Tutorial No. 5

Title: String Handling with Java Programming.

Batch: SY-IT(B3) Roll No.: 16010423076 Tutorial No.: 5

Aim: To understand and explore string handling mechanism in Java with the following tasks:

1. Compare and Contrast String and StringBuffer Classes.
2. Designing a basic text editor application.

Resources needed: Java

Theory:

In Java, handling text data efficiently is crucial for many applications. Java provides two primary classes for manipulating strings: String and StringBuffer. Each serves a different purpose and offers distinct features.

1. The String Class

The String class in Java is used to represent a sequence of characters. It is immutable, meaning once a String object is created, its value cannot be changed.

Features:

- **Immutability**: Any modification to a String creates a new String object.
- **Thread Safety**: String objects are inherently thread-safe due to their immutability.

Methods:

1. **Creation and Initialization**
   - String s1 = "Hello";
     Creates a String object with the value "Hello".
2. **Length**
   - int length = s1.length();
     Returns the number of characters in the String.
3. **Concatenation**
   - String s2 = s1.concat(" World!");
     Concatenates " World!" to the end of s1.
4. **Substring**
   - String sub = s1.substring(1, 4);
     Extracts a substring from index 1 to index 4 (exclusive) from s1.
5. **Character Extraction**
   - char ch = s1.charAt(2);
     Returns the character at index 2 of s1.
6. **Comparison**
   - boolean isEqual = s1.equals("Hello");
     Compares s1 with "Hello" for equality.
7. **Case Conversion**
   - String upper = s1.toUpperCase();
     Converts s1 to uppercase.
8. **Finding Substring**

   - int index = s1.indexOf("lo");
     Finds the index of the first occurrence of the substring "lo".

9. **Trimming**
   - String trimmed = s1.trim();
     Removes leading and trailing whitespace.
10. **Replacing**
    - String replaced = s1.replace('l', 'L');
      Replaces all occurrences of character 'l' with 'L'.

Example String Class :

```
public class StringExample {
 public static void main(String[] args) {
 String str = "Hello World!";
 System.out.println("Length: " + str.length());
 System.out.println("Substring: " + str.substring(6));
 System.out.println("Uppercase: " + str.toUpperCase());
 System.out.println("Index of 'World': " + str.indexOf("World"));
  }
 }
```

2. The StringBuffer Class

The StringBuffer class is used for creating and manipulating mutable sequences of characters. Unlike String, StringBuffer objects can be modified after they are created.

Features:

· **Mutability**: Allows modifications to the character sequence without creating new objects. · **Thread Safety**: Methods are synchronized, making StringBuffer thread-safe.

Methods:

1. **Creation and Initialization**
   - StringBuffer sb = new StringBuffer("Hello");
     Creates a StringBuffer object with the initial content "Hello".
2. **Length**
   - int length = sb.length();
     Returns the length of the StringBuffer content.
3. **Append**
   - sb.append(" World!");
     Appends the specified string to the end of the StringBuffer.
4. **Insert**
   - sb.insert(5, " Java");
     Inserts the specified string at the specified position (index 5).
5. **Replace**
   - sb.replace(6, 11, "Universe");
     Replaces the substring from index 6 to 11 with "Universe".
6. **Delete**
   - sb.delete(5, 11);
     Deletes the characters from index 5 to 11 (exclusive).
7. **Reverse**
   - sbreverse();
     Reverses the sequence of characters.
8. **Capacity**
   - int capacity = sb.capacity();
     Returns the current capacity of the StringBuffer.
9. **Char At**
   - char ch = sb.charAt(2);

Returns the character at index 2.

Example StringBuffer Class :

```
public class StringBufferExample
 {
 public static void main(String[] args)
 {
 StringBuffer sb = new StringBuffer("Hello");
 sb.append(" World!");
 sb.insert(6, "Java ");
 sb.replace(11, 16, "Universe");
 System.out.println("Content: " + sb.toString());
 sb.reverse();
 System.out.println("Reversed: " + sb.toString());
  }
 }
```

Task: Prepare a document containing the answers of the following question:

Design a simple text editor where user could perform various text editing operations such as-

- Append text
- Insert text at a specific position
- Replace text in a given range
- Delete text from a specific position
- Display the current content

Note: -

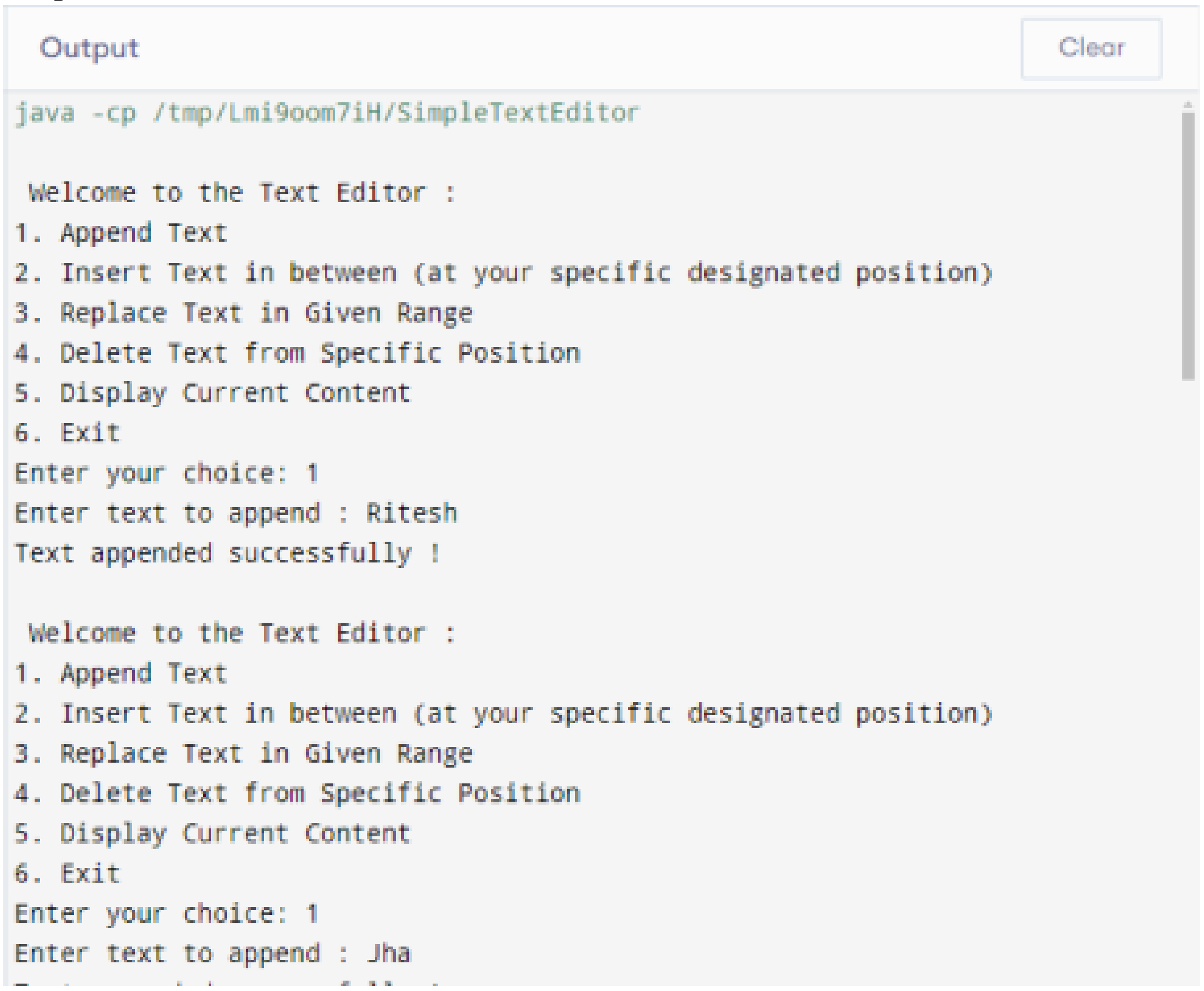
1. Use Menu driven Operation to perform task.
2. Use String or String Buffer Class and methods to complete the task.

CODE :

```
import java.util.Scanner;

public class SimpleTextEditor {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
StringBuffer text = new StringBuffer();
int choice;

do {
 System.out.println("\n Welcome to the Text Editor :");
 System.out.println("1. Append Text");
 System.out.println("2. Insert Text in between (at your specific designated position)");
System.out.println("3. Replace Text in Given Range");
 System.out.println("4. Delete Text from Specific Position");
 System.out.println("5. Display Current Content");
 System.out.println("6. Exit");
 System.out.print("Enter your choice: ");
 choice = scanner.nextInt();
 scanner.nextLine();
 switch (choice) {
 case 1:
 System.out.print("Enter text to append : ");
 String appendText = scanner.nextLine();
 text.append(appendText);
 System.out.println("Text appended successfully !");
 break;

 case 2:
 System.out.print("Enter text to insert : ");
 String insertText = scanner.nextLine();
 System.out.print("Enter position to insert the text : ");  int
insertPosition = scanner.nextInt();
 if (insertPosition >= 0 && insertPosition <= text.length()) {
text.insert(insertPosition, insertText);
 System.out.println("Text inserted successfully !");  }
 else {
 System.out.println("Invalid position !");
 }
 break;

 case 3:
 System.out.print("Enter start position of the range : ");  int
startPos = scanner.nextInt();
 System.out.print("Enter end position of the range : ");  int
endPos = scanner.nextInt();
 scanner.nextLine();
 if (startPos >= 0 && endPos <= text.length() && startPos <= endPos) {
System.out.print("Enter text to replace : ");
 String replaceText = scanner.nextLine();
 text.replace(startPos, endPos, replaceText);
 System.out.println("Text replaced successfully !");  }
 else {
 System.out.println("Invalid range !");
 }
 break;

 case 4:
 System.out.print("Enter start position to delete text: ");  int
deleteStart = scanner.nextInt();
 System.out.print("Enter end position to delete text: ");
 int deleteEnd = scanner.nextInt();
 if (deleteStart >= 0 && deleteEnd <= text.length() && deleteStart <= deleteEnd) {
text.delete(deleteStart, deleteEnd);
 System.out.println("Text deleted successfully.");
 }
 else {
```

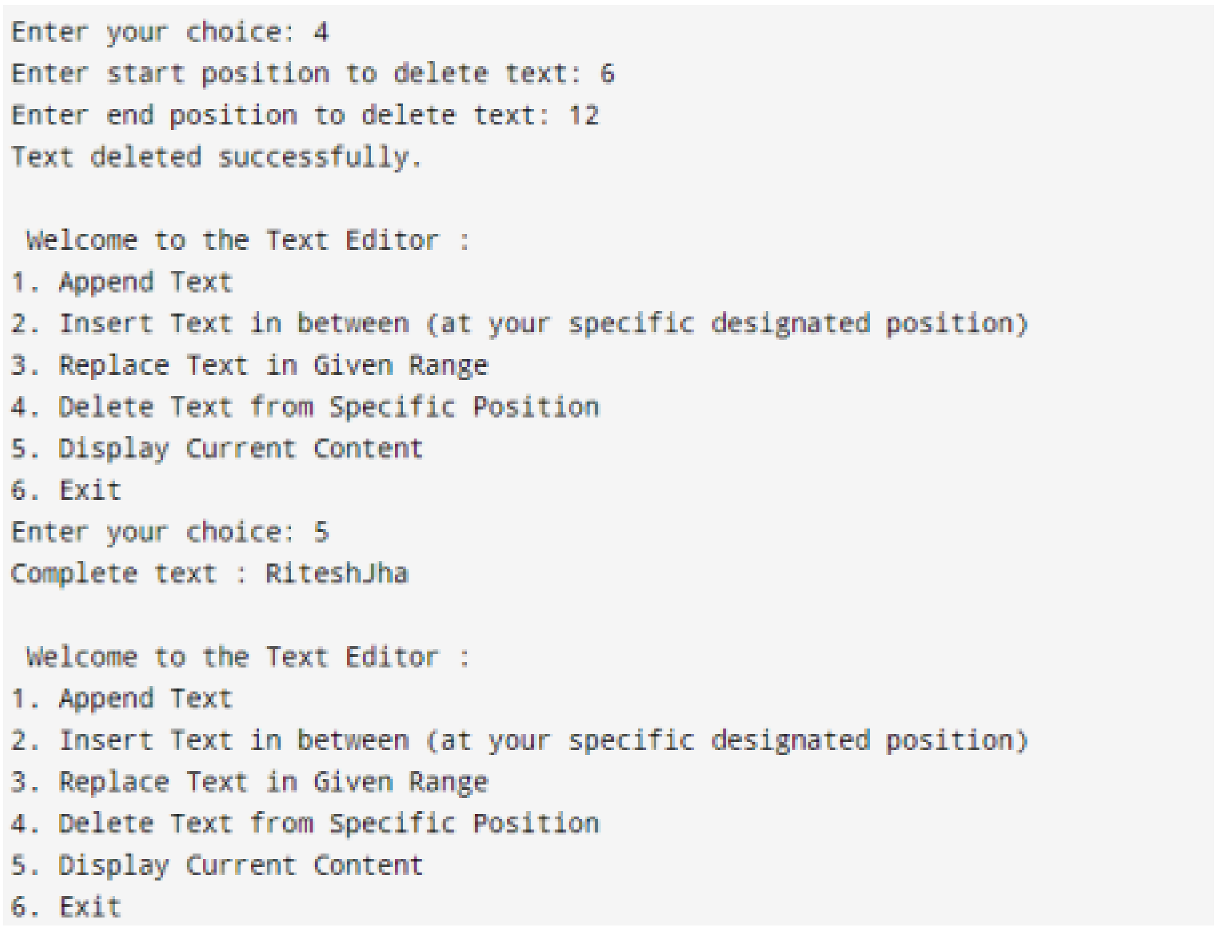
```
System.out.println("Invalid position.");
}
break;

case 5:
System.out.println("Complete text : " + text.toString());
break;

case 6:
System.out.println("Editor suspended successfully....");
break;

default:
System.out.println("Invalid choice. Please try again.");
break;
}
} while (choice != 6);

scanner.close();
}
}
```

Output :

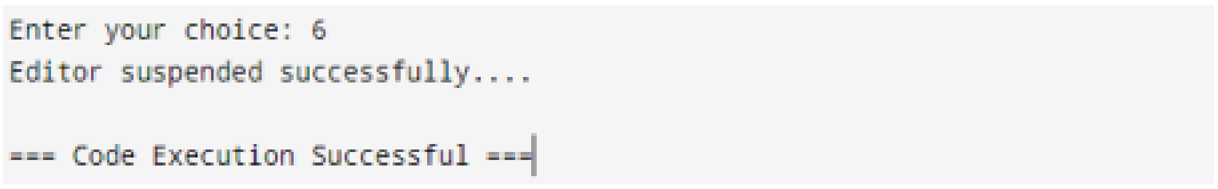


```
java -cp /tmp/Lmi9oom7iH/SimpleTextEditor

 Welcome to the Text Editor :
1. Append Text
2. Insert Text in between (at your specific designated position)
3. Replace Text in Given Range
4. Delete Text from Specific Position
5. Display Current Content
6. Exit
Enter your choice: 1
Enter text to append : Ritesh
Text appended successfully !

 Welcome to the Text Editor :
1. Append Text
2. Insert Text in between (at your specific designated position)
3. Replace Text in Given Range
4. Delete Text from Specific Position
5. Display Current Content
6. Exit
Enter your choice: 1
Enter text to append : Jha
Enter text to append : Jha
Text appended successfully !

 Welcome to the Text Editor :
1. Append Text
2. Insert Text in between (at your specific designated position)
3. Replace Text in Given Range
4. Delete Text from Specific Position
5. Display Current Content
6. Exit
Enter your choice: 5
Complete text : RiteshJha

 Welcome to the Text Editor :
1. Append Text
2. Insert Text in between (at your specific designated position)
3. Replace Text in Given Range
4. Delete Text from Specific Position
5. Display Current Content
6. Exit
```

```
Enter your choice: 2
Enter text to insert : Sudhir
Enter position to insert the text : 6
Text inserted successfully !

 Welcome to the Text Editor :
1. Append Text
2. Insert Text in between (at your specific designated position)
3. Replace Text in Given Range
4. Delete Text from Specific Position
5. Display Current Content
6. Exit
Enter your choice: 5
Complete text : RiteshSudhirJha

 Welcome to the Text Editor :
1. Append Text
2. Insert Text in between (at your specific designated position)
3. Replace Text in Given Range
4. Delete Text from Specific Position
5. Display Current Content
6. Exit
```



```
Enter your choice: 4
Enter start position to delete text: 6
Enter end position to delete text: 12
Text deleted successfully.

 Welcome to the Text Editor :
1. Append Text
2. Insert Text in between (at your specific designated position)
3. Replace Text in Given Range
4. Delete Text from Specific Position
5. Display Current Content
6. Exit
Enter your choice: 5
Complete text : RiteshJha

 Welcome to the Text Editor :
1. Append Text
2. Insert Text in between (at your specific designated position)
3. Replace Text in Given Range
4. Delete Text from Specific Position
5. Display Current Content
6. Exit
```



```
Enter your choice: 6
Editor suspended successfully....

=== Code Execution Successful ===
```

Post Lab Questions:-

1. What are the difference between string builder and String buffer class? •
Mutability :
Both StringBuilder and StringBuffer are mutable, meaning their contents can be modified after creation, unlike `String`, which is immutable.

• Thread-Safety :
StringBuffer`is synchronized, making it thread-safe. This means it can be safely used by multiple threads simultaneously, but it comes with a performance cost. StringBuilder is not synchronized, making it faster but not thread-safe. It should be used when thread safety is not a concern.

• Performance :
StringBuilder generally performs better than StringBuffer because it does not have the overhead of synchronization.

2. How to use equals() for case-sensitive comparison and equalsIgnoreCase() for case insensitive comparison?

• equals() Method: This method is used for case-sensitive comparison of two strings. It returns `true` if both strings are exactly the same, including case.

```
String str1 = "Java”;
String str2 = "java";
```

```
boolean result = str1.equals(str2); // result will be false
```

• equalsIgnoreCase() Method: This method is used for case-insensitive comparison of two strings. It returns `true` if both strings are the same, ignoring case.

```
String str1 = "Java";
String str2 = "java";
boolean result = str1.equalsIgnoreCase(str2); // result will be true
```

3. What is role of trim, split and format method in decorating java output formatting? Explain with suitable example.

• trim() Method: This method removes any leading and trailing whitespace from a string. It's useful when formatting output where extra spaces are not desired.

• split() Method: This method splits a string into an array of substrings based on a given delimiter. It is commonly used to break down a string for formatting or further processing.

• format() Method: This method returns a formatted string using specified format specifiers. It's useful for creating output that adheres to a particular layout or style.

4. Modify the code to ensure that the concatenated string as –"JAVA Programming "is correctly displayed in below code

```
public class StringConcatenation {
public static void main(String[] args) {
 String str = "Java";
 str.concat(" Programming");
System.out.println("String: " + str);
```

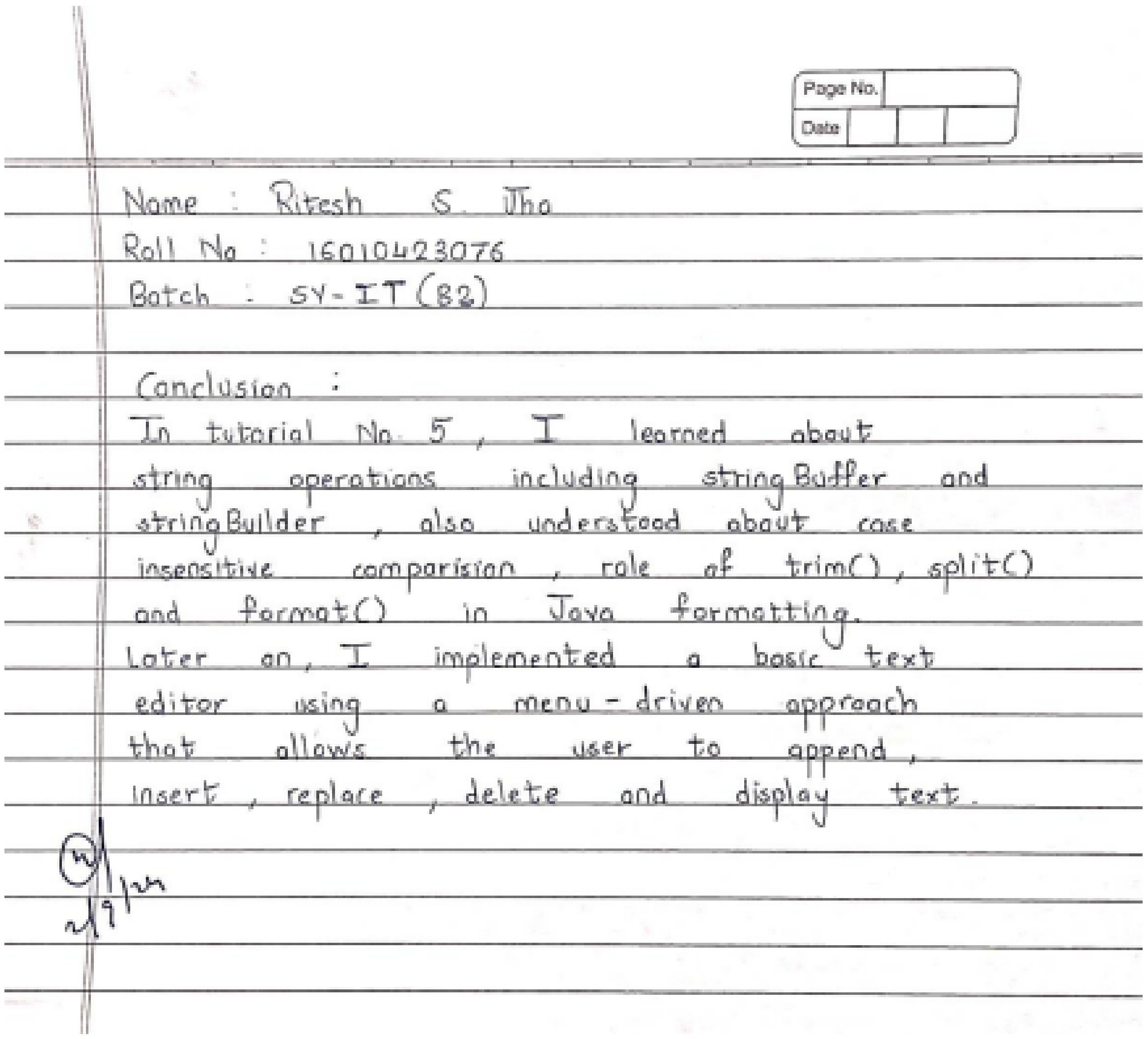
Corrected Code without without :

```
        public class StringConcatenation {
        public static void main(String[] args) {
         String str = "Java";
         str = str.concat(" Programming");
         System.out.println("String: " + str);
         }
}
```

Outcomes:

CO2: Apply String manipulation functions ,inheritance and polymorphism using Java programming

Conclusion: (Conclusion to be based on the outcomes achieved)



Page No.
Date

Name : Ritesh S. Jha
Roll No : 16010423076
Batch : SY-IT (B2)

Conclusion :
In tutorial No. 5, I learned about string operations including stringBuffer and stringBuilder, also understood about case insensitive comparision, role of trim(), split() and format() in Java formatting.
Later on, I implemented a basic text editor using a menu-driven approach that allows the user to append, insert, replace, delete and display text.

[illegible] 2/9/24

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References Books

1. Herbert Schildt; JAVA The Complete Reference; Seventh Edition, Tata McGraw Hill Publishing Company Limited 2007.
2. Java 7 Programming - Black Book : Kogent Learning Solutions Inc. 3. Sachin Malhotra, Saurabh Chaudhary "Programming in Java", Oxford University Press, 2010 4. Jaime Nino, Frederick A. Hosch, 'An introduction to Programming and Object Oriented Design using Java', Wiley Student Edition.