Overview

Artificial intelligence is an active and exciting area of computer science. Can computers be made to do tasks that seem "intelligent"? Some of the goals (and partial successes) of artificial intelligence include important areas such as medical diagnosis, drug discovery, and robotics. For this assignment, we will visit the part of artificial intelligence referred to as "natural language processing," i.e. getting a computer to work with a natural language such as English.

More specifically, you will write a program called AIAuthor that will actually write original prose in the style of a chosen human author. Can computers be creative? We'll find out!

This is a team assignment.

Details

Find a source text. Specifically, find a text file that contains some text that you want the computer to creatively imitate. Project Gutenberg is a great place to start.

The technique that you will use works conceptually as follows:

- Ask the user to specify a source text file, to specify an integer to server as a window size (which we'll denote as `w` throughout this description), and the number of characters that the program should generate.
- Read the source text into memory as a single string. Unlike previous examples of reading text (like for the spelling dictionary), we actually want to retain spaces in the original document. Therefore, you should read in an entire line at a time, instead of a word at time. Check out the `Scanner` class `nextLine` method.

  You will also want to use a `StringBuilder` instead of a `String` to hold the data as you read it in. The problem with using a `String` to hold the data is that every time you read a new line, you need to append it to the old one. Every time that you append to a `String`, Java creates a brand new one by copying the old `String` and then appending the new text. This is extemely slow. A `StringBuilder` is much faster way to manage appending multiple strings of text. When you're done appending, you can convert the `StringBuilder` back to a normal `String`. Here's an example:

  ```
  StringBuilder sbtext = new StringBuilder();
  sbtext.append("hello");
  sbtext.append(" goodbye ");
  sbtext.append("friends");
  String text = sbtext.toString(); // converts the StringBuilder to a normal strin
  ```

  Be careful when you append the lines together that you append a space appropriately at the end of each line. If you don't handle this right, the last character of each line and the first character of the next next line will always appear right next to each other.

- Once you have the source text in memory, choose a random substring of length `w` to start as the *seed*. You may find the `substring` method of `String` to be helpful here. Print this random substring to the screen.
- Look through the entire source text to find all occurrences of this seed in order to determine what character follows. For example, if `w=5`, and your seed is `memor`, you may find that the text `memor` appears multiple times throughout your document sometimes followed by an `i` (as in the word "memories") and sometimes followed by a `y` (as in the word "memory"). Determine all characters (including punctuation, etc.) that follow your seed, and choose one randomly. Make sure that you do this in such a way that if the `i` appears more often than the `y` for example, it is more likely to be chosen. Print this letter to the screen. Again, you may find the `substring` method of `String` to be useful.

- Create a new seed. This new seed should consist of all characters from your previous seed except the first, with the character you chose in the previous step appended at the end. For example, if your previous seed was `memor` and you chose an `i` as the next character, your new seed is `emori`. Repeat the above process to generate another random character.

Here's an example of some text that my program generates, using a window size of 5 on Grimm's fairy tales:

```
I will you learn home to taken his way, and around him, and left his come
intender him in they cried: 'The little child,' said the other; 'let us for the
cloak, and brough the peasant was to King.
```

Alternatively, when I tried it with a window size of 8, here's what I got one of the times that I ran it:

```
The servants running across the stables to salt and butter to eat and drink, the
head you must get up quickly, and fell fainting to the test.
```

For part 1 of this assignment, your program should at least be able to read in a source text, ask for a window size, and choose a random seed from the text (which should be printed to the screen). Of course, you can turn in more of the entire assignment (or the whole thing) for part 1 if you like.

Final notes

Make sure that you use good design. Think carefully about how to layout your program so that you use classes and objects appropriately, and so that your methods remain short and self-contained.

Once you get your program running, try to get your program to write the most funny or insightful text that you can.