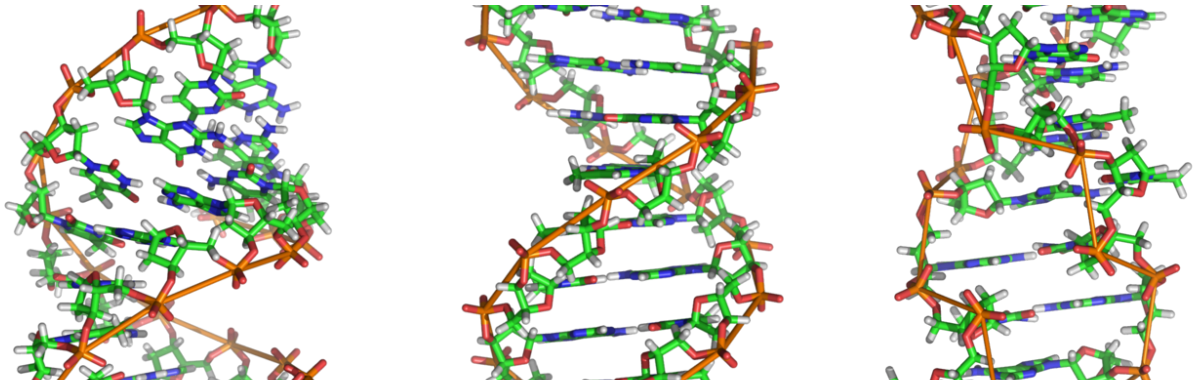
P08 DNA Transcription

Overview

Bioinformatics is the application of computer science techniques to biological problems. In the following program, you'll be using a linked queue structure to simulate the process of DNA transcription - since this isn't a biology class, I won't go into depth on anything, but if you're interested, this is an excellent primer on the topic.

We're not interested in the classical double-helix structure here; we'll just be representing what's called the primary structure of the DNA, its nucleotide sequence - the ACGT letters you may have seen before. From there, we'll translate to a complementary mRNA sequence, and then take those mRNA sequences and group them into threes to find the final amino acid sequence that will become a protein.



Psst - there's a lot more to this field than just DNA transcription, and the CS department offers classes in it. Check out **CS 576** for an introduction to bioinformatics.

Grading Rubric

5 points	**Pre-assignment Quiz**: accessible through Canvas until 11:59PM on **4/18**.
20 points	**Immediate Automated Tests**: accessible by submission to Gradescope. You will receive feedback from these tests *before* the submission deadline and may make changes to your code in order to pass these tests. Passing all immediate automated tests does **not** guarantee full credit for the assignment.
25 points	**Additional Automated Tests**: these will also run on submission to Gradescope, but you will not receive feedback from these tests until after the submission deadline.

Learning Objectives

The goals of this assignment are:

- Explore a basic linked queue implementation
- Practice iterating through a linked list without using an Iterator
- Additional practice with object-oriented programming and unit test development

Additional Assignment Requirements and Notes

Keep in mind:

- Pair programming is **NOT ALLOWED** on this assignment. Sorry, everyone.
- You are allowed to define any **local** variables you may need to implement the methods in this specification. You may NOT define any additional data fields or static fields.
- You are allowed to define any **private** helper methods you may need to implement the methods in this specification.
- The **ONLY** import statements you may include are for exceptions.
- All methods, public or private, must have their own Javadoc-style method header comments in accordance with the CS 300 Course Style Guide (lec1).
- Any source code provided in this specification may be included verbatim in your program without attribution.
- ALL test methods must be **public static**, take **zero** arguments, and return a **boolean** value. These methods MUST be contained within the DNATester class.
- Be careful when copying text directly from the specification; occasionally some characters will not be formatted correctly. If you run into compilation errors on Gradescope, read the error messages carefully.

CS 300 Assignment Requirements

You are responsible for following the requirements listed on both of these pages on all CS 300 assignments, whether you've read them recently or not. Take a moment to review them if it's been a while:

- Academic Conduct Expectations and Advice (lec1), which addresses such questions as:
  - How much can you talk to your classmates?
  - How much can you look up on the internet?
  - What do I do about hardware problems?
  - and more!
- Course Style Guide (lec1) , which addresses such questions as:
  - What should my source code look like?
  - How much should I comment?
  - and more!

Getting Started

1. Create a new project (lec1) in Eclipse, called something like **P08 DNA**.
   a. Ensure this project uses Java 11. Select "JavaSE-11" under "Use an execution environment JRE" in the New Java Project dialog box.
   b. Do **not** create a project-specific package; use the default package.
2. **Download** the following two (2) Java source files and add them to your project's src folder:
   a. QueueADT.java (the queue interface; generic)
   b. DNATester.java
3. **Create** three (3) Java source files within that project's src folder:
   a. Node.java (a **generic** class)
   b. LinkedQueue.java (a **generic** class; implements the QueueADT interface)
   c. DNA.java
4. **Add** the protected static data field from this file to your DNA class. You will use this later to translate the mRNA sequence to amino acids.

Implementation Requirements Overview

The required methods and fields for the classes in this project are listed in these Javadocs:

- Node<T>
- LinkedQueue<T>
- DNA
- DNATester

The Node is *almost* identical to P07's, except this week it is singly-linked.

Note that LinkedQueue is also generic - you will be creating both queues of Characters and Strings in the DNA class, so don't give yourself extra work.

We've provided a couple of sample tests for you in the tester class, but make sure you add more - 10 of the 20 immediate points for the autograder will focus on whether your tester class can detect problems in our broken implementations! (And whether it can tell us that our working code is correct, too.)

For the tester class, you ***must*** add AT LEAST:

- `testEnqueueDequeue()` to test adding and removing things from your queue (hint: use its toString method)
- `testQueueSize()` to test the queue's size and isEmpty methods
- `testMRNATranslate()` to test the DNA class' mRNATranslate method

DNA Method Implementation Details

The two primary methods of the DNA class are `transcribeDNA()` and `mRNATranslate()`. Let's talk a little more about how they work.

Transcribe DNA

This method takes each character from the DNA sequence and translates it into mRNA, where A becomes U, T becomes A, G becomes C, and C becomes G. It's a pretty simple one-to-one translation, EXCEPT - you cannot use linked node methods (like `getNext()`) to move through the values, since all the Nodes in our queue are private.

This means you'll need to use the `dequeue()` method to get the next character in the list - but we don't want to destroy the DNA! Make sure to re-`enqueue()` it when you're finished.

(How will you know when to stop? Well, how many nodes are in the queue?)

mRNA Translate

Here's the method where you can just destroy a queue. The queue you're given as an argument will be a queue of Characters, and you'll be producing a queue of Strings to return. You can dequeue THREE characters at a time - but be careful, not every mRNA sequence has exactly a multiple of three characters in it!

Be careful! There are TWO ways to end an mRNA translation process. Either:

1. You run out of characters to translate (the argument queue is empty), OR
2. You encounter a group of three characters that maps to the word STOP instead of a single amino acid character (your return value ends where you encountered STOP, and the argument queue may or may not still contain characters)

Translate DNA

As the Javadoc says, this method just passes the result of the first method as an argument to the second, allowing the user to get their amino acid string without having to call both methods ;)

Example Translation

A very simple example of DNA to mRNA to amino acid is a three-nucleotide sequence:

- DNA input: **GGA**
- mRNA transcription: **CCU**
- Amino acid: **P**

You'll probably want to explore something a little more interesting, though; perhaps:

- DNA input: **GATTACA**
- mRNA transcription: **CUAAUGU**
- Amino acids: **L M**

In this case, we don't have enough nucleotides for a full third amino acid, so we'll stop at two.

Here's one of my favorites:

- DNA input: **CCGGCCCTCCGGTGGATCCAA**
- mRNA transcription: **GGCCGGGAGGCCACCUAGGUU**
- Amino acids: **G R E A T** (STOP)

Note that after translation from mRNA, the mRNA queue should still have GUU left in it - the translation ends when a STOP codon is found (NOTE: the translation itself does ***not*** include STOP).

Assignment Submission

Once you're satisfied with your work, both in terms of adherence to this specification and the academic conduct (lec1) and style guide (lec1) requirements, submit your source code through Gradescope.

For full credit, please submit ONLY the following files (source code, *not* .class files):

- Node.java
- LinkedQueue.java
- DNA.java
- DNATester.java

Do NOT submit (or modify!) the interface QueueADT.java; we will provide this for you in the autograder.

Your score for this assignment will be based on the submission marked "**active**" prior to the deadline. You may select which submission to mark active at any time, but by default this will be your most recent submission.

Copyright Notice