String Fun!

1 Introduction

This homework will explore working with Strings, using loops, and getting into the Java API. It will be broken up into multiple parts like last week's, so be sure to do them all and read through the whole document before turning everything in. Don't forget Checkstyle!

2 Crazy Cipher

2.1 Problem Description

In cryptography, a cipher, is an algorithm that takes in some text, performs some transformations on the letters, and outputs an encoded message. We in CS 1331 have what we like to call the Crazy Cipher to send secret messages to each other without everyone knowing. Here's how it works:

- Our cipher only works on text with 5 letters
- The first letter stays the same
- The second letter is always shifted up 4 letters (shifting as in: a = e, f = k, etc...)
- The third letter is always shifted *back* 4 letters (e = a, k = f, etc..)
- The fourth and fifth letters are swapped

For example, the word `HaPpY` would become `HeLYp`.

2.2 Solution Description

Build a program that prompts a user with three options:

- Encode a word
- Decode a word
- Quit

You must then follow the rules of the Crazy Cipher to either encode or decode the next word entered by a user (for this assignment, you can assume we will only input 5 letter, valid words). A sample output for this program is below:

```
javac CrazyCipher.java
java CrazyCipher

Welcome to the CS1331 Crrrrrrazy Cipher :D!
What would you like to do?
1 - Encode
2 - Decode
3 - Quit
1

Ok. What would you like to encode? HaPpY
Result: HeLYp

What would you like to do?
1 - Encode
2 - Decode
3 - Quit
2

Ok. What would you like to decode? HeLYp
Result: HaPpY

What would you like to do?
1 - Encode
2 - Decode
3 - Quit
3

Bye :D!
```

2.3 Tips

- You should have the String API open on your computer while you're doing this assignment. There are lots of helpful methods to use in there (substring, and charAt would be ones to look at first :))
- You can assume you will always get a 5 letter word and that each letter is different (as in the example above)
- In regards to shifting... Characters in Java are actually represented as Unicode numbers. When you have a 'char' variable, you actually have a number underneath the letter/symbol! So you can do things like:

  ```
  char mander = 'd';
  char meleon = (char)(mander + 1);
  ```

  Meleon would print to be 'e'. (If you are curious, the cast is because of a loss of precision error, can you think of why?)
- If you shift past the alphabet, you might get some sort of non-letter symbol, that's totally fine!

3 String Conversions

3.1 Problem Description

It is exceptionally useful to convert Strings from one type to another. Java can do a lot of this for you, but we want you to try writing one yourself. You will be writing a program that gets positive integers *as Strings* from the user, and converts them to variables of type `int`. This is a fairly contrived thing to do (after all, you may have already seen something like Scanner's `nextInt()` method). However, this gets you to practice many of the skills we've taught you and you'd be surprised at how often this question arises in interviews.

For example, suppose the user enters `328`. Your Scanner statement receives the String "328". Now, your goal is to somehow put the corresponding numerical value, 328, into an integer variable. **However, you must do this only using the methods listed below.** You'll likely need to process a character at a time and do some clever math.

- `Character.getNumericValue(char c)`
- `length()` in the String class
- `charAt(int i)` in the String class
- `Math.pow(double a, double b)`

You should look all of these up to figure out what they do. A simple Google search for any of them will get you on the right track. If you don't find links going to the API, you should also try to find information on them from the API (Hint: getNumericValue and pow are in the Character and Math classes respectively).

You are allowed to do any sort of arithmetic operations ($+, -, *, /$) and loops in addition to the built-in functions listed above.

3.2 Solution Description

Your program will continue to listen to a user's inputs numerical input and then simply echo the result back to them using the newly converted `int` variable. The program should quit if the user enters `-1`.

An example output of the program is below:

```
Your number: 50
Converted to: 50

Your number: 72
Converted to: 72

Your number: -1
Bye!
```

Remember, we read your code *thoroughly*. Simply echoing back the String or using any of the not allowed functions will result in *heavy* penalties.

4 Checkstyle

As always, we want you to run Checkstyle on your submission. We will still *not* be taking off for Checkstyle, but this won't last much longer. Take the time now to learn how to get rid of Checkstyle errors so when they start to count (within the next Homework or two probably) it will be easy to fix everything! Again, the full style guide for this course that you must adhere to can be found by clicking **here**.

Come to us in office hours or post on Piazza if you have specific questions about what Checkstyle is looking for and how to fix Checkstyle errors.

First, make sure you download the `checkstyle-6.0-all.jar` and `cs1331-checkstyle.xml` from the T-Square assignment page. Then, make sure you put *both* of those files in the same directory (folder) as the `.java` files you want to run Checkstyle on. Finally, to run Checkstyle, type the first line into your terminal while in the directory of your Java files and press enter.

```
$ java -jar checkstyle-6.0-all.jar -c cs1331-checkstyle.xml *.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. You can easily count Checkstyle errors by piping the output of Checkstyle through `wc -l` and subtracting 2 for the two non-error lines printed above (which is how we will deduct points). For example:

```
$ java -jar checkstyle-6.0-all.jar -c cs1331-checkstyle.xml *.java | wc -l
      2
```

Alternatively, if you are on Windows, you can use the following instead:

```
C:\> java -jar checkstyle-6.0-all.jar -c cs1331-checkstyle.xml *.java | findstr /v "Starting
    audit..." | findstr /v "Audit done" | find /c /v "hashcode()"
0
```

5 Turn-in Procedure

Submit your `CrazyCipher.java` and `StringConverion.java` files on T-Square as an attachment. Do not submit any compiled bytecode (`.class` files), the Checkstyle jar file, or the `cs1331- checkstyle.xml` file. When you're ready, double-check that you have submitted and not just saved a draft.

6 Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.

2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.

3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.

4. Recompile and test those exact files.

5. This helps guard against a few things.

   (a) It helps insure that you turn in the correct files.

   (b) It helps you realize if you omit a file or files.[1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)

   (c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is 8PM Thursday. Do not wait until the last minute!