Joseph Quinn

New Technologists Second Round

GitHub Repository Link: https://github.com/josephjquinn/new-technologies

Question 1:

A significant challenge in the tech field that I'm eager to tackle is the issue of data privacy and protection. In today's data-driven world, the collection, storage, and analysis of personal data have become vital across various sectors, raising concerns about privacy infringement and data misuse. Many people nowadays don't realize the value in their data and should understand the importance in understanding who and what has access to it. Especially with the rise of machine learning and the massive influx of non-human data being integrated into the web as time progresses.

To address this challenge, I want to work on the development and adoption of comprehensive data privacy regulations and frameworks that prioritize individuals' rights to control their personal information.Furthermore, promoting transparency and accountability in data handling practices is essential for building trust between organizations and their users. This includes implementing clear privacy policies, obtaining explicit consent for data collection and processing, and providing users with greater control over their data.

By prioritizing data privacy and protection as core principles in tech development and operations, we can uphold individuals' privacy rights and ensure responsible and ethical use of data in the as we become more and more technology driven.

Question 2:

One notable journey of growth for me was transitioning from being a novice to proficient in working effectively within a team setting and working to open my shell. Initially, I found collaboration challenging as I grappled with communication barriers and differing work styles. However, through my recent experiences on different teams, along with seeking out opportunities for feedback, I gradually honed my teamwork skills and became better at contributing meaningfully to team objectives.

To achieve similar progress more efficiently in the future, I would adopt several strategies. Firstly, I would prioritize building strong relationships and fostering open communication within the team from the outset. Clear communication channels and regular check-ins can help align team members' expectations, mitigate conflicts, and promote a culture of trust and transparency.

I would try to actively seek out opportunities for continuous learning and skill development related to teamwork and collaboration. Participating in workshops, training sessions, or team-building exercises focused on effective communication, conflict resolution, and leadership would help me and the team become more efficient at working as a single unit.

Most importantly however, I would cultivate a growth mindset and embrace feedback as a catalyst for improvement. Soliciting input from teammates, supervisors, and peers on areas for development and actively using this feedback into my approaches would be the most beneficial for me in becoming more outspoken and team oriented.

Question 3:

My favorite app at the moment is Letterboxd. Basically, a social media platform where you can rate movies, leave reviews, and see what your friends have been watching. Something I would want to add is a recommendation page similar to Spotify where it sees what movies you have watched and enjoyed and gives you a list of recommendations. Maybe even something like a movie of the day feature where it gives you a random movie that is a little outside your current watched genre pool. This would expose users to a diverse range of films they might not have otherwise come across.

The recommendation algorithm could consider various factors, such as genres you enjoy, directors you follow, actors you like, and even specific themes or motifs present in your favorite films. It could use machine learning techniques and models to update and learn more about the user, the algorithm could continuously learn and adapt to your evolving tastes, ensuring that the recommendations remain relevant and engaging over time.

Question 4:

Being part of Vanderbilt's Change++ Developer team has been an invaluable experience that highlights my ability to thrive in a collaborative setting. This is a student-led software development organization where we create web app technology solutions to nonprofit organizations.

Our projects involve developing and deploying web and mobile applications using various developer technologies, including Typescript, React, Node.js, Express, MongoDB, and AWS EC2. Collaboration is at the core of our work, and we utilize Git for version control to ensure seamless coordination and integration of our codebase. This includes proper documentation in our commit messages, branch control for new features, and proper pull requests to maintain our apps stability.

One experience that stands out is our weekly meetings and sprint schedule. These meetings serve as a forum for discussing project progress, brainstorming ideas, and resolving any challenges or roadblocks that raised. By maintaining open lines of communication and actively seeking input from team members, we foster a collaborative environment where we can be the most productive.

We operate using a two-week sprint schedule, this ensures that we work efficiently and effectively towards our project goals and can adjust our project terms on the fly. By breaking down our tasks into manageable chunks and setting clear objectives for each sprint, we maintain momentum and make steady progress towards delivering high-quality solutions to our nonprofit partners.

Question 5:

"I am a versatile because I am always leaning and growing to find new pathways."

My passion for learning begins with an open mind and a willingness to explore diverse topics and perspectives. Whether it's picking up a new hobby or learning a new tech stack. I enjoy new opportunities to expand my horizons and grow my skillset. I try my best to always view mistakes and setbacks as valuable learning opportunities rather than obstacles to be avoided. I approach challenges with optimism, knowing that each setback is an opportunity to learn, grow, and improve. I enjoy making connections between fields and becoming more multifaceted in the tech world. I believe that by integrating insights from various sources, I can gain a deeper understanding and enrich my perspective.

Question 6:

Part a:

I do not feel as though the response from ChatGPT correctly solved the prompt given. It runs into some issues with more complicated scenarios and edge cases. Here is what I felt the response did correctly.

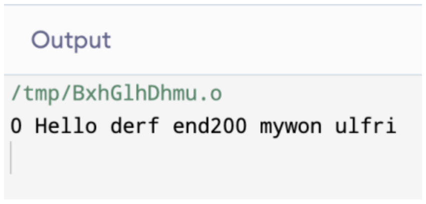
<u>Section 3: Parse:</u>

It correctly removes all whitespaces from the provided string using str.erase function, I tested this with edge cases of spaces at the start, middle, and end of the provided string, all returning the properly formatted string.

<u>Section 8: Sorting the List:</u>

The sorting section works as expected. The sorting algorithm uses a custom comparison function to sort the list of words in the `result` vector. This function iterates over pairs of characters from two strings, comparing them based on the set of criteria. Additionally, the algorithm handles the edge cases where characters are equal by advancing further into both words until a difference is found or one of the strings ends. Then the algorithm appends the compared outcome to the result vector.

Feeding the program with parameters "Hellomywonderfulfriend2000" outputs this;


Output

```
/tmp/BxhGlhDhmu.o
0 Hello derf end200 mywon ulfri
```

Which, is sorted properly according to the prompt. The numbers come first, then uppercase, then lowercase sorted alphabetically.

Section 4: Loop through the string

The program loops properly

Section 6: Out of Bounds Check

The program properly checks to see if there are enough characters left in the string to create another word of the correct size, if not it will make the remining letters into their own slightly smaller word. We see this with the previous output image, showing that the last chunk ("0") came last before it was sorted.

Section 7: Add to result.

The program properly adds each newly created word to the result vector.

Section 9: Return result.

The program properly returns the result vector.

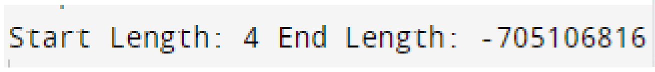
Part b:

Section 1: Declare

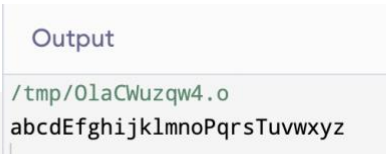
The program has a major issue where it does not load the algorithm header file whose functions are utilized in other sections i.e. the std:sort. Without this line of code, the program will not compile and throw an error. This is an easy fix by adding #include <algorithm> to line 3.

Section 2: Parse

This is one of the major issues with the program as it does not consider single digit parameters, only range. If the parameter is entered with a single digit like "4", it will output a large negative number for end_len. When the parameter is a range i.e. "4-6", the program does correctly assign start and end length. This then causes the grouping to malfunction and the program will just return the inputted string.



```
Start Length: 4 End Length: -705106816
```

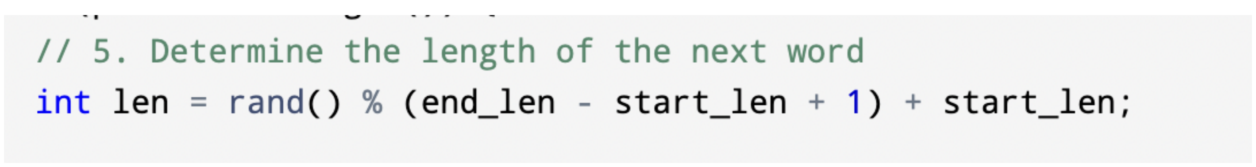


Output

```
/tmp/OlaCWuzqw4.o
abcdEfghijklmnoPqrsTuvwxyz
```

Section 5: Determine length of next word

According to the prompt, the program when entered a range should create chunks based on the alternating numbers given.

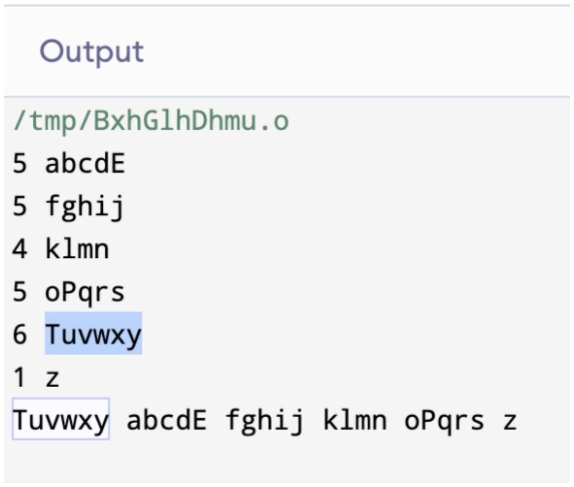


```
// 5. Determine the length of the next word
int len = rand() % (end_len - start_len + 1) + start_len;
```

This line of code is where the issue lies, the program is taking a random number between the inputted ranges as the length of the next chunk, this code should instead be alternating between the two inputted range values.

When feeding it "abcdEfghijklmnoPqrsTuvwxyz" and "4-6" as the parameters, the program outputs this: "abcdE fghij klmn oPqrs Tuvwxy z" if we look at the chunk sizes before they were sorted, it created chunks with length 6, 5, 5, 4, 5 and 1. Not following the proper alternation sequence of 4, 6, 4, 6, 4, 1.

Here is an image of the output with added console logs showing the chunk size before sorting.



Output

```
/tmp/BxhGlhDhmu.o
5 abcdE
5 fghij
4 klmn
5 oPqrs
6 Tuvwxy
1 z
Tuvwxy abcdE fghij klmn oPqrs z
```

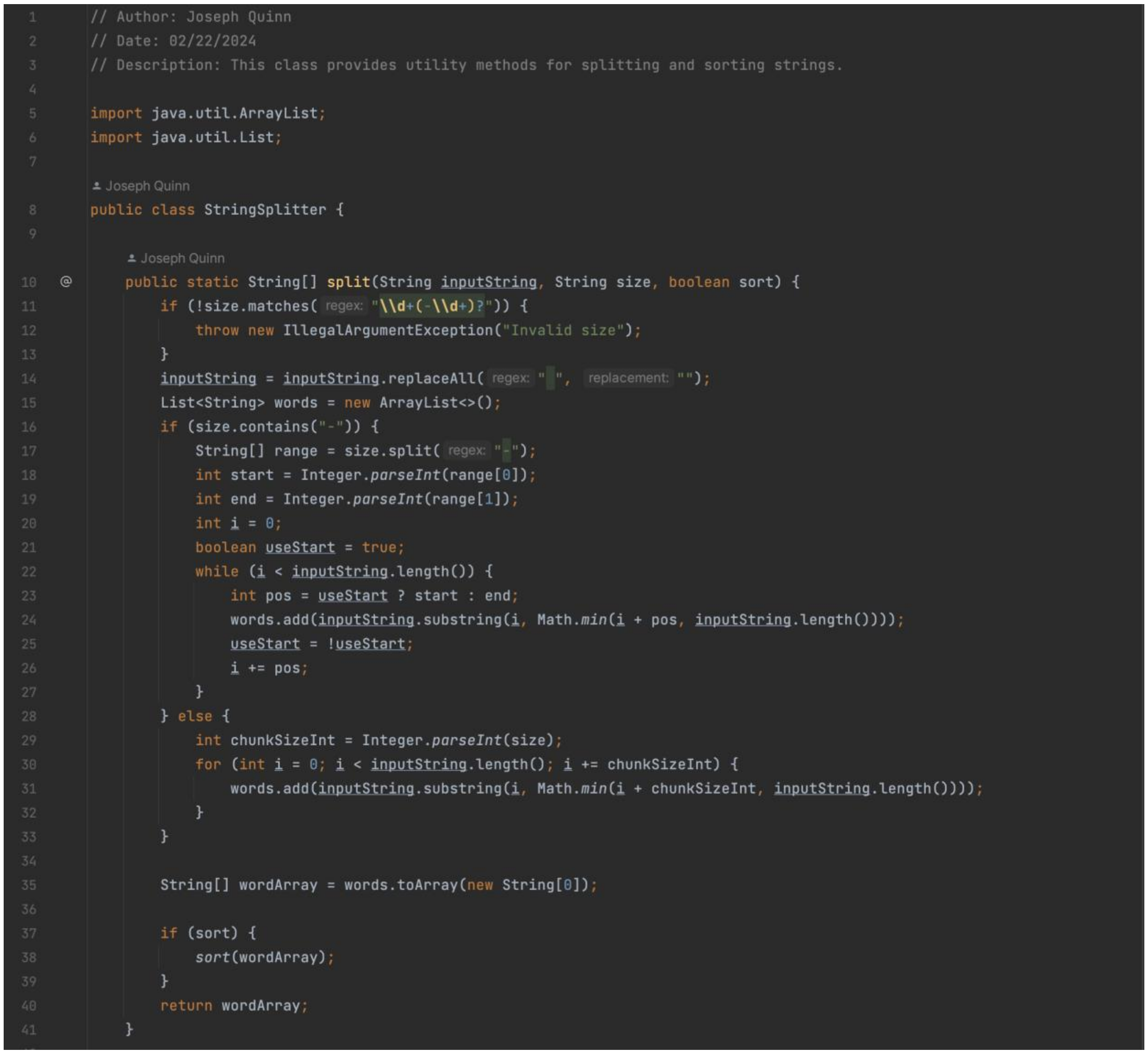
Section 8: Sorting the List:

While the function does properly sort the chunks, according to the prompt some datasets need to be sorted while others don't. Currently there is no way to set this within the function and it will automatically sort all chunks before returning them.

Part c:

Word Size

The first thing I updated was the system in determining whether the inputted size is a range or single number.



```
1   // Author: Joseph Quinn
2   // Date: 02/22/2024
3   // Description: This class provides utility methods for splitting and sorting strings.
4
5   import java.util.ArrayList;
6   import java.util.List;
7
    Joseph Quinn
8   public class StringSplitter {
9
        Joseph Quinn
10  @   public static String[] split(String inputString, String size, boolean sort) {
11          if (!size.matches( regex: "\\d+(-\\d+)?")) {
12              throw new IllegalArgumentException("Invalid size");
13          }
14          inputString = inputString.replaceAll( regex: " ",  replacement: "");
15          List<String> words = new ArrayList<>();
16          if (size.contains("-")) {
17              String[] range = size.split( regex: "-");
18              int start = Integer.parseInt(range[0]);
19              int end = Integer.parseInt(range[1]);
20              int i = 0;
21              boolean useStart = true;
22              while (i < inputString.length()) {
23                  int pos = useStart ? start : end;
24                  words.add(inputString.substring(i, Math.min(i + pos, inputString.length())));
25                  useStart = !useStart;
26                  i += pos;
27              }
28          } else {
29              int chunkSizeInt = Integer.parseInt(size);
30              for (int i = 0; i < inputString.length(); i += chunkSizeInt) {
31                  words.add(inputString.substring(i, Math.min(i + chunkSizeInt, inputString.length())));
32              }
33          }
34
35          String[] wordArray = words.toArray(new String[0]);
36
37          if (sort) {
38              sort(wordArray);
39          }
40          return wordArray;
41      }
```

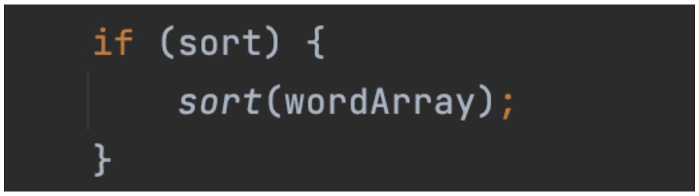
My method first checks to see if the inputted size parameter is valid, on line 11 it checks to see if the size string is either in range format (num-num), or single format (num). If not, it throws the error “Invalid Size”.

Then, on line 9, if no error is thrown. The method checks if the input parameter contains the “-“ char. If it does it enters the if branch. It then creates chunks based on the first range value, it adds this word to the words array and uses the useStart Boolean to keep track of which range value to use. This alternates so the chunk sizes alternate between the ranges inputted. This will continue until the word is fully chunked.

Sorting

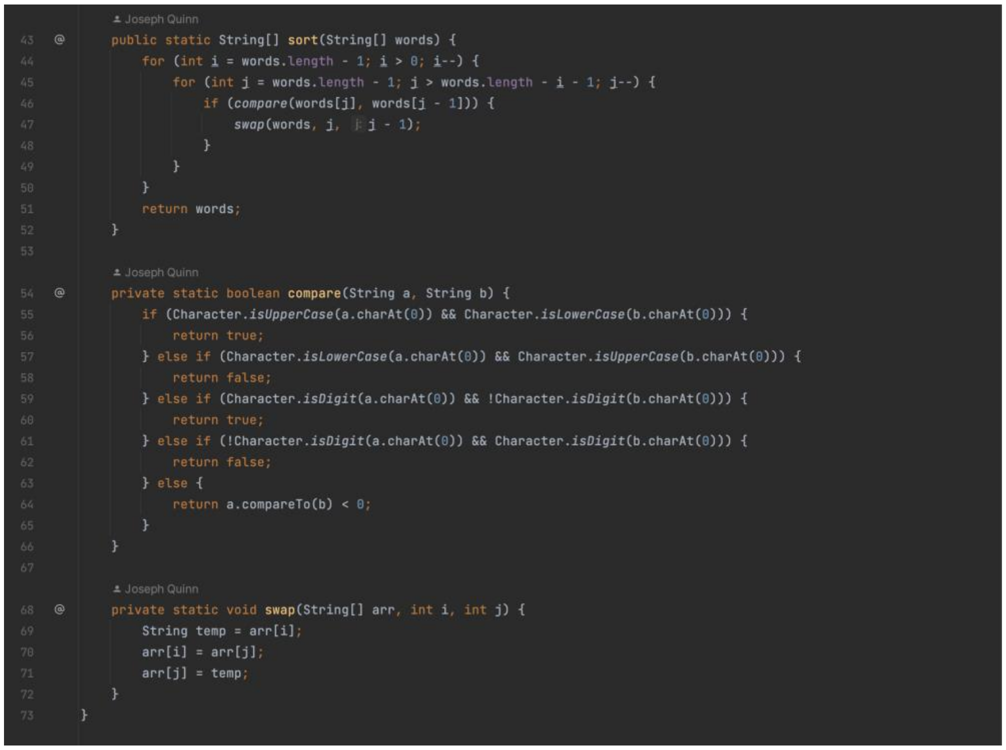
Because the generated solution has no way to determine if the words need to be sorted or not, I added a third parameter to the method being a Boolean type called sort, after the word chunks are created this would allow you to control weather or not the word array needs to be sorted.

To do this, I would create a method in my StringSplitter class called sort, if passed with a true parameter, it will sort the array after chunking, if passed with false, it won't sort.



```java
if (sort) {
    sort(wordArray);
}
```

The sorting method would take a string array parameter and return the sorted array following the prompt guidelines.

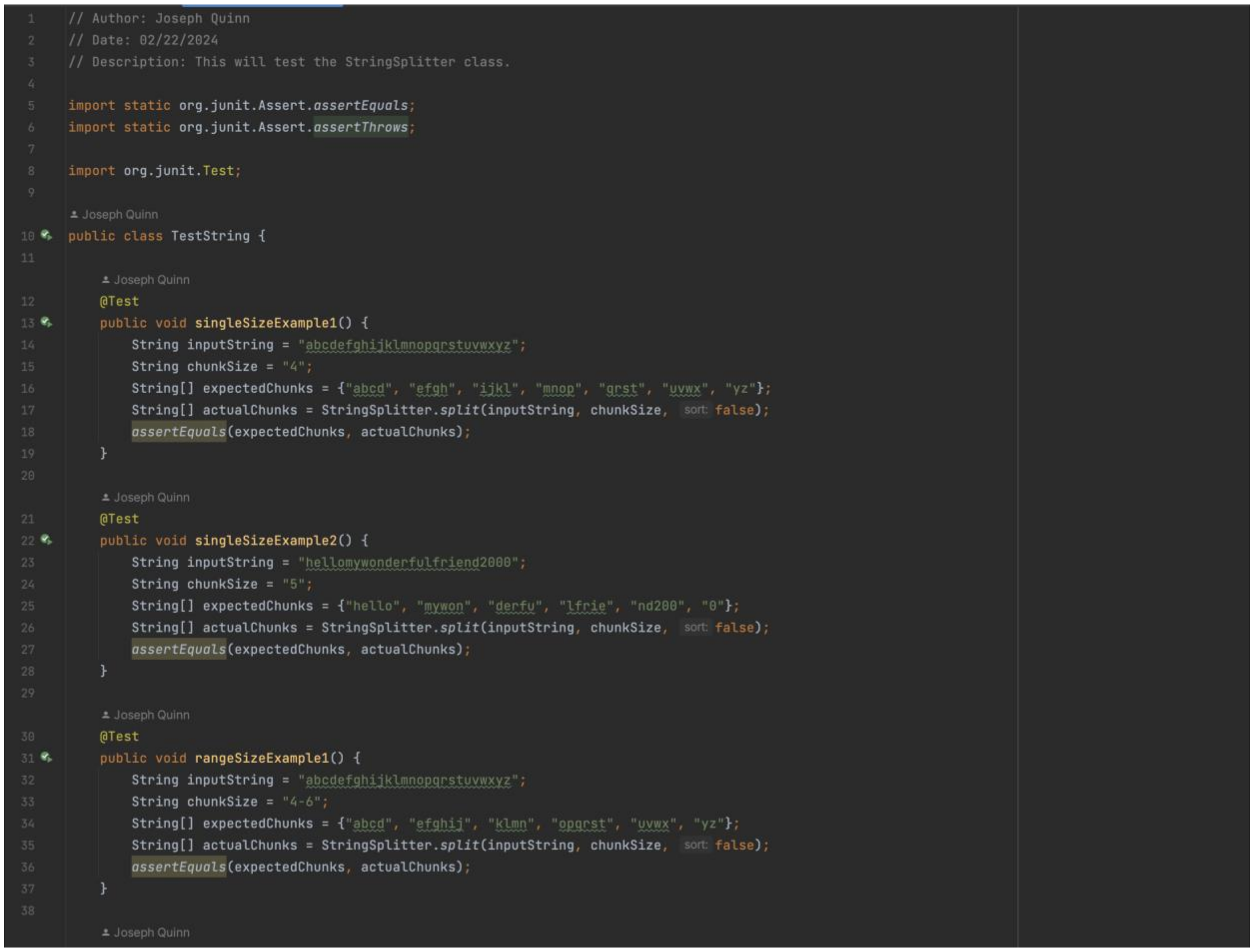


```java
    Joseph Quinn
    public static String[] sort(String[] words) {
        for (int i = words.length - 1; i > 0; i--) {
            for (int j = words.length - 1; j > words.length - i - 1; j--) {
                if (compare(words[j], words[j - 1])) {
                    swap(words, j, j - 1);
                }
            }
        }
        return words;
    }

    Joseph Quinn
    private static boolean compare(String a, String b) {
        if (Character.isUpperCase(a.charAt(0)) && Character.isLowerCase(b.charAt(0))) {
            return true;
        } else if (Character.isLowerCase(a.charAt(0)) && Character.isUpperCase(b.charAt(0))) {
            return false;
        } else if (Character.isDigit(a.charAt(0)) && !Character.isDigit(b.charAt(0))) {
            return true;
        } else if (!Character.isDigit(a.charAt(0)) && Character.isDigit(b.charAt(0))) {
            return false;
        } else {
            return a.compareTo(b) < 0;
        }
    }

    Joseph Quinn
    private static void swap(String[] arr, int i, int j) {
        String temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}
```

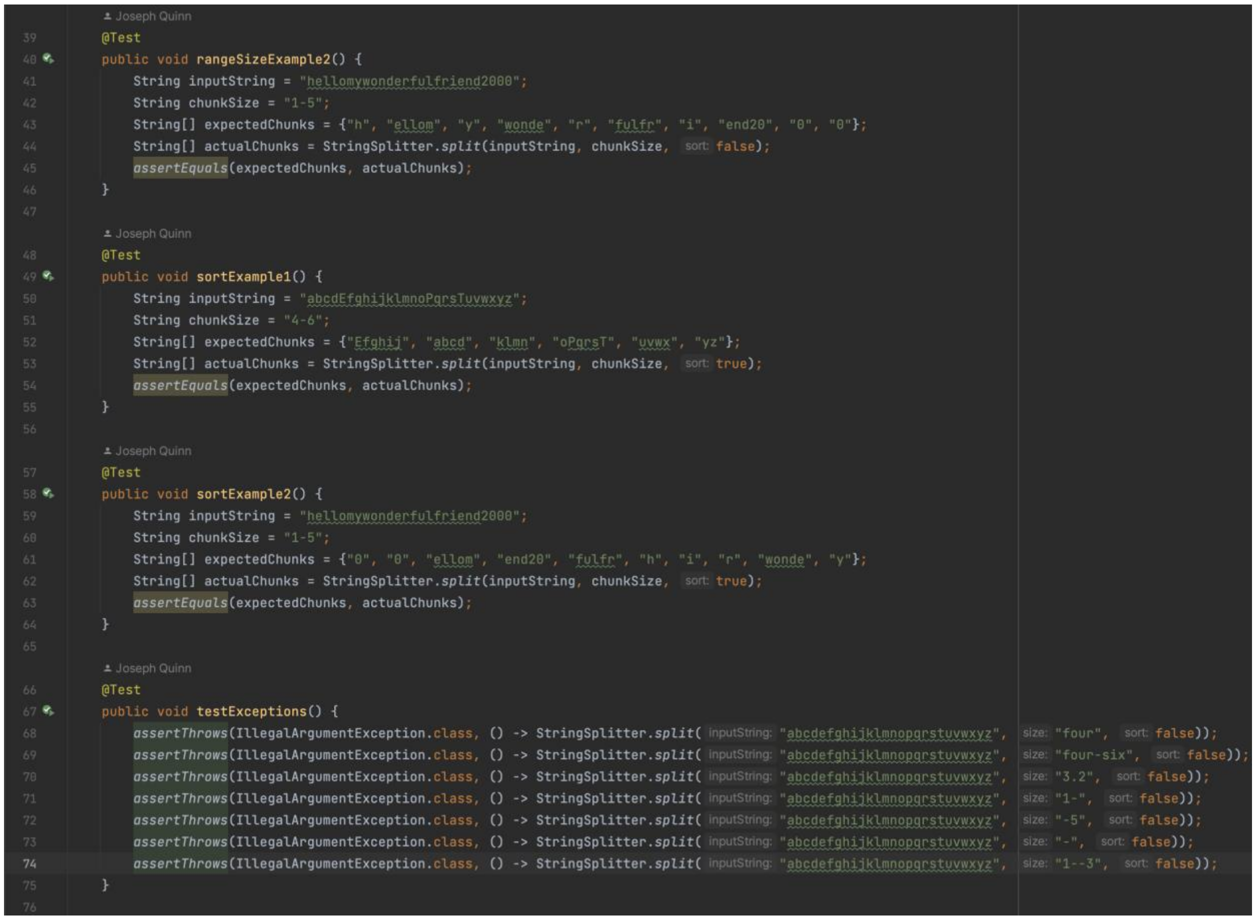
This method essentially is a bubble sort algorithm starting from the back of the array. It checks each pair through the compare method which determines the relative order of two strings based on predefined criteria. It first checks whether one string begins with an uppercase letter and the other with a lowercase one. If such a case arises, it returns true indicating that the uppercase string should precede the lowercase one. If one string starts with a number and the other with a letter, it returns true again signifying that the numerical string should come first. For cases where both strings start with either uppercase letters, lowercase letters, or numbers, it checks comparison based on alphabetical order. If the first string is alphabetically earlier than the second, it returns true indicating that the first string should be positioned before the second. This iterates throughout the array sorting it fully.

Testing

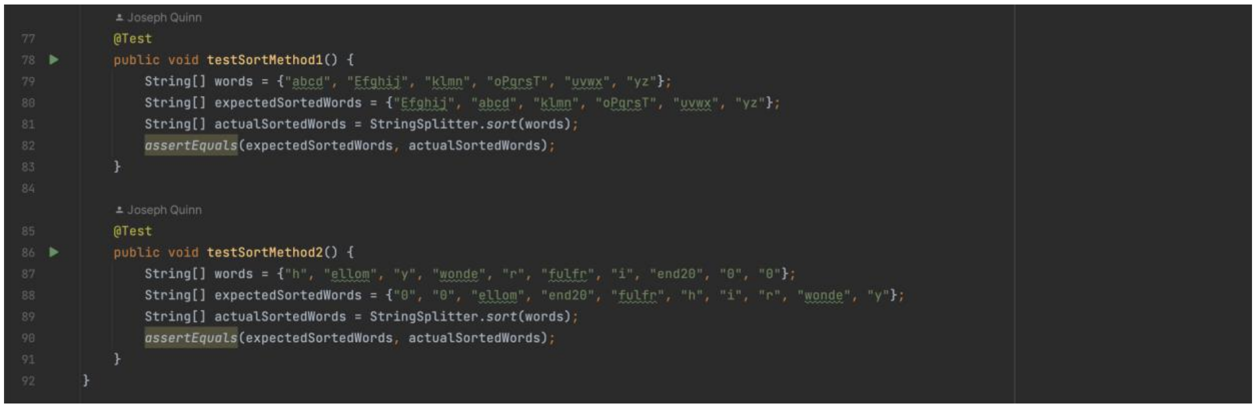
After applying all my fixes, I created some JUnits tests to test my methods. I wrote methods to test the provided prompt examples, the edge cases, and throwing exceptions.



```java
// Author: Joseph Quinn
// Date: 02/22/2024
// Description: This will test the StringSplitter class.

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertThrows;

import org.junit.Test;

Joseph Quinn
public class TestString {

    Joseph Quinn
    @Test
    public void singleSizeExample1() {
        String inputString = "abcdefghijklmnopqrstuvwxyz";
        String chunkSize = "4";
        String[] expectedChunks = {"abcd", "efgh", "ijkl", "mnop", "qrst", "uvwx", "yz"};
        String[] actualChunks = StringSplitter.split(inputString, chunkSize, sort: false);
        assertEquals(expectedChunks, actualChunks);
    }

    Joseph Quinn
    @Test
    public void singleSizeExample2() {
        String inputString = "hellomywonderfulfriend2000";
        String chunkSize = "5";
        String[] expectedChunks = {"hello", "mywon", "derfu", "lfrie", "nd200", "0"};
        String[] actualChunks = StringSplitter.split(inputString, chunkSize, sort: false);
        assertEquals(expectedChunks, actualChunks);
    }

    Joseph Quinn
    @Test
    public void rangeSizeExample1() {
        String inputString = "abcdefghijklmnopqrstuvwxyz";
        String chunkSize = "4-6";
        String[] expectedChunks = {"abcd", "efghij", "klmn", "opqrst", "uvwx", "yz"};
        String[] actualChunks = StringSplitter.split(inputString, chunkSize, sort: false);
        assertEquals(expectedChunks, actualChunks);
    }

    Joseph Quinn
```



```
    Joseph Quinn
    @Test
    public void rangeSizeExample2() {
        String inputString = "hellomywonderfulfriend2000";
        String chunkSize = "1-5";
        String[] expectedChunks = {"h", "ellom", "y", "wonde", "r", "fulfr", "i", "end20", "0", "0"};
        String[] actualChunks = StringSplitter.split(inputString, chunkSize,  sort: false);
        assertEquals(expectedChunks, actualChunks);
    }

    Joseph Quinn
    @Test
    public void sortExample1() {
        String inputString = "abcdEfghijklmnoPqrsTuvwxyz";
        String chunkSize = "4-6";
        String[] expectedChunks = {"Efghij", "abcd", "klmn", "oPqrsT", "uvwx", "yz"};
        String[] actualChunks = StringSplitter.split(inputString, chunkSize,  sort: true);
        assertEquals(expectedChunks, actualChunks);
    }

    Joseph Quinn
    @Test
    public void sortExample2() {
        String inputString = "hellomywonderfulfriend2000";
        String chunkSize = "1-5";
        String[] expectedChunks = {"0", "0", "ellom", "end20", "fulfr", "h", "i", "r", "wonde", "y"};
        String[] actualChunks = StringSplitter.split(inputString, chunkSize,  sort: true);
        assertEquals(expectedChunks, actualChunks);
    }

    Joseph Quinn
    @Test
    public void testExceptions() {
        assertThrows(IllegalArgumentException.class, () -> StringSplitter.split( inputString: "abcdefghijklmnopqrstuvwxyz",  size: "four",  sort: false));
        assertThrows(IllegalArgumentException.class, () -> StringSplitter.split( inputString: "abcdefghijklmnopqrstuvwxyz",  size: "four-six",  sort: false));
        assertThrows(IllegalArgumentException.class, () -> StringSplitter.split( inputString: "abcdefghijklmnopqrstuvwxyz",  size: "3.2",  sort: false));
        assertThrows(IllegalArgumentException.class, () -> StringSplitter.split( inputString: "abcdefghijklmnopqrstuvwxyz",  size: "1-",  sort: false));
        assertThrows(IllegalArgumentException.class, () -> StringSplitter.split( inputString: "abcdefghijklmnopqrstuvwxyz",  size: "-5",  sort: false));
        assertThrows(IllegalArgumentException.class, () -> StringSplitter.split( inputString: "abcdefghijklmnopqrstuvwxyz",  size: "-",  sort: false));
        assertThrows(IllegalArgumentException.class, () -> StringSplitter.split( inputString: "abcdefghijklmnopqrstuvwxyz",  size: "1--3",  sort: false));
    }
```



```
    Joseph Quinn
    @Test
    public void testSortMethod1() {
        String[] words = {"abcd", "Efghij", "klmn", "oPqrsT", "uvwx", "yz"};
        String[] expectedSortedWords = {"Efghij", "abcd", "klmn", "oPqrsT", "uvwx", "yz"};
        String[] actualSortedWords = StringSplitter.sort(words);
        assertEquals(expectedSortedWords, actualSortedWords);
    }

    Joseph Quinn
    @Test
    public void testSortMethod2() {
        String[] words = {"h", "ellom", "y", "wonde", "r", "fulfr", "i", "end20", "0", "0"};
        String[] expectedSortedWords = {"0", "0", "ellom", "end20", "fulfr", "h", "i", "r", "wonde", "y"};
        String[] actualSortedWords = StringSplitter.sort(words);
        assertEquals(expectedSortedWords, actualSortedWords);
    }
}
```

Part d:

Using a conversational AI like ChatGPT to solve problems has its advantages and pitfalls. One advantage is the rapid generation of solutions, which can save time and effort, especially for routine tasks or simple bulk work. However, a major pitfall is the lack of human oversight and critical thinking, leading to potential errors or inefficiencies in the solutions provided. ChatGPT likely came to the given solution by analyzing the prompt and generating code based on common programming patterns and structures and not considering any edge cases. We see this as the provide code does not function at all when it's not running with the optimal parameters.

Part e:

In my future tech career, I see myself incorporating AI to streamline repetitive tasks, and a streamlined search engine. I would use it as a tool to efficiently find trivial solutions to things like syntax questions or how specific functions work in different languages. I would utilize it to help me with the micro small aspects of my software projects, I would not use it for any sort of critical thinking or macro ideas unless it was for brainstorming purposes. Having an efficient search engine tailored to your needs can save you valuable time and allow you to focus on more complex and creative aspects of your work. By integrating these AI-driven tools into your workflow, you can stay utilize its advantages in the ever-evolving tech landscape and drive creativity withing your project.