1. Supercalifragilisticexpialidocious is a very normal word with many uses!

   As Mary Poppins also said, we could say it backwards
   'Dociousaliexpiesticfragicalirupes'

   ML
   You know you can say it backwards which is
   Dociousaliexpiesticfragicalirupes
   but that's going a bit too far don't you think?

   But something seems a little off with her spelling of this word.

   What should Supercalifragilisticexpialidocious backwards ACTUALLY look like?
    a. "Supercalifragilisticexpialidocious" is "suoicodilaipxecitsiligarfilacrepuS".
2. In Supercalifragilisticexpialidocious, there are a large number of vowels. What INDEX VALUES are all of the vowels at? What about the backwards word?
    a. Index values of the vowels in "Supercalifragilisticexpialidocious":
        i. a: 6, 11
        ii. e: 3, 20
        iii. i: 8, 13, 15, 18, 23, 27
        iv. o: 25, 28
        v. u: 1, 29
    b. Index values of the vowels in "suoicodilaipxecitsiligarfilacrepuS"
        i. a: 9,18, 20,22,27
        ii. E: 13, 30,
        iii. I: 3, 7, 10, 15 ,18 ,20, 25
        iv. O: 5,24
        v. u: 1, 32
3. You'll possibly notice the word 'out of bounds' during this lab and in your class sessions. Why do certain languages complain when you go 'out of bounds'?
    a. It's an index error that occurs when you are trying to access something that doesn't exist.
4. **You don't have to write down your answer for this question.**
   How do you 'build' an email address? As in, what combinations of characters do you need to recognize a string as an email address?

a. The possible characters for an email address would be a-z, A-Z, 0-9, ! #$%&'*+-/=? ^_`{|}~, . as long as it is not the first character or the character before the @.

Lab06a: Manipulating Strings
Driver: Froilan Zarate
Scribe: Alan Xiao

Scribe: Which of the above four methods should have a parameter tag? Which methods should have a return tag?

All of the following should have the return tag, but public String scrambleAdjacentTwo(int index) and public String scrambleNotAdjacentTwo(int index) should have a param tag as well.

Scribe: Did Codecheck reject your submission? Why or why not?
Scribe: If Codecheck accepted your submission, what is the score?

There are two types of scores:

n/31 which are the points you are graded on in this lab.

Checkstyle errors which will not be important right now. How many checkstyle errors did you get?

It was accepted, but we got a score of 0 with a multitude of errors; The constructor String Scrambler couldn't be applied for the given types. Check style ended with one error because of a missing Javadoc element

Scribe: What should the comment block look like?

```
/**
  * Swaps the first letter and the last letter
  *
  * @return  null if the word is 1 character, else return the edited string
  */
```

Scribe: Did the driver use a local variable for the length of 'word'?

No

Scribe: What was the new score from Codecheck? How many Checkstyle errors did you get?

- The new score from Codecheck is 24. 0 Checkstyle errors.

Scribe: What conditions would return null here?

word.length() <= x + 2 || x <= 0 || word.length() <= 1

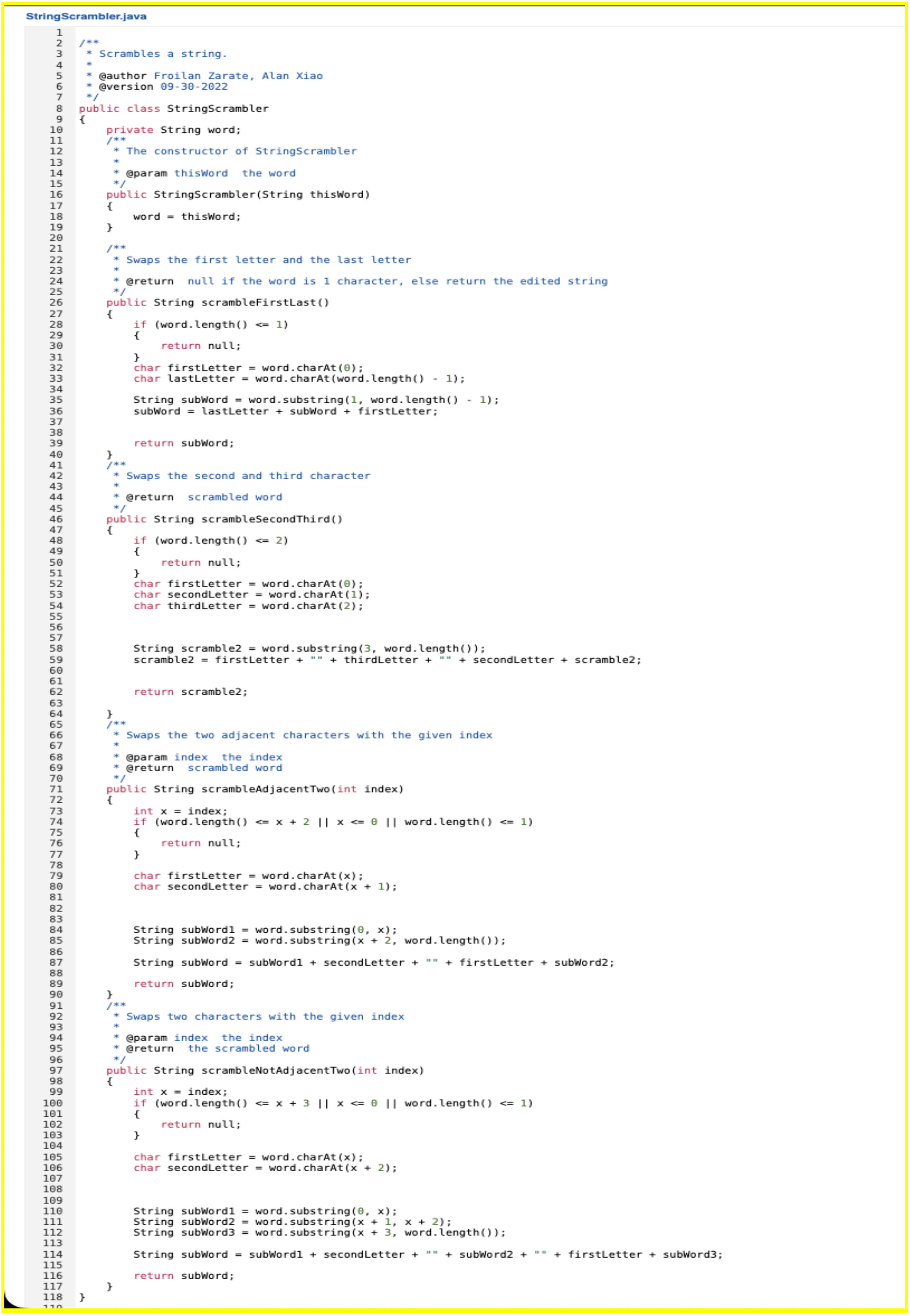
Driver: Complete method **scrambleNotAdjacentTwo()**

This method takes the input int **x**, where **x** is the INDEX of the character they want to swap.
DO NOT swap the first or the last character of the word.

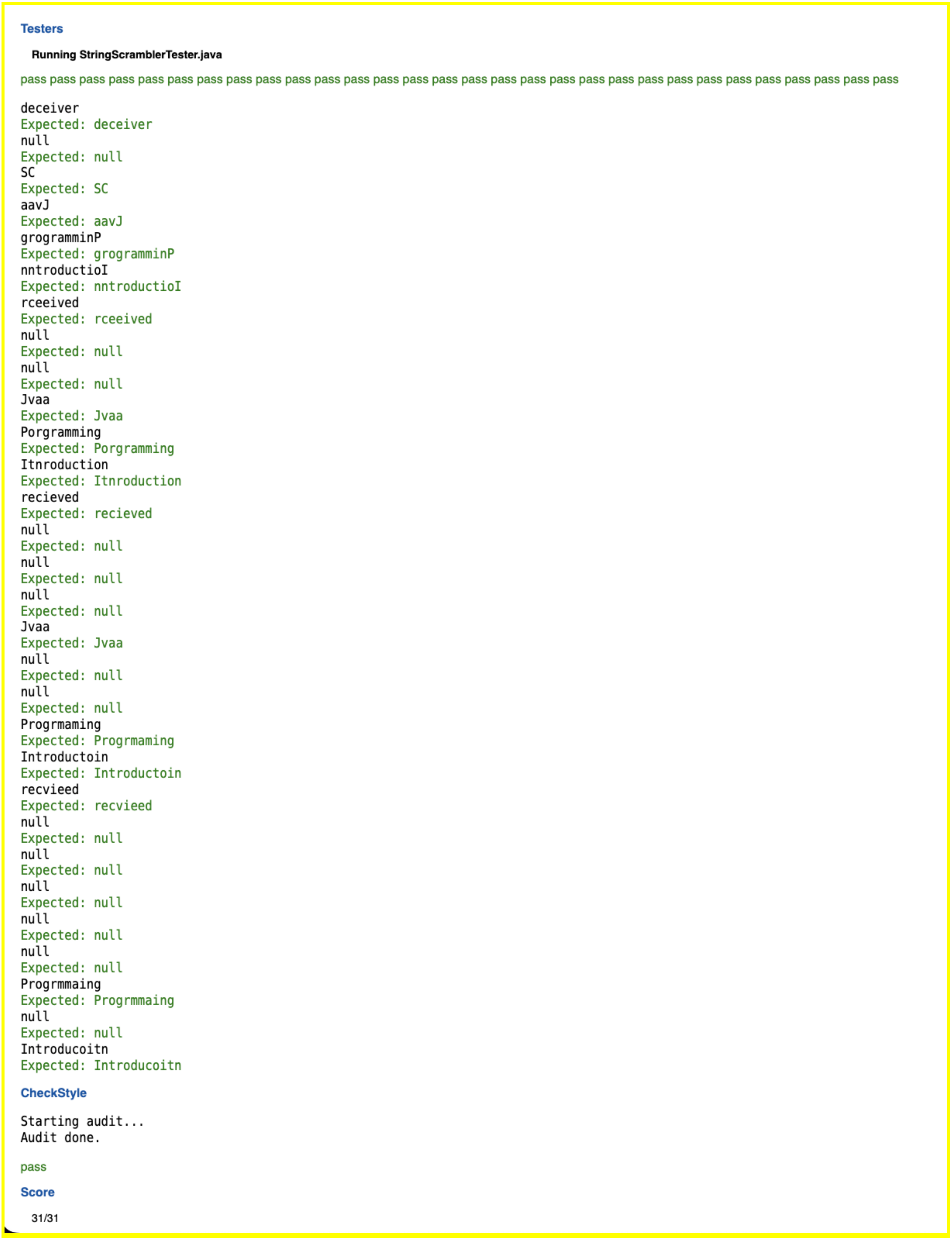
It swaps with the character at **x+2 // notice the difference**

So if the word was **'hello'** and you sent scrambleAdjacentTwo(2).
The return would be **heoll**

Scribe: What conditions would return null here:
word.length() <= x + 3 || x <= 0 || word.length() <= 1


StringScrambler.java

```java

/**
 * Scrambles a string.
 *
 * @author Froilan Zarate, Alan Xiao
 * @version 09-30-2022
 */
public class StringScrambler
{
    private String word;
    /**
     * The constructor of StringScrambler
     *
     * @param thisWord  the word
     */
    public StringScrambler(String thisWord)
    {
        word = thisWord;
    }

    /**
     * Swaps the first letter and the last letter
     *
     * @return  null if the word is 1 character, else return the edited string
     */
    public String scrambleFirstLast()
    {
        if (word.length() <= 1)
        {
            return null;
        }
        char firstLetter = word.charAt(0);
        char lastLetter = word.charAt(word.length() - 1);

        String subWord = word.substring(1, word.length() - 1);
        subWord = lastLetter + subWord + firstLetter;


        return subWord;
    }
    /**
     * Swaps the second and third character
     *
     * @return  scrambled word
     */
    public String scrambleSecondThird()
    {
        if (word.length() <= 2)
        {
            return null;
        }
        char firstLetter = word.charAt(0);
        char secondLetter = word.charAt(1);
        char thirdLetter = word.charAt(2);



        String scramble2 = word.substring(3, word.length());
        scramble2 = firstLetter + "" + thirdLetter + "" + secondLetter + scramble2;


        return scramble2;

    }
    /**
     * Swaps the two adjacent characters with the given index
     *
     * @param index  the index
     * @return  scrambled word
     */
    public String scrambleAdjacentTwo(int index)
    {
        int x = index;
        if (word.length() <= x + 2 || x <= 0 || word.length() <= 1)
        {
            return null;
        }

        char firstLetter = word.charAt(x);
        char secondLetter = word.charAt(x + 1);



        String subWord1 = word.substring(0, x);
        String subWord2 = word.substring(x + 2, word.length());

        String subWord = subWord1 + secondLetter + "" + firstLetter + subWord2;

        return subWord;
    }
    /**
     * Swaps two characters with the given index
     *
     * @param index  the index
     * @return  the scrambled word
     */
    public String scrambleNotAdjacentTwo(int index)
    {
        int x = index;
        if (word.length() <= x + 3 || x <= 0 || word.length() <= 1)
        {
            return null;
        }

        char firstLetter = word.charAt(x);
        char secondLetter = word.charAt(x + 2);



        String subWord1 = word.substring(0, x);
        String subWord2 = word.substring(x + 1, x + 2);
        String subWord3 = word.substring(x + 3, word.length());

        String subWord = subWord1 + secondLetter + "" + subWord2 + "" + firstLetter + subWord3;

        return subWord;
    }
}
```



Testers

Running StringScramblerTester.java

pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass

```
deceiver
Expected: deceiver
null
Expected: null
SC
Expected: SC
aavJ
Expected: aavJ
grogramminP
Expected: grogramminP
nntroductioI
Expected: nntroductioI
rceeived
Expected: rceeived
null
Expected: null
null
Expected: null
Jvaa
Expected: Jvaa
Porgramming
Expected: Porgramming
Itnroduction
Expected: Itnroduction
recieved
Expected: recieved
null
Expected: null
null
Expected: null
null
Expected: null
Jvaa
Expected: Jvaa
null
Expected: null
null
Expected: null
Progrmaming
Expected: Progrmaming
Introductoin
Expected: Introductoin
recvieed
Expected: recvieed
null
Expected: null
null
Expected: null
null
Expected: null
null
Expected: null
null
Expected: null
Progrmmaing
Expected: Progrmmaing
null
Expected: null
Introducoitn
Expected: Introducoitn
```

CheckStyle

```
Starting audit...
Audit done.
```

pass

Score

31/31