Example Program Spec: altertxt

## Project Goals

In this project, you will be developing a Java application, altertxt, using an agile, test-driven development process across multiple deliverables. For this assignment you will use version 19 of the Java Development Kit. You will receive one grade for the entire project, but each deliverable must be completed by its own due date and all deliverables will contribute to the overall project grade. Note that each deliverable is equally weighed.

## Specification

altertxt is a command-line utility written in Java with the following specification:

### Summary

altertxt allows for simple text manipulation of the contents of a file.

### Syntax

altertxt [OPTIONS] FILE

### Description

The program altertxt performs basic text transformation on the lines of text from an input file. It is invoked as a command-line tool using the [syntax described above](#_tyjcwt), after compilation. The program writes the transformed text to the standard output and errors or usage messages to the standard error without modifying the input file. The FILE parameter is required and must be the last parameter as shown above. The only options allowed in the program, which are optional, delimited by the left [ and right ] brackets, may be provided in any order and are described as follows:

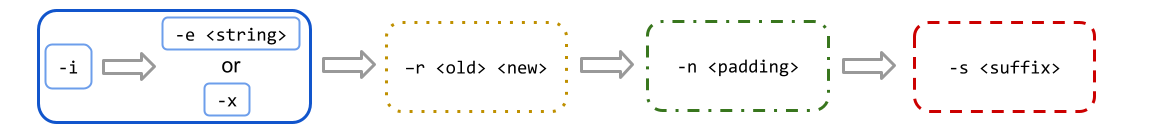
| **Option** | **Description** |
| --- | --- |
| -e <string> | **Exclude Lines**: Exclude lines containing the given substring. This option is mutually exclusive with -x. The case sensitivity for the substring search is determined by -i. |
| -r <old> <new> | **Replace String**: Replaces the first instance of substring old **in each line** with string new. The case sensitivity for the search of the old substring is determined by -i. |
| -i | **Case Insensitive**: Applies case insensitive matching when used with "Exclude Lines" and "Replace String". This option takes no parameters. |
| -x | **Remove Empty Lines**: Removes empty lines from the input file. This option is mutually exclusive with the -e option. An empty line is any line that is composed of the empty string "" immediately followed by a line separator. |
| -n <padding> | **Add Line Numbers**: Add a line number with numeric padding followed by a **single space** to the beginning of each line, where padding is an integer in the inclusive range of 1 to 9 specifying the minimum padding of the line number field. If the number of digits in the line number is less than the specified padding, zeros are left padded until the minimum padding is reached. If the number of digits in the line number is greater than the specified padding, the line number is **never truncated**. Line numbering should start at 1. |
| -s <suffix> | **Add Suffix**: Adds the string parameter suffix at the end of each line. |

#### Order of execution

The last command-line parameter provided is always treated as the filename, as shown in the [syntax section](#_tyjcwt), while OPTIONS flags can appear in any order and parsed as they appear from left to right. This means that the following two commands are equivalent when executed on the command line:

|  |  |
| --- | --- |
| Example 1 | altertxt -e test -s ..? input.txt |
| Example 2 | altertxt -s ..? -e test input.txt |

In the above examples, (Example 1) parses -e first, then -s, and finally input.txt while (Example 2) parses -s first, then -e, and finally input.txt. These two will result in the same output (assuming that the same input.txt is used for both) because the **parsing of options is independent of their execution order**. The order of execution for each option is given in the diagram below (note that the colors and border lines are there for ease of viewing):



The above diagram of the execution order of options can also be described as follows:

1. -i shall be processed first, as it determines global parameters of the computation.
2. If -e is present, and if -i **is present**, then -e shall exclude lines containing matched strings with case **insensitive** manner, and if -i is **not present**, then -e shall exclude lines containing exact matched strings with case **sensitive** manner, or if -x is present, then empty lines from the input file shall be removed.
3. If -r is present, then replacements are performed based on the option parameters, using a case **insensitive** search if -i **is present**.
4. If -n is present, then a line number is applied.
5. If -s is present, then a suffix shall be applied.

#### Notes

* To keep this application simple, all errors shall result in the display of the standard usage message. This is shown in the [syntax section](#_tyjcwt) above as well as in the files provided to you which are linked in [later sections](#_6aksms3x82t6).
* If options are repeated, only their last occurrence is applied. The [Examples section](#_2s8eyo1) below shows a case of repeated options.
* An empty input file shall produce an empty output. The [Examples section](#_2s8eyo1) below shows a case of an empty input file.
* You shall assume that the command line parameter strings will not contain newline characters (\r, \n, \r\n) as the behavior of the program would be platform dependent and may result in error during grading. Therefore there should be no test cases using these values as option parameters. Additionally, you may assume that your application will be called with valid String[] args.
* Specifying option -e with an empty string as the string parameter shall exclude all lines, resulting in an empty output.

#### Program errors

* The last line of a **non-empty** input file must be terminated by a newline. If the non-empty input file does not terminate in a newline, the program shall generate an error.
* All parameters of program options are required and shall result in an error if omitted. This means that parsing an option that should include parameters but doesn't should result in an error. The [Examples section](#_2s8eyo1) below shows a case of missing option parameters.
* Specifying option parameters with the wrong type shall result in an error.
* Specifying option -i without -e or -r shall result in an error.
* Specifying option -i with parameters shall result in an error.
* Specifying options -e and -x simultaneously shall result in an error.
* Specifying option -n with a non-integer value or an integer out of range shall result in an error.
* Specifying option -s with an empty string or option -r with an empty string as the old parameter should result in error.

### Examples of Usage

The examples described here can also be seen in JUnit 5 form on the MainTest.java file [provided to you in the below sections](#_3whwml4). In the following, "↵" represents a newline character.

|  |  |
| --- | --- |
| Example 1 | |
| altertxt sample.txt | |
| input sample.txt |  |
| stdout | **nothing sent to stdout** |
| stderr | **nothing sent to stderr** |

|  |  |
| --- | --- |
| Example 2 | |
| altertxt -i sample.txt | |
| input sample.txt | Hello, world!↵  ↵  How are you?↵ |
| stdout | **nothing sent to stdout** |
| stderr | Usage: altertxt [ -i | -e substring | -x | -r old new | -n padding | -s suffix ] FILE↵ |

|  |  |
| --- | --- |
| Example 3 | |
| altertxt -r 02 two sample.txt | |
| input sample.txt | Some words are: "one", "02", and "three"↵ |
| stdout | Some words are: "one", "two", and "three"↵ |
| stderr | **nothing sent to stderr** |

|  |  |
| --- | --- |
| Example 4 | |
| altertxt -e Bird sample.txt | |
| input sample.txt | Dog goes "woof".↵  Cat goes "meow".↵  Bird goes "tweet".↵ |
| stdout | Dog goes "woof".↵  Cat goes "meow".↵ |
| stderr | **nothing sent to stderr** |

|  |  |
| --- | --- |
| Example 5 | |
| altertxt -r blue red -e red sample.txt | |
| input sample.txt | The vibrant red roses bloomed in the garden↵  She wore a beautiful blue dress to the party↵  The sky turned into a brilliant shade of blue↵  His favorite color is red, her favorite is blue↵ |
| stdout | She wore a beautiful red dress to the party↵  The sky turned into a brilliant shade of red↵ |
| stderr | **nothing sent to stderr** |

|  |  |
| --- | --- |
| Example 6 | |
| altertxt -s ..? sample.txt | |
| input sample.txt | How are you↵  Is this a question↵ |
| stdout | How are you..?↵  Is this a question..?↵ |
| stderr | **nothing sent to stderr** |

|  |  |
| --- | --- |
| Example 7 | |
| altertxt -n 5 -n 3 -s ## -s $$ sample.txt | |
| input sample.txt | Baby Shark, doo-doo, doo-doo, doo-doo↵  Mommy Shark, doo-doo, doo-doo, doo-doo↵ |
| stdout | 001 Baby Shark, doo-doo, doo-doo, doo-doo$$↵  002 Mommy Shark, doo-doo, doo-doo, doo-doo$$↵ |
| stderr | **nothing sent to stderr** |

|  |  |
| --- | --- |
| Example 8 | |
| altertxt -s # -r cat fox -x -n 2 -i sample.txt | |
| input sample.txt | Dog goes "woof".↵  ↵  Cat goes "meow".↵  ↵  Bird goes "tweet".↵ |
| stdout | 01 Dog goes "woof".#↵  02 fox goes "meow".#↵  03 Bird goes "tweet".#↵ |
| stderr | **nothing sent to stderr** |

## 