CSE1322L - Assignment 5 (Spring 2025)

Introduction:

One of the utilities of DNA in living creatures is the storage of information. Specifically, whenever the body needs to generate more proteins of a certain type, a DNA helix associated with that protein may be broken up so its contents can be read. At its core, the DNA helix is composed of four nucleobases: Adenine, Cytosine, Guanine, and Thymine (coded A, C, G, and T, respectively). These nucleobases are usually read in groups of 3, which is called a "codon". When reading the DNA, the body will read a codon and fetch the appropriate amino acid associated with that codon, using it as a building block for assembling the desired protein.

Biologists, with the use of computers, may want to quickly determine what possible proteins can be generated by a DNA strain. The reason why different proteins can be generated from a single strain is due to the fact that the body has different ways of reading a DNA strain. While the specifics of this process are beyond the scope of this assignment, we'll write a simple program which would allow a biologist to determine one of the possible proteins, given a DNA strain. For the biologists reading this, we will be doing a Compact 3'5' Frame 1 standard translation.

Requirements

The features described below must be in your program.

- A class classed DNALib
- DNALib has 4 static methods:
    - **static boolean validator(String)**: If the argument is empty, doesn't have a length that is a multiple of 3, or has any characters besides A, C, G, or T, returns false. Otherwise, returns true. **This must be done recursively**.
        - E.g.: "GCCCAT" is valid
        - E.g.: "GCCAT" is invalid as it has a length of 5 (5 isn't a multiple of 3)
        - E.g.: "" is invalid as it is empty
        - E.g.: "GCCYAT" is invalid as it contains invalid characters.
    - **static String reverser(String)**: Returns the reverse of the argument. **This must be done recursively**.
        - E.g.: "GCCCAT" returns "TACCCG"
    - **static String inverser(String)**: Replaces all the letters in the argument as follows: all As with Ts, all Cs with Gs, all Gs with Cs, and all Ts with As. The resulting string, after the replacements are done, is then returned. **This must be done recursively**.
        - E.g.: "GCCCAT" returns "CGGGTA"
    - **static String translator(String):** Creates a new string based on the argument. This new string, which will be returned by the method, takes in 3 characters at a time from the argument and then translates them into a single character, according to the table below. **This must be done recursively**.
        - E.g.: "GCCCAT" returns "AH"

AAA = K	AAC = N	AAG = K	AAT = N
ACA = T	ACC = T	ACG = T	ACT = T
AGA = R	AGC = S	AGG = R	AGT = S
ATA = I	ATC = I	ATG = M	ATT = I
CAA = Q	CAC = H	CAG = Q	CAT = H
CCA = P	CCC = P	CCG = P	CCT = P
CGA = R	CGC = R	CGG = R	CGT = R
CTA = L	CTC = L	CTG = L	CTT = L
GAA = E	GAC = D	GAG = E	GAT = D
GCA = A	GCC = A	GCG = A	GCT = A
GGA = G	GGC = G	GGG = G	GGT = G
GTA = V	GTC = V	GTG = V	GTT = V
TAA = -	TAC = Y	TAG = -	TAT = Y
TCA = S	TCC = S	TCG = S	TCT = S
TGA = -	TGC = C	TGG = W	TGT = C
TTA = L	TTC = F	TTG = L	TTT = F

- Your driver should prompt the user for a DNA sequence and pass it to validator(). If the sequence is valid, pass the sequence to reverser(), then inverser(), then translator(), and then print out the result. If the string is invalid, prints that the string is invalid. **Allow the user to type in upper or lower case but convert the input to upper case.**

Deliverables

- Assignment5.java (driver)
- DNALib.java

Considerations

- Remember that you will get partial credit for partial work. Try to deliver as much of the assignment as you can.
- You should not need any helper methods, though you can add them. They will not count towards your grade.
- No loops are necessary to do this assignment.
- Recall that, for a method to be used recursively, it needs to have at least two cases: one base case (solving the simplest instance of the problem) and one recursive case (which simplifies the problem and calls the function again, with the simplified parameters). The recursive case must eventually converge towards the base case.
- See at the end of the assignment sheet code for translating codons into amino acids.

Sample Output (user input in red)

[DNA Reverser and Translator]
Enter a sequence: TGTTGTTCTTTTTGAACCTTATTGATGTTT
Your DNA sequence reversed and translated is:
KHQ-GSKRTT

Sample Output (user input in red)

[DNA Reverser and Translator]
Enter a sequence: TGTTATTATTTTGAACCTTATTGATGTTT
Your DNA sequence is not valid.

Sample Output (user input in red)

[DNA Reverser and Translator]
Enter a sequence: TGTTATTATTTTUGAACCTTATTAATGTTT
Your DNA sequence is not valid.

SWITCH statement to translate codons into amino acids:

```
switch(sequence){
        case "GCA":
        case "GCC":
        case "GCG":
        case "GCT":
            aminoacid = "A";
            break;
        case "TGC":
        case "TGT":
            aminoacid = "C";
            break;
        case "GAC":
        case "GAT":
            aminoacid = "D";
            break;
        case "GAA":
        case "GAG":
            aminoacid = "E";
            break;
        case "TTC":
        case "TTT":
            aminoacid = "F";
            break;
        case "GGA":
        case "GGC":
        case "GGG":
        case "GGT":
            aminoacid = "G";
            break;
        case "CAC":
        case "CAT":
            aminoacid = "H";
            break;
        case "ATA":
        case "ATC":
        case "ATT":
            aminoacid = "I";
            break;
        case "AAA":
        case "AAG":
            aminoacid = "K";
            break;
        case "CTA":
        case "CTC":
        case "CTG":
        case "CTT":
        case "TTA":
        case "TTG":
            aminoacid = "L";
            break;
        case "ATG":
            aminoacid = "M";
            break;
        case "AAT":
        case "AAC":
            aminoacid = "N";
            break;
        case "CCA":
        case "CCC":
        case "CCG":
        case "CCT":
            aminoacid = "P";
            break;
        case "CAA":
        case "CAG":
            aminoacid = "Q";
            break;
        case "AGA":
        case "AGG":
        case "CGA":
        case "CGC":
        case "CGG":
        case "CGT":
            aminoacid = "R";
            break;
        case "AGC":
        case "AGT":
        case "TCA":
        case "TCC":
        case "TCG":
        case "TCT":
            aminoacid = "S";
            break;
        case "ACA":
        case "ACC":
        case "ACG":
        case "ACT":
            aminoacid = "T";
            break;
        case "GTA":
        case "GTC":
        case "GTG":
        case "GTT":
            aminoacid = "V";
            break;
        case "TGG":
            aminoacid = "W";
            break;
        case "TAC":
        case "TAT":
            aminoacid = "Y";
            break;
        case "TAA":
        case "TAG":
        case "TGA":
            aminoacid = "-";
            break;
    }
```