A2.2 Genome

Announcements and Clarifications

None

Brief Description

Prokaryotic bacteria are cells that do not contain membrane-bound nuclei. Instead the genetic material of this type of bacteria floats freely within the cell. The DNA of prokaryotic cells is contained in a chromosome and in numerous circular rings called plasmids. Your task is to a write a Java program that can read the DNA contained on a plasmid from a file and then perform operations on it.

Your design should be object-oriented using natural objects in the problem space: BasePair, DNA, CircularDoublyLinkedList, ... You should not use any Java collection classes in your implementation (if you are unsure whether you have done this or not please ask).

Goals / Requirements

- Implement a circular doubly linked list
- Represent the DNA structure using your circular doubly linked list
- Review program implementation and design in Java
    - Using command line arguments
    - Reading from files
    - Handling IOExceptions
    - Menu based programs
    - Object oriented design

DNA Description

DNA consists of a sequence of nucleotide pairs called base pairs. There are four distinct nucleotide bases: adenine(A), cytosine(C), guanine(G), and thymine(T). Each base matches with exactly one other to form the base pairs adenine with thymine (A-T) and cytosine with guanine (C-G). Here is an example of a small fragment of DNA containing 8 base pairs:

```
ACCTGGAA
||||||||
TGGACCTT
```

However, the plasmids in prokaryotic cells are rings so the sequence loops around and repeats, there is neither a beginning nor an end on the plasmid sequences. Since the base pairs are distinct we can describe the sequence using the only the first line of nucleotides.

```
ACCTGGAA
```

In order to visualize the sequence and to fix a frame of reference; one base pair in the sequence will be known as the focus. All operations on the sequence will start from the focus. Each command may set a new focus at the end of their operation. The example of the interface do not include displaying the options for sake of brevity. User input is marked in bold. The test files we used are accessible here test.gen, test2.gen.

Program Description

1. Read in DNA sequence from file specified as a single command line argument. If the file does not exist the program should exit and print an error. If their are characters in the file that do not correspond to a nucleotide in a base pair it should print a warning and ignore that character. Do not assume there is any limit on the number of bases we list in an input file.
2. Display the current DNA sequence for the next 20 base pairs (if the total length of the sequence is shorter than 20, display the whole sequence).
3. Display menu and ask for user input.
4. Perform the appropriate task.
5. Repeat steps 2-4.

You program will have six menu options:

1. Reverse and complement the DNA sequence

This option will reverse the sequence, operations will change from operating clockwise to operating counter-clockwise and vice versa. This will affect all subsequent functions until the option is chosen again.

```
ATGC=> ACGT   (reverse)
```

This option will also flip from the outer nucleotide to the inner nucleotide and vice versa. This will also affect all subsequent functions until the option is chosen again.

```
ATGC => TACG   (complement)

ATGC => TGCA   (reverse and complement)
```

The focus remains on the same base pair.

```
wolf(45)% java Genome test.gen
Sequence:

        ATGC

Genome Options:
... menu options ...
Enter choice (1-6): 1
Sequence:

        TGCA

Genome Options:
... menu options ...
Enter choice (1-6):
```

2. Insert base pairs

Prompt the user to enter a sequence of base pairs. If the sequence is invalid for any reason prompt the user again. Insert this sequence of base pairs immediately before the focus. The new focus should be at the beginning of the inserted sequence. This function depends on both the reversal and complement functionality.

```
wolf(52)% java Genome test.gen
Sequence:

        ATGC

Genome Options:
... menu options ...
Enter choice (1-6): 2
Please enter a sequence:
4180581590
Invalid input - '4'
Please enter a sequence:
jqotqjfoifawn
Invalid input - 'j'
Please enter a sequence:
CATTAG
Sequence:

        CATTAGATGC

Genome Options:
... menu options ...
Enter choice (1-6):
```

3. Remove base pairs

Prompt the user to a enter a number of base pairs to remove. Remove that number of base pairs starting with the focus. The new focus is the base pair immediately after the last base pair removed. If the number of base pairs to be removed is greater or equal to the total number of base pairs in the plasmid, remove them all.

```
wolf(53)% java Genome test.gen
Sequence:

        ATGC

Genome Options:
... menu options ...
Enter choice (1-6): 3
Please enter the number of base pairs to delete:
-1
Please enter the number of base pairs to delete:
0
Please enter the number of base pairs to delete:
2
Sequence:

        GC

Genome Options:
... menu options ...
Enter choice (1-6): 3
Please enter the number of base pairs to delete:
100
Sequence:
```

```
Genome Options:
... menu options ...
Enter choice (1-6):
```

4. Shift

Prompt the user to enter a positive number. Shift the focus of the plasmid sequence by that number of base pairs. The direction shifted will depend whether the sequence is reversed or not.

```
wolf(54)% java Genome test.gen
Sequence:

        ATGC

Genome Options:
... menu options ...
Enter choice (1-6): 4
Please enter the number of base pairs to shift by:
2
Sequence:

        GCAT

Genome Options:
... menu options ...
Enter choice (1-6):
```

5. Search for genes

Prompts the user for two valid base pair sequences (called the start and end codons). These codons are sequences of bases that act as markers on the DNA for specific genes. Each codon should be exactly **three** bases long. Starting at the focus search for the first sequence of three base pairs that match the start codon. When start codon is located move to the base pair past the end of the start codon. Begin searching for the end codon looking at each set of three base pairs. Record all base pairs that you pass until you find the end codon. Display the base pairs occurring between the start codon and the end codon. The new focus is the base pair after the end of the end codon.

For example (the base in focus is marked by the "^" character):

```
Start sequence:    CATAGATAG
                   ^
Start Codon: CAT
End Codon: TAG
Gene sequence: AGA

After search:      CATAGATAG
                   ^
Start Codon: AGA
End Codon: TAG
Gene sequence: nothing

After search:      CATAGATAG
                   ^
Start Codon: TAG
End Codon: AGA
Gene sequence: CAT
```

```
After search:      CATAGATAG
                         ^
```

Only sets of three bases are considered during the search so there are only three sets of sequences in the above example:

```
CATAGATAG => CAT AGA TAG
^
```

If the focus is different the sequences may also be different:

```
CATAGATAG => ATA GAT AGC
 ^
```

We will not test start and end codons that do not exist in the plasmid we give you, assume it is always possible to find a gene of finite length. Also, the start and end codons will never overlap. However, the gene may contain zero base pairs.

This function depends on both the reversal and complement functionality.

```
wolf(67)% java Genome test2.gen
Sequence:

        CATAGATAG

Genome Options:
... menu options ...
Enter choice (1-6): 5
Start Codon:
Please enter a sequence:
AG
Codon must be exactly 3 bases.
Please enter a sequence:
AGA
End Codon:
Please enter a sequence:
CAT
Result of search: TAG
Sequence:

        AGATAGCAT

Genome Options:
... menu options ...
Enter choice (1-6):
```

6. Quit

Exit the program.

Commenting and Style

- Your program should be written in a style that makes it easy to read and understand.
- At the beginning of each .java file you should include a description of the class and how it interacts with the other parts of your program.
- You are **not** required to use javadoc style comments.

- If your code is doing something complex or non-standard please comment that portion heavily.

Handin

Please hand all necessary files into your handin directory in a subdirectory named **Genome**. If you program does anything strange (bugs), awesome(extra features) or has a non-intuitive interface please include a file called README.txt which explain them. If there are bugs in your program but you do not describe them in your README you will lose more credit than if you had described them.

Hints

- Develop incrementally:
    - Get your circular doubly linked list working
    - Use your circular doubly linked list with the basic classes in the program (BasePair)
    - Develop the menu options incrementally (from easiest to hardest)
- Test with small plasmid sequences first.