Jamison (Jami) Biddle
JKB115
HW2 Testing Document

Method Replace first K:

This method is supposed to replace the first ‘k’ characters in a string with another given character.
Below are the following tests.

```
 HW2 t = new HW2()
// Tests first & many. Should return: "MIssIssIppi River"
> t.replaceFirstK("Mississippi River", 'i', 'I', 3)
"MIssIssIppi River"
```
(Correct)

```
//Tests for no replacements. Should return "Mississippi River"
> t.replaceFirstK("Mississippi River", 'z', 'I', 3)
"Mississippi River"
```
Correct!

```
//Tests for one replacement. Should return "MIssissippi River"
> t.replaceFirstK("Mississippi River", 'i', 'I', 1)
"MIssissippi River"
```
Correct.

```
//Tests for 0 replacements. Should return "Mississippi River"
> t.replaceFirstK("Mississippi River", 'i', 'I', 0)
"Mississippi River"
```
Correct

```
//Tests for first letter replacement. Should return "mississippi River"
> t.replaceFirstK("Mississippi River", 'M', 'm', 3)
"mississippi River"
```
Correct.

```
//Tests for last letter replacement. Should return "Mississippi RiveR"
> t.replaceFirstK("Mississippi River", 'r', 'R', 3)
"Mississippi RiveR"
```
Correct.

Method allChars()

Prints the alphabet between the two characters, including the characters inputted. Given the option for what to do it they are in the wrong order, I chose to just return the two in the proper order.

Tests:
HW2 t = new HW2()

//Tests for the full quantity of the alphabet. Should output the whole alphabet
> t.allChars('a', 'z')
"Abcdefghijklmnopqrstuvwxyz"
Correct!

//tests for just one character. B/c it's inclusive, I want it to output "aa"
> t.allChars('a', 'a')
"aa"
Correct!

//Tests few, in the middle. I'd expect "rstuv"
> t.allChars('r', 'v')
"rstuv"
Correct.

//Wrong order test. I'd expect if to return "az"
> t.allChars('z', 'a')
"az"
Correct.

//Tests few, and first. I'd expect "ab"
> t.allChars('a', 'b')
"ab"
Correct.

Method showCharOfString()

Outputs a new String, except for each char of the first string ,if it's not present in the second string, it is 'replaced' with an underscore.

Tests:
> HW2 test = new HW2()

//Tests for full string being present, with same 2 input strings. Should output "Missouri River".
> test.showCharOfString("Missouri River", "Missouri River")
"Missouri River"
//Correct!

//Tests for several, interspersed blanks, including in the first letter. Should output: "__ss__r_ R___r"
> test.showCharOfString("Missouri River", "s SR!r")
"__ss__r_ R___r"
//Correct.

//Tests for a blank 2nd string input. Should output entirely underscores.
> test.showCharOfString("Missouri River", "")
"_____"
//Correct.

//Tests a blank 1st string. Shouldn't output anything in the quotes.
> test.showCharOfString("", "s SR!r")
""
//Correct.

//Makes sure it outputs the first letter when applicable. Should return "M_ss__r_ R___r"
> test.showCharOfString("Missouri River", "s SR!rM")
"M_ss__r_ R___r"
//Correct.

Method Hangman()

Plays hangman! When a word is inputted, it prints the number of blank spaces, and the number of wrong guesses. Player guesses using a JOption Pane. When a wrong guess is inputted, the number should go up. If a correct letter is guessed, every iteration of it should be placed in the correct position. The guesses counter should not go up in the event of a correct guess.

HW2 t = new HW2()

//Guesses: s,s,s,s,s. Tests for completely incorrect guesses & repeat wrong guesses. It should output false, with no letters placed in the sequence and the wrong guesses at 4.
> t.hangman("hello", 5)
____0
1

____1
2
____2
3
____3
4
____4
5
False
//Correct!

//Guesses: h, e, l, o. Tests whether it inputs the every iteration of a letter (double l's), whether it will return the correct answer with all correct guesses. Should return true, with hell_ and 0.
> t.hangman("hello", 5)
_____ 0
h____ 0
he___ 0
hell_ 0
True
//Correct

//Guesses: o, 0, s, l, h, o. Tests a mix of correct and incorrect guesses, with the correct guesses out of order this time. Should output true, and the final wrong guesses counter should be 2 (for 0 & s).
> t.hangman("hello", 5)
_____ 0
____o 0
____o 1
____o 2
__llo 2
h_llo 2
True
//Correct!

First Method Hidden String

Searches through a 1D array of characters to check if the inputted string is present, forwards or backwards. For convenience, when the string is present, I bold the relevant chars.

//Tests for a string in the middle. Should return true.
> HW2.hiddenString(new char[]{'a','b','r',**'a','c','a','d'**,'a'}, "acad")

True
//Correct.

//Tests for a mostly, but not completely, present string. Should return false.
> HW2.hiddenString(new char[]{'a','b','r','a','c','a','d','a'}, "brad")
False
//Correct

//Tests a present, backwards string. Should return true.
> HW2.hiddenString(new char[]{'a',**'b','r','a','c'**,'a','d','a'}, "carb")
True
//Correct.

//tests a single, present char. Should return true.
> HW2.hiddenString(new char[]{'a','b',**'r'**,'a','c','a','d','a'}, "r")
True
//Correct

//Tests for a single char that is present many times. Should return true.
> HW2.hiddenString(new char[]{**'a'**,'b','r','a','c','a','d','a'}, "a")
True
//Correct

//Tests to see if a string of the whole array is present. SHould return true.
> HW2.hiddenString(new char[]{**'a','b','r','a','c','a','d','a**'}, "abracada")
true
//Correct

Second Method Hidden String

Checks in a 2D array whether a string is present with crossword rules (u, d, l, r, diagonals). The table below shows the 2D array used for testing, with a t added in only for the third from last test.

a	**b**	**c**	
r	**c**	**a**	**d**
b	**r**	**{t}**	

//Tests a single char input, with only one present. Should output true.
> t.hiddenString(new char[][]{{'a', 'b', 'c'},{'r','c','a','d'},{'b','r'}}, "d")
True
//Correct

//Tests a single char input, with many present. Should output true.
> t.hiddenString(new char[][]{{'a', 'b', 'c'},{'r','c','a','d'},{'b','r'}}, "a")
True
//Correct

//Tests reading down a column. (1st column). Should output true.
> t.hiddenString(new char[][]{{'a', 'b', 'c'},{'r','c','a','d'},{'b','r'}}, "arb")
True
//Correct.

//Tests reading up a column, using the second column. Should output true.
> t.hiddenString(new char[][]{{'a', 'b', 'c'},{'r','c','a','d'},{'b','r'}}, "rcb")
True
//correct.

//Tests reading backwards across a row, in this case the 1st row. Should output true.
> t.hiddenString(new char[][]{{'a', 'b', 'c'},{'r','c','a','d'},{'b','r'}}, "cba")
True
//Correct

//Tests reading forwards across a row, the 1st. It should output true.
> t.hiddenString(new char[][]{{'a', 'b', 'c'},{'r','c','a','d'},{'b','r'}}, "abc")
True
//Correct

//Tests reading diagonally, here to the bottom right. Also tests if it works for 2 chars. Should return true.
> t.hiddenString(new char[][]{{'a', 'b', 'c'},{'r','c','a','d'},{'b','r'}}, "cd")
True
//Correct

//Tests reading diagonally down & left. Should return true.
> t.hiddenString(new char[][]{{'a', 'b', 'c'},{'r','c','a','d'},{'b','r'}}, "ccb")
True
//Correct.

//Tests reading up & left, using a t in the bottom to make sure it can only be caught by the top left diagonal test.
> t.hiddenString(new char[][]{{'a', 'b', 'c'},{'r','c','a','d'},{'b','r','t'}}, "tca")
True
//Correct

//Tests that it outputs false with a char not present.
> t.hiddenString(new char[][]{{'a', 'b', 'c'},{'r','c','a','d'},{'b','r','t'}}, "q")
False
//Correct

//Tests that it outputs false with a string not present.
> t.hiddenString(new char[][]{{'a', 'b', 'c'},{'r','c','a','d'},{'b','r','t'}}, "btx")
False
//Correct