Cormac Dacker

Class: HW2

Method: samePrefix

Case 1: test true

Inputting matching Strings with a length shorter than either of the Strings, expecting to return “true”

```
> HW2.samePrefix("this is a test", "this is a test", 11)
true
```

Case 2: test 0

Inputted matching String with an int of 0 so that only the first char is compared, expected return is “true”

```
> HW2.samePrefix("this is a test", "this is a test", 0)
true
```

Case 3: test 1

Inputted a matching Strings of a length that is shorter than the full length of the strings, expected return is “true”

```
> HW2.samePrefix("this is a test", "this is a test", 1)
true
```

Case 4: test many

Inputting matching Strings but length far exceeds length of either string, expecting to return “false”

```
> HW2.samePrefix("this is a test", "this is a test", 100)
false
```

Case 5: test first

Inputting non-matching Strings of equal length, with the first char being mismatched, expecting to return “false”

```
> HW2.samePrefix("this is a test", "This is a test", 11)
```

```
false
```

Case 6: test last

Inputtend non-matching String with the last char being mismatched, expected return "false"

```
> HW2.samePrefix("this is a test", "this is a tesT", 14)
false
```

Case 7: test middle

Inputted non-matching String with a char in the middle being mismatched, expected to return "false"

```
> HW2.samePrefix("this is a test", "this is @ test", 14)
false
```

Method: trimSpacesFromFront

Case 1: test first

Inputting a string with 'whitespaces' in front, expected return should be the same string but with the white spaces removed

```
> HW2.trimSpacesFromFront("     Good day!")
"Good day!"
```

Case 2: test last

Inputting string without 'whitespaces' in the front but some in the back, expected return should be the inputted string

```
> HW2.trimSpacesFromFront("Good day!     ")
"Good day!     "
```

Case 3: test middle

Inputted String with whitespaces in the middle and none at either the beginning of the end of the String, expected return is no change in the inputted String.

```
> HW2.trimSpacesFromFront("Good     day!")
"Good     day!"
```

Method: countWords

Case 1: test 0

Inputted a String with no char (“”), expected return is 0

```
> HW2.countWords("      ")

1
```

Case 2: test 1

Inputted a String with one word, expected return is 1

```
> HW2.countWords("hi")

1
```

Case 3: test many

Inputting String with eleven words, expected return is 11

```
> HW2.countWords("01010010 01001001 01010000 00100000 01001000 01100001 01110010
01100001 01101101 01100010 01100101")
11
```

Method: truncate

Case 1: test 0

Inputted string of no words and with an int of 0. Expected return of the string.

```
> HW2.truncate("", 0)

""
```

Case 2: test 0.5

Inputted string of no words, but several spaces, and an int in the middle of the length of the string. Expected return of just the sting

```
> HW2.truncate("    ", 3)
"    "
```

Case 3: test 1

Inputted one word in the string with one word and no spaces, int is in half the strings length. Expected return is the string.

```
> HW2.truncate("thanks", 3)
error
```

Case 4: test 1.5

Inputted one word in the string with one word and spaces to either side, int is in half the strings length. Expected return is the string with just the spaces in front.

```
> HW2.truncate("  thanks  ", 6)
" "
```

Case 5: test many

Inputted multiple words with an int on a white space in the middle. Expected return is the string truncated at the space.

```
> HW2.truncate("I cry myself to sleep", 12)
"I cry myself"
```

Case. test many.5

Inputted multiple words with an int on aword in the middle. Expected return is the string truncated at the space.

```
> HW2.truncate("I cry myself to sleep", 9)
"I cry"
```

Case 6: test first

Inputted string with multiple words with int on the very first char. Expected return is the very first word.

```
> HW2.truncate("I cry myself to sleep", 0)
error
```

Case 7: test middle

Inputted sting with int in the middle of the length of the string. Expected return in the truncation of that sting.

```
> HW2.truncate("I cry myself to sleep", 12)
"I cry myself"
```

Case 8: test last

Inputted string with multiple words and int on the last char in the string. Expected return is the string with the last word truncated.

```
> HW2.truncate("I cry myself to sleep", 19)
"I cry myself to"
```

Method: padString

Case 1: test 0

Inputted an int of the same of the length of the string of words. Expected return is the inputted string.

```
> HW2.padString("This really is fun!", 18)
"This really is fun!"
```

Case 2: test 1

Inputted string with an int one greater that the total length of the string. Expected return is the string with one extra space in the last space slot.

Case 3: test many

Inputted string with an int far greater than the length of the string. Expected return is the string plus the amount of additional spaces.

Method: prettyPrint

Case 1: test 0

Inputted string inputted with an int of 0. Expected return is an error with a request for a larger int.

Case 2: test 1

Inputted string inputted with an int of 1. Expected return is an error with a request for a larger int.

Case 3: test many

Inputted string with an int at an appropriate point in the middle. Expected return is the string divided onto multiple line and stretched to meet those lines lengths.

Method: isAnagram

Case 1: test 0

Inputted string with no matching characters, making it not an anagram of each other. Expected return is the boolean false

Case 2: test 1

Inputted string with some matching characters but now all making them not anagrams of each other. Expected return is the boolean false.

Case 3: test many

Inputted string with all chars matching, making it an anagram of each other. Expected return is the boolean true.