CMPSC 111
Introduction to Computer Science I
Fall 2014

Lab 5 for Sections 03 and 04
2 October 2014
Due Thursday, 9 October by 2:30pm

Objectives

To write a Java program that manipulates strings of DNA by appropriately using methods from the `String` and `Random` classes.

Special Notes

This is your first team-based assignment. You have to work in groups of two for this assignment. You may select your own partner.

This is a longer laboratory assignment than your previous assignments. Please plan your time this week accordingly and work on it incrementally. Remember, divide and conquer!

General Guidelines for Labs

- **Work on the Alden Hall computers.** If you want to work on a different machine, be sure to transfer your programs to the Alden machines and re-run them before submitting.
- **Update your repository often!** You should add, commit, and push your updated files each time you work on them. I will not grade your programs until the due date has passed.
- **Review the Honor Code policy.** You may discuss programs with others, but programs that are nearly identical to others will be taken as evidence of violating the Honor Code.

Reading Assignment

To learn more about Java strings and random numbers, review Sections 3.1–3.5 in your textbook. Also carefully study the sample program described in the section called `A Sample Program` in this lab assignment document.

Create a new directory and a Java program

In your own `cs111F2014-<your user name>` repository inside `labs` directory, create a directory called `lab5` by typing "`mkdir lab5`". Type "`cd lab5`" to move to the new directory. Then using `gvim` create a Java program named `Lab5.java`. See the section called `DNA Manipulation` in this document for the specifics on what you need to do in your `Lab5.java` program.

Study a Sample Program

Go to the shared course repository and pull `StringDemo.java`. Copy this program into `lab5` directory inside your own `cs111F2014-<your user name>` repository. Open this program and examine it to see what it does. Then run it a few times with different input strings.

Questions to Discuss with Your Group Member (not to be handed in):

- Where is the random number generator *declared*?
- Suppose you name the random number generator `wilbur` instead of `r`. What other lines in the program need to be changed?
- Where is the scanner *declared*?
- Suppose you name the scanner `orville` instead of `scan`. What other lines in the program need to be changed?
- Suppose the line:

  ```
  s1 = s1.toUpperCase();
  ```

  is changed to:

  ```
  s1.toUpperCase();
  ```

  Will the program still correctly display the upper-case version of `s1`?
- Do the statements `s1.toUpperCase();` and `s1.toLowerCase();` change the contents of `s1`? (See previous question!)
- Suppose the user types `abcde` when asked to enter a string.
  - Write this down with numbers to indicate the positions (indices) of each character.
  - What position number is immediately to the left of the letter `a`?
  - What position number is immediately to the right of the letter `e`?
  - How many different ways are there to insert an "`x`" into the string `abcde`? List all the position numbers where this `x` could be placed. (Don't forget the beginning and end.)
  - According to the book, the `nextInt(num)` method of the `Random` class returns a random number in the range 0 to `num - 1`. List all the values that could be returned by "`r.nextInt(5)`".
  - Answer the question in the comment next to the statement "`location = r.nextInt(len+1);`".
- On paper, write down the string `ABCDEFG`, with position numbers between the letters.
  - Underline the portion corresponding to the expression:

    ```
    "ABCDEFG".substring(0,3)
    ```

– Underline the portion corresponding to the expression:

```
"ABCDEFG".substring(3)
```

– What string do we get if we evaluate the expression:

```
"ABCDEFG".substring(0,3) + "ABCDEFG".substring(3)
```

Explain, in English words, what the following statement does:

```
s2 = s1.substring(0,location) + 'x' + s1.substring(location);
```

- What is the value of the expression `"PQRST".charAt(0)` ?
- What is the value of the expression `"PQRST".charAt(2)` ?
- What is the value of the expression `"PQRST".charAt(4)` ?
- List all possible values that can be returned by the expression `r.nextInt(5)`
- Explain, in English words, what the following statement does:

```
c = "PQRST".charAt(r.nextInt(5));
```

Don't continue with the assignment until you understand the answers to all the questions above! *Please* feel free to discuss these with your classmates, teaching assistants and your instructor!

DNA Manipulation Program

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 (DNA) 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, "`CAATGTCAC`". 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 "`CAATGTCAC`" is "`GTTACAGTG`". 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 table below shows the examples of the replacement of letters and the complement of the given sequence.

Strand	**Sequence**
S	ACGTGCCTCTTGGTAC
A $\rightarrow$ T	TCGTGCCTCTTGGTTC
T $\rightarrow$ A	TCGAGCCACAAGGATC
C $\rightarrow$ G	TGGAGGGAGAAGGATG
G $\rightarrow$ C	TGCACGGAGAACCATG
$S_{complementary}$	TGCACGGAGAACCATG

Write a Java program that does the following. Note that all changes are made to the *original* input string; they are not cumulative.

1. Ask the user to type in a string of DNA, declare a variable and save the DNA string that the user inputs into a variable called "`dnaString`".

2. Print the complement of `dnaString`, appropriately labeled. (HINT: 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 `dnaString`, then I will say `dnaString.replace('A','T');`, but there's still a trick in this case of getting the complement!)

   Another HINT regarding the trick: As you make your substitutions to get your complementary string, remember that you are replacing, for example, A's to T's from the S to make our complementary sequence ($S_{complementary}$). Do not simply perform such substitution directly because you will be unable to figure out which T's were original T's (and not the replaced ones) that you were supposed to change to A's. You can check your complementary sequence from the website (using the *Complement* option after you enter your sequence): `arep.med.harvard.edu/labgc/adnan/projects/Utilities/revcomp.html`.

NOTE: The next three parts of your program will also use Java's `Random` class. You may want to read about the `Random` class at the bottom of this document to understand the concepts behind the `Random` class better.

3. Perform a random mutation consisting of inserting a randomly-chosen extra letter into `dnaString`; it must be one of the four allowed letters. Print this, 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 `dnaString`; print this, appropriately labeled and identifying the position of the insertion and the letter removed.

5. Perform a random mutation consisting of altering a single letter from a randomly-chosen position in `dnaString`; it must be changed to a randomly-chosen letter from the set of allowed letters. Print it, appropriately labeled and identifying the position of the replacement, the new letter, and the letter it replaces.

Here are two sample runs. You will get different values, of course, since the changes are random, but the structure of the output will be the same:

```
aldenv5:lab5 jjumadinova$ java Lab5
Janyl Jumadinova
Lab 5
Thu Oct 2 12:51:58 EDT 2014

Enter a string containing only C, G, T, and A: actg
Complement of ACTG is TGAC
Inserting T at position 0 gives TACTG
Deleting from position 1 gives ATG
Changing position 2 gives ACGG
```

```
aldenv5:lab5 jjumadinova$ java Lab5
Janyl Jumadinova
Lab 5
Thu Oct 2 12:52:58 EDT 2014

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
```

In the second example, nothing was changed in the last line—the program randomly replace the letter "`A`" with the letter "`A`" ! This behavior is ok.

Additional Program Requirements

- Make sure your program prints the names of all team members, the lab number, and the date as the first few output lines.
- Make sure your program contains the comment header with the honor pledge, the names of all team members, lab number, date, and the purpose of the program.
- Make sure you document your program properly, by using comments throughout your program whenever appropriate.
- Make sure your output is neat (no missing spaces, etc.) and that your program is neat (indenting, etc.).

Required Deliverables

In addition to turning in printed and signed versions, for this assignment you are invited to submit electronic versions of the following deliverables through the Bitbucket repository. As you complete this step, you should make sure that you created a `lab5/` directory within the Git repository. Then, you can save all of the required deliverables in the `lab5/` directory—please see the course instructor or a teaching assistant if you are not able to create your directory properly.

1. A completed, properly commented and formatted `Lab5.java` program.
2. An output document containing at least **three different** outputs obtained after running `Lab5` in the terminal window at least three times.
3. A document describing the strategy your team developed for completing this assignment and the work each team member has completed. While the `Lab5` program and the output file have to be the same for both members of the same team, you may include any challenges you individually faced in this document.

Share your program, the output file and the document describing your team work with me through your Git repository by correctly using "`git add`", "`git commit`", and "`git push`" commands. When you are done, please ensure that the Bitbucket Web site has a `lab5/` directory in your repository with the two files called `Lab5.java`, `output` and `report`. You should see the instructor if you have questions about assignment submission.

A Quick Tutorial on Random Number Generation

To generate random numbers, we need an object of the `Random` class. You can name this variable anything you want—"`rand`" or "`random`" are good names. To create a new random number generator named `rand`, be sure to import the `Random` class:

```
import java.util.Random;
```

and then create an instance of this class:

```
Random rand = new Random();
```

The three most useful methods in the `Random` class are `nextInt`, `nextFloat` and `nextDouble`. If `rand` is the name of our random number generator (it can be called something else) and `n` is a positive integer, `rand.nextInt(n)` produces an `int` in the range 0, . . . , `n`-1 and `rand.nextDouble()` and `rand.nextFloat()` produce a `double` and a `float` respectively in the range from 0 to 1 (not including 1).

By being clever, we can get different ranges—here are a few examples. In the Java code, assume `i` is an `int` variable, `d` is a `double` variable, and `rand` is an object of the `Random` class:

Desired Range	**Java Statement**
0, 1, 2, 3, 4, 5	`i = rand.nextInt(6);`
10, 11, . . . , 19, 20	`i = rand.nextInt(11) + 10;`
-5, -4, -3, . . . , 4, 5	`i = rand.nextInt(11) - 5;`
0, 3, 6, 9, 12	`i = 3 * rand.nextInt(5);`
-1, 1	`i = 2 * rand.nextInt(2) - 1;`
$0 \leq d < 1$	`d = rand.nextDouble();`
$0 \leq d < 10$	`d = 10 * rand.nextDouble();`
$-5 \leq d < 5$	`d = 10 * rand.nextDouble() - 5;`
0.0, 0.1, 0.2, . . . , 0.9, 1.0	`d = rand.nextInt(11)/10.0;`

You can also use random numbers to do other creative things, for example pick out characters randomly. Let's say I want to pick out a random letter out of a first name, I can first create and initialize a variable `String professorName = "Janyl";` Then I can generate a random number between 0 and 4 and save it into a variable of type `int` as `int randomNumber = rand.nextInt(5)`. Finally, I use `charAt` method to select the character at a position created by the random number generator as `professorName.charAt(randomNumber)`.