA sequences. Create bar charts showing the count of A, C, G, and T and a pie chart showing the mass percentages of each nucleotide.
- **DNA Comparison**: Write a method that will compare two different sequences of DNA to highlight matching codons and generate a similarity score based on the number of matching nucleotides.
- **Real World Analysis:** Find a publicly available DNA sequence and analyze it using your program. Here's a public repository of available DNA sequences: https://www.ncbi.nlm.nih.gov/genbank/.

Good luck! Take your time and plan ahead. The more time you spend designing and planning, the easier it will be to write your code. I am available for you to ask questions or get ideas, but you are in charge of your own work. You got this. Have fun!