Assignment #1

CSIS 1275 – Section 002

May 24, 2017

Create a very simple text file manager, which enables user to create, retrieve, append to, and encode a text file.

Here is how the welcome page of your program execution looks like.

```
What can I help you with? Please select an option to continue:
Press “C” to create a new text file.
Press “R” to retrieve the content of an existing text file.
Press “A” to append to an existing text file.
Press “E” to encode an existing text file.
Press “T” to terminate the program.
```

From this point, proceed to the next step based on the user’s selection. If the user enters:

“**c**” or “**C**”:

1. Ask user to enter the name of the text file he/she would like to create.
2. Check if such a file exists.
3. If the text file already exists, inform user with a proper message.
4. If there is no file with such name, do as follows:
   4.1.Create a file with the specified name.
   4.2.Ask user to enter the content he/she would like to write into the text file.
   4.3.Write content to the file.
5. In either case, prompt user with the welcome page again.

“**r**” or “**R**”:

1. Ask user to enter the name of the text file he/she would like to retrieve.
2. Check if such a file exists.
3. If the text file does not exist, inform user with a proper error message.
4. In the case of an existing text file, read all the content of the file and print them out to the output console.
6. In either case, prompt user with the welcome page again.

"**a**" or "**A**":

1. Ask user to enter the name of the text file he/she would like to append to.
2. Check if such a file exists.
3. If the text file does not exist, inform user with a proper error message.
4. If the file is a cipher (encoded) file, or has a corresponding cipher file, inform user that the file is Read-only and proceed to step 6.
5. In the case of an existing non-cipher text file, ask user to enter what he/she would like to append to the file and add the new content to the end of the file.
6. Prompt user with the welcome page again.

"**e**" or "**E**":

1. Ask user to enter the name of the text file he/she would like to encode.
2. Check if such a file exists.
3. If the text file does not exist, inform user with a proper error message.
4. If the file is a cipher file, inform user and proceed to step 6.
5. In the case of an existing non-cipher text file, do the following steps:
   5.1. Ask user to enter a specific character that he/she would like it to be replaced in the file.
   5.2. Prepend the Underscore character "_" to the name of the existing file and create a new (cipher) text file with this name.
   5.3. Read from the original file, replace all the occurrences of the input character with the Inverted Question Mark character (¿), and write the result to the cipher file.
6. Prompt user with the welcome page again.

"**t**" or "**T**":

Terminate the execution of the program.

Tips and Assumptions

1) For this program append "`throws IOException`" to the signature of the main method, as shown bellow:
```
public static void main(String[] args) throws IOException{
…
}
```
2) Assuming the Number sign "#" does not appear in any text file, use this character to indicate the end of the input content.
3) Assuming user-defined file names are not started with the Underscore character "_", this character has been used as the prefix of a cipher file name. As an example, the corresponding cipher file of the "Test.txt" text file is named as "_Test.txt".

4) A very simple Substitution Cipher is used in this program to encode content of a file, which only replaces all the occurrences of a user-selected character with the Inverted Question Mark character. For example, the content of the cipher file corresponding a text file containing "This is a simple plain text file" is: "Th¿s ¿s a s¿mple pla¿n text f¿le".
5) Java allows you to specify character literals using their hexadecimal Unicode values in the following format: '\uxxxx'. For example, the hexadecimal Unicode value of the Inverted Question Mark character (¿) is 00BF. Therefore, this character can be specified by the literal '\u00BF' in a Java program.
6) The method "`exists()`" can be called on a `File` object to check if the corresponding file exists. This method returns `true` if the file exists, and returns `false` otherwise.
7) The method "`hasNextLine()`" can be called on a `Scanner` object to check if there is another line in the input of the scanner object. This method returns `true` if another line exists, and returns `false` otherwise.