Project: Implementing a Custom String ADT in Java

1. Overview

In this project, you will design and implement your own version of the Java String class. Your custom String ADT will encapsulate a character array as its internal data representation and provide a suite of constructors and methods that mimic the behavior of the standard Java `String` class. This project will help you understand the principles of object-oriented programming (OOP), encapsulation, immutability, and method implementation in Java.

2. Objectives

- **Understand ADTs:** Gain a deep understanding of Abstract Data Types by designing your own version of a widely used type.
- **Practice OOP Concepts:** Implement encapsulation, constructor overloading, and method overloading.
- **Array Manipulation:** Work with character arrays to manage and manipulate string data.
- **Error Handling:** Learn to manage exceptions and edge cases (e.g., index out of bounds).
- **Algorithm Design:** Implement algorithms for common string operations like searching, substring extraction, and case conversion.

3. Project Requirements

3.1 Constructors

Implement the following constructors:

1. **`String()`**
    - Initializes a newly created `String` object to represent an empty character sequence.
2. **`String(char[] value)`**
    - Allocates a new `String` so that it represents the sequence of characters currently contained in the character array argument.
3. **`String(String original)`**
    - Initializes a newly created `String` object to represent the same sequence of characters as the argument. In other words, create a deep copy of the given `String` object.

3.2 Methods

Your custom String class should include, but is not limited to, the following methods:

- **Character and Substring Access**
  - `char charAt(int index)` : Returns the character at the specified index.
  - `String substring(int beginIndex)` : Returns a substring from `beginIndex` to the end.
  - `String substring(int beginIndex, int endIndex)` : Returns a substring from `beginIndex` (inclusive) to `endIndex` (exclusive).
- **Comparison Operations**
  - `int compareTo(String anotherString)` : Compares two strings lexicographically.
  - `int compareToIgnoreCase(String str)` : Compares two strings lexicographically, ignoring case differences.
  - `boolean equalsIgnoreCase(String anotherString)` : Compares two strings for equality, ignoring case considerations.
- **Concatenation and Replacement**
  - `String concat(String str)` : Concatenates the specified string to the end of this string.
  - `String replace(char oldChar, char newChar)` : Returns a string resulting from replacing all occurrences of `oldChar` with `newChar` .
  - `String replace(CharSequence target, CharSequence replacement)` : Replaces each substring that matches the literal target sequence.
  - `String replaceAll(String regex, String replacement)` : Replaces all substrings matching the given regular expression.
  - `String replaceFirst(String regex, String replacement)` : Replaces the first substring matching the given regular expression.
- **Search Operations**
  - `boolean contains(CharSequence s)` : Returns true if this string contains the specified sequence of characters.
  - `int indexOf(int ch)` : Returns the index of the first occurrence of the specified character.
  - `int indexOf(int ch, int fromIndex)` : Returns the index of the first occurrence of the specified character, starting at the given index.
  - `int indexOf(String str)` : Returns the index of the first occurrence of the specified substring.

- `int indexOf(String str, int fromIndex)` : Returns the index of the first occurrence of the specified substring, starting at the given index.
- `int lastIndexOf(int ch)` : Returns the index of the last occurrence of the specified character.
- `int lastIndexOf(int ch, int fromIndex)` : Returns the index of the last occurrence of the specified character, searching backwards from the given index.
- `int lastIndexOf(String str)` : Returns the index of the last occurrence of the specified substring.
- `int lastIndexOf(String str, int fromIndex)` : Returns the index of the last occurrence of the specified substring, searching backwards from the given index.

- **Utility Methods**
  - `boolean isEmpty()` : Returns true if the string is empty.
  - `int length()` : Returns the length of the string.
  - `char[] toCharArray()` : Returns a new character array representing the string.
  - `String toLowerCase()` : Converts all characters to lower case using the default locale.
  - `String toLowerCase(Locale locale)` : Converts all characters to lower case using the given locale.
  - `String toUpperCase()` : Converts all characters to upper case using the default locale.
  - `String toUpperCase(Locale locale)` : Converts all characters to upper case using the given locale.
  - `String trim()` : Returns a string with leading and trailing whitespace removed.
  - `String toString()` : Returns this string object.
- **Prefix/Suffix Testing**
  - `boolean startsWith(String prefix)` : Tests if the string starts with the specified prefix.
  - `boolean startsWith(String prefix, int toffset)` : Tests if the substring starting at the specified index begins with the prefix.

3.3 Implementation Guidelines

- **Internal Representation:**
  Use a private character array to store the string data. Ensure that your class is immutable where applicable (e.g., methods should return new instances rather than modifying the existing object).