

[◀ Back to Week 2](#)[X Lessons](#)[Prev](#)[Next](#)

A PDF copy of this programming exercise can be found in the **Resources** tab.

For files related to this assignment, visit the DukeLearnToProgram Project Resources page for this course: <http://www.dukelearntoprogram.com/course2/files.php> (also linked under **Resources** here in the Coursera course).

You can also find the frequently asked questions page for this course's assignments on DukeLearnToProgram: <http://www.dukelearntoprogram.com/course2/faq.php> (also linked in the **Resources** tab).

Part 1

This assignment is to write the code from the lesson to use a `StorageResource` to store the genes you find instead of printing them out. This will help you see if you really understood how to put the code together, and might identify a part that you did not fully understand. If you get stuck, then you can go back and watch the coding videos that go with this lesson again.

Specifically, you should do the following:

1. Create a new Java project named `StringsThirdAssignments`. You can put all the classes for this programming exercise in this project.
2. Create a new Java Class named `Part1`. Copy and paste the code from your `Part1` class in your `StringsSecondAssignments` project into this class.
3. Make a copy of the `printAllGenes` method called `getAllGenes`. Instead of printing the genes found, this method should create and return a `StorageResource` containing the genes found. Remember to import the `edu.duke` libraries otherwise you will get an error message cannot find the class `StorageResource`.
4. Make sure you test your `getAllGenes` method.

Part 2

Write the method `cgRatio` that has one `String` parameter `dna`, and returns the ratio of C's and G's in `dna` as a fraction of the entire strand of DNA. For example if the `String` were `"ATGCCATAG,"` then `cgRatio` would return `4/9` or `.4444444`.

Hint: `9/2` uses integer division because you are dividing an integer by an integer and thus Java thinks you want the result to be an integer. If you want the result to be a decimal number, then make sure you convert one of the integers to a decimal number by changing it to a float. For example, (float) `9/2` is interpreted by Java as `9.0/2` and if one of the numbers is a decimal, then Java assumes you want the result to be a decimal number. Thus (float) `9/2` is `4.5`.

Write a method `countCTG` that has one `String` parameter `dna`, and returns the number of times the codon `CTG` appears in `dna`.

Part 3

Write the void method `processGenes` that has one parameter `sr`, which is a `StorageResource` of strings. This method processes **all** the strings in `sr` to find out information about them. Specifically, it should:

- print **all** the `Strings` in `sr` that are longer than 9 characters
- print the number of `Strings` in `sr` that are longer than 9 characters
- print the `Strings` in `sr` whose C-G-ratio is higher than 0.35
- print the number of strings in `sr` whose C-G-ratio is higher than 0.35
- print the length of the longest gene in `sr`

Write a method `testProcessGenes`. This method will call your `processGenes` method on different test cases. Think of five DNA strings to use as test cases. These should include: one DNA string that has some genes longer than 9 characters, one DNA string that has no genes longer than 9 characters, one DNA string that has some genes whose C-G-ratio is higher than 0.35, and one DNA string that has some genes whose C-G-ratio is lower than 0.35. Write code in `testProcessGenes` to call `processGenes` five times with `StorageResources` made from each of your five DNA string test cases.

We have some real DNA for you to test your code on. Download the file `brca1line.fa` from the DukeLearnToProgram Project Resources page. Make sure you save it in your BlueJ project so that your code can access it. You can use a `FileResource` to open the file and the `FileResource` method `asString` to convert the contents of the file to a single string so that you can use it like the other DNA strings you have been using. Here is an example:

```
1 FileResource fr = new FileResource("brca1line.fa");
2 String dna = fr.asString();
```

Modify your `processGenes` method so that it prints **all** the `Strings` that are longer than 60 characters and prints the number of `Strings` that are longer than 60 characters (you do not need to make changes to the rest of the method).

Modify the method `testProcessGenes` to call `processGenes` with a `StorageResource` of the genes found in the file `brca1line.fa`.

✓ Complete

