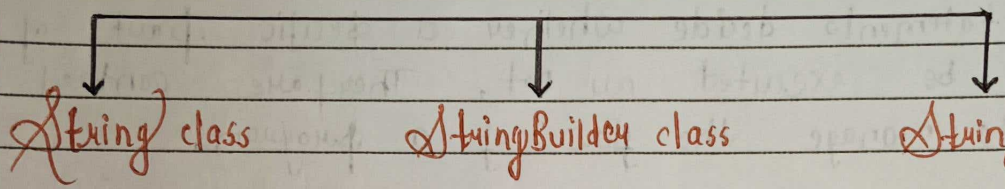
07/02/2024

String

"A string is a sequence of characters or an array of characters." In Java, string is basically an object, that represents a sequence of char values.

There are three classes availble in Java to create a String :-



String Class

→ The "String" class in Java is a fundamental class for working with strings. 'String' class is part of the "java.lang" package, which is automatically imported into every Java program. This means you do not need to explicitly import the 'String' class to use in your code.

→ java.lang

↳ java → This is a top-level package in Java. "java" package contains core java libraries provided by Oracle.

↳ lang → This is a sub-package within the 'java' package.

→ String class is used to create strings which are immutable, meaning once a 'String' object is created, its value cannot be changed.

→ Strings are used extensively in Java APIs and libraries as well as in communication between different systems. The ubiquity of the "String" class ensures compatibility and interoperability across various Java applications and environments.

→ String literal Pool

Java maintains a string pool, which is a pool of unique string literals. When a string literals is created, JVM first checks the pool. If the string already exists in the pool, a reference to the existing string is returned instead of creating a new one. This helps conserve memory and optimize performance, especially when dealing with frequently used string literals. String literal pool is inside heap memory.

→ String class syntax in Java :-

```
public final class String extends Object, implements charsequence,
serializable, comparable
{
// code
}
```

→ How to create a string object

There are two ways to create String object using string class.

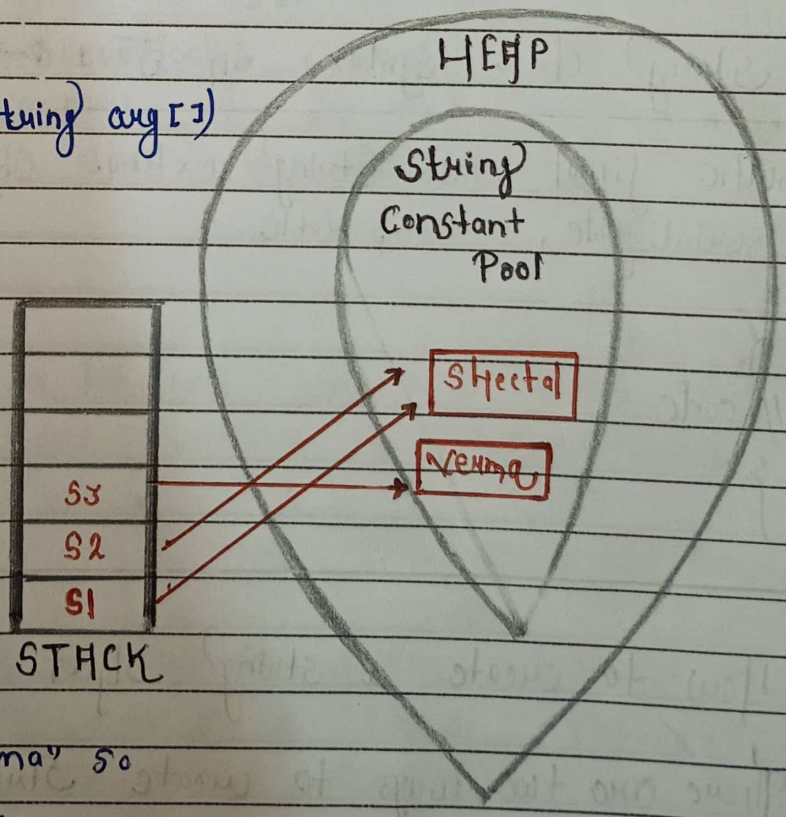
→ By string literal
→ By new keyword

Java String literal is created by using double quotes. Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string does not exist in the pool, a new string instance is created and placed in the pool.

String literal syntax :-

String Variable_Name = "String_literal";

eg class string
{
public static void main(String arg[])
{
String s1 = "Sheetal";
String s2 = "Sheetal";
String s3 = "Verma";
S.O.P (s1);
}
}



Here s1 and s2 have same object "Sheetal" so no new object is created inside SCP. And there is no object "Verma" so memory is allocated for it.

Advantages

This method of creating string literal is used when multiple references are pointing to the same object. There is no need to create same object (string literal) again if it already exist, it will save memory.

Disadvantages

When we change the value of one reference variable it will affect all the reference variables pointing to that object. This will lead to ambiity, ambiguity, and inconsistency.

2. By new Keyword

We can also create string using "new" operator. It will use when you want to create strings dynamically at runtime. It creates an object on the Heap memory, regardless of whether an identical string already exists in the string pool.

eg) Cl

Syntax :-

```
String variable = new String("string literal");
```

:- It is not necessary to always initialize the string variable.

Advantage

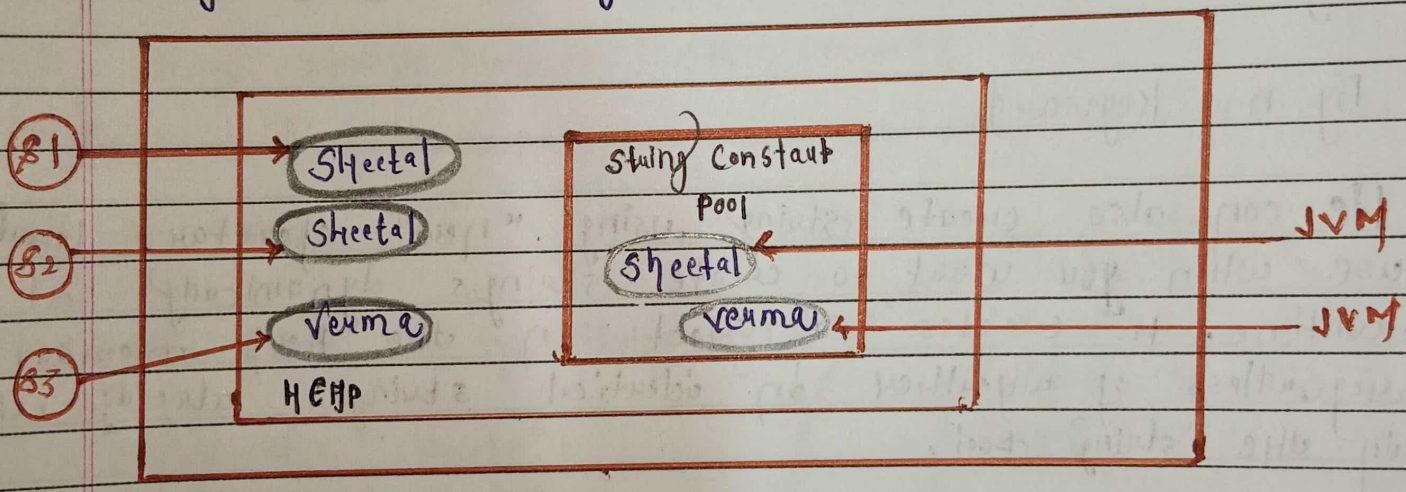
When we modify or change one reference variable it will not affect other variables even though they have same value.

Disadvantage

It will create new string each time even if we have that string already present in the memory [Heap]. It will consume more memory, and each time it will create two objects one inside heap and one inside SCP. (If SCP if that object not present).

eg

```
String s1 = new String("Sheetal"); (2) → object
String s3 = new String("Verma"); (2) → object
String s2 = new String("Sheetal"); (1) → object
```



Why Strings are immutable in Java ?

1. Security
Strings are often used to store sensitive information like passwords or cryptographic keys. It will helps to prevent malicious modification of the sensitive data.

2. Memory efficient

3. Hashing and Caching

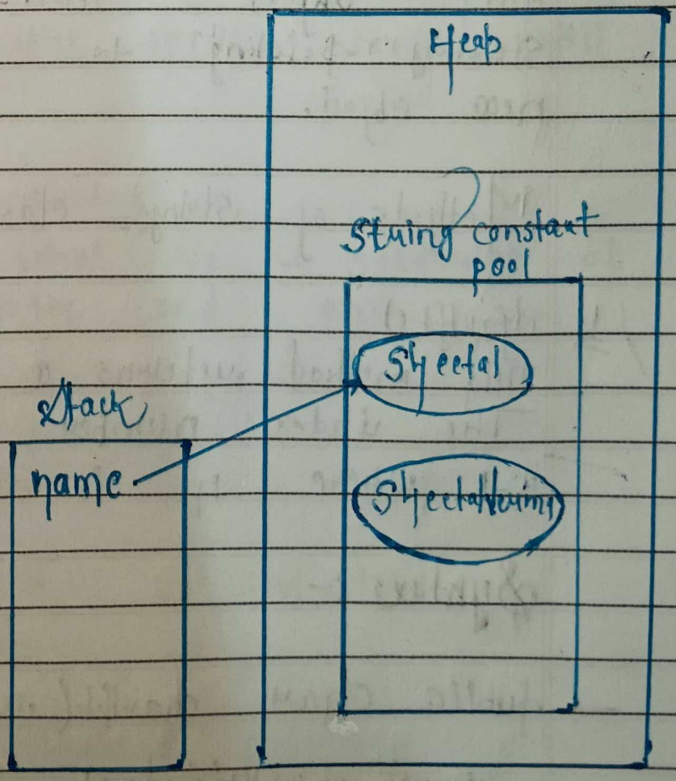
Since strings are immutable, their hash codes can be cached and computed once when the string is created. This can improve the performance of operations like hash table lookups, where hash codes are used for quick retrieval.

4. Thread safety

Immutable objects like strings are inherently thread safe. Since their state cannot be changed after creation, multiple threads can safely access them concurrently without the risk of one thread modifying the state while another thread is using it. It helps in multithreaded programming.

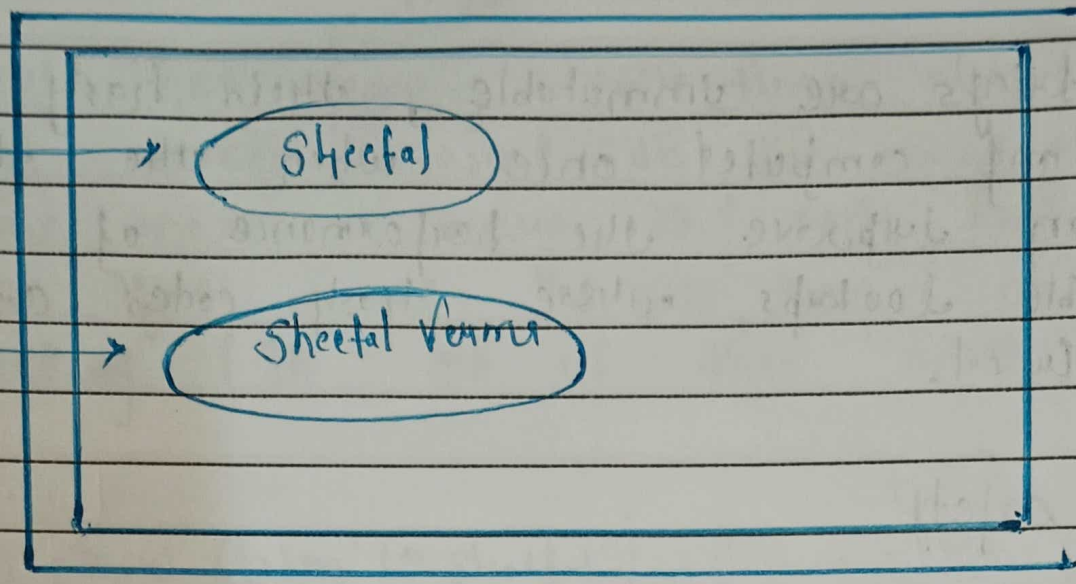
How String objects are immutable ?

```
class ImmutableString
{
p s v main(String a[])
{
String name = "Sheetal";
name.concat("Verma");
S.o.p(name);
}}
```



Two objects are created "Sheetal" and "sheetal verma" but the reference 'name' pointing to "sheetal" and no reference for "sheetal verma".

eg String name = "Sheetal";
name = name.concat("Verma");
S.O.P(name);



→ Here we explicitly assign new value to 'na
but the original object "sheetal" is
is memory.

→ In both the cases original "string" is

→ In future whenever a new reference variable
with object "Sheetal" that reference
starting pointing to "sheetal" instead of
new object.

Methods of String class

1. charAt()

This method returns a char value at the given index. The index number starts with 0 and goes to n-1, where n is the length of the string.

Syntax :-

→ public char charAt(int index) (signature)

→ String.charAt(index)

eg :- String name = "Sheetal";
Char ch = name.charAt(0);
S.O.p(ch);

output :-
S

eg :- String name = "Sheetal";
char ch = name.charAt(0);
S.O.p(ch);

output :-
S

2. length()
This method finds the length of a string. It will return an integer value. It counts the number of characters in the string.

→ public int length()
→ string.length()

eg int length = name.length();
S.O.p(length);

output :-
7

3. substring()
This method returns a part of the string. We have two categories of this method.

→ substring(int beginIndex)
It will return a new string that is a substring of this string, starting from the specific index till the end of the string.

→ substring(int beginIndex, int endIndex)
It will return a new string that is a substring of given string start from beginIndex and end at index (endIndex - 1).

→ public String substring(int startIndex)
or
→ public String substring(int startIndex, int endIndex)

→ string.substring(startIndex);
→ string.substring(startIndex, endIndex);

eg
```
String name = "sheetal";
String s = name.substring(2);
S.o.p (s);

String s1 = name.substring(0,4);
S.o.p (s1);
```

output :-
eetal
shee

4. format() method
This method allows you to create formatted strings using placeholders and arguments. This method returns a String.

→ Signature
public static String format(String format, arguments)
or
public static String format(Locale locale, String format, arguments)

We can also set width, padding of any value using format() method.

eg
```
String name = "sheetal";
int age = 25;

String s = String.format("Your name is %s", name);
String s1 = String.format("Your age is %d", age);

S.o.p (s);
S.o.p (s1);
```

output :-
Your name is sheetal
Your age is 25

5. contain()

This method searches the sequence of characters in given string. It returns true if the sequence of char values is found otherwise returns false.

signature

public boolean contains(charsequence)

syntax → string.contain("sequence");

eg String s = "Hello sheetal verma here!";
boolean b = s.contains("sheetal");
S.o.p(b);

output :
true

6. join()

This method returns a string joined with a given delimiter. There are two types of join() method in Java. Here delimiter means the value you wants between each string.

eg String date = String.join("/", "12", "02", "2024");
S.o.p(date);

↑ delimiter

output : 12/02/2024

7. isEmpty

This method checks if the input string is empty or not. It return true if the length is zero otherwise false.

public boolean isEmpty()

syntax → string.isEmpty();

eg String name = "";
S.o.p(name.isEmpty());

output :
true

8. equals()

This method compares the two given string based on the content of the string. If all characters are matched it returns true otherwise false.

Syntax :- string1.equals(string2)

eg
```
String s1 = "Sheetal";
String s2 = "Sheetal";
String s3 = "verma";

s.o.p(s1.equals(s2));
s.o.p(s2.equals(s3));
```

output :-
true
false

9. replace()

→ This method returns a string replacing all the old characters to new characters.

→ JDK 1.5 replace() method also replace a sequence of char values.

Signature → public String replace(char oldchar, char newchar)
and
public String replace(charSequence target, charSequence newsquence)

Syntax :- String.replace('oldchar', 'newchar');

String.replace("stringold", "newstring");

eg
```
String s1 = "Sheetal verma";
String w = s1.replace('e', 'i');
s.o.p(w)
```

output :-
Shiital verma

eg String s2 = " My name was Sheetal Verma";
String s = s2.replace (" was", "is");
S.O.p (s);

output :- My name is Sheetal Verma.

10. concat()
This method combines two strings. The concat() method is used to concatenate one string to the end of another. It returns a new string that represents the concatenation of the two strings.

String result = string1.concat(string2);

This will concat string2 at the end of string1.

eg String name = " Sheetal";
String surname = " Verma";
String fullname = Name.concat(surname);
S.O.p (fullname);

output :-
Sheetal Verma.

11. equalsIgnoreCase()
This method compares the two strings on the basis of the content of the string irrespective of the case. (upper or lower case). It return true if all characters found otherwise it returns false.

eg String s1 = " sheetal";
String s2 = " Sheetal";
S.O.p (s1.equalsIgnoreCase (s2))

output :-
true

12. split()

This method is used to split a string into an array of substrings based on a specified delimiter.

Syntax → public String[] split(String regularExpression)

↑ return array of strings

↑ pattern to match with

Calling syntax → string.split("expression");

↑ delimiter

eg String week = "Monday, Tuesday, Wednesday, Thursday,";
String[] names = week.split(",");
for(String element : names)
{
s.o.p(element);
}
}

output :-
Monday
Tuesday
Wednesday
Thursday

→ We can also limit number of strings in an array.

for(String element : week.split(",", 3))
{
s.o.p(element);
}

output :-
Monday
Tuesday
Wednesday, Thursday

13. intern()

This method is used to obtain the canonical representation of the string. (Canonical R. Means a string stored in the string Constant pool.)

→ The intern() method, when called on a string, checks whether that string is already in the pool. If it is, it returns a reference to the string from the pool. If not, it adds the string to the pool and returns a reference to it.

→ Signature : public String intern()

→ Syntax : String.intern();

eg String s1 = new String("sheetal"); // heap
String s2 = "sheetal"; // SCP
String s3 = s1.intern(); // s3 points to the string at SCP.
S.o.p(s3 == s2); // output → true.

14 indexOf()
This method returns the position of the first occurrence of the specified character or string in a specified string. There are four overloaded indexOf() method in Java.

Signature of indexOf() methods :-

1. int indexOf(int char)
2. int indexOf(int char, int fromIndex)
3. int indexOf(String str)
4. int indexOf(String str, int fromIndex)

→ str : A String value, representing the string to search.
→ fromIndex : An int value, representing the index position to start the search from.
→ char : An int value, representing a single character, or a Unicode value.

→ It returns '-1' if given char or string not found.

→ It will search complete string and return the starting index of search string in the given string.

eg String line = "Hello Sheetal verma";
int index = line.indexOf('e');
S.O.P (index);

int index1 = line.indexOf('e', 5);
S.O.P (index1);

output :-
1
8

15. toLowerCase()
This method will convert all characters of the string into lower case.

public String toLowerCase()

16. toUpperCase()
This method will convert all characters of the string into upper case.

public String toUpperCase()

eg String s = "Sheetal VERMA";
S.O.P (s.toUpperCase());
S.O.P (s.toLowerCase());

output
SHEETAL VERMA
sheetal verma

17. trim()

The trim() method is a string class method that removes any leading and trailing whitespace from a string and returns the trimmed string. The String trim() does not omit middle spaces. The Unicode value of space character is '\u0020'.

→ public String trim()

→ String.trim()

eg String s = " Sheetal verma ";
s.o.p(s);
s.o.p(s.trim());

output
Sheetal verma
Sheetal verma

18. valueOf()

This method converts different types of values into string. Using valueOf() method we can convert int to string, long to string, boolean to string, float to string, double to string, object to string, char array to string

→ static String valueOf(int value)
→ static String valueOf(long value)
→ static String valueOf(float value)
→ static String valueOf(double value)
→ static String valueOf(char value)
→ static String valueOf(boolean value)
→ static String valueOf(Object value)

eg int i = 20;
String s = String.valueOf(i);
s.o.p(s);

output
20

19 codePointAt()

→ This method returns the Unicode value of the character at the specified index in a string.

→ public int codePointAt(int index)

→ If we pass index value greater or equal to length (or negative) of string it will throw IndexOutOfBound exception.

→ Calling syntax : String.codePointAt(index)

eg
```
String s = "Sheetal";
int r = s.codePointAt(0);
S.o.p (r);
```
output
83

20 codePointBefore()

→ This method returns the unicode value of the character before the specified index.

21. endsWith()

This method is used to determine whether a string ends with the specified suffix. if suffix found it returns true otherwise it returns false.

→ public boolean endsWith(String variable)

eg
```
String name = "Sheetal";
S.o.p (name.endsWith("l"));
S.o.p (name.endsWith("de"));
S.o.p (name.endsWith("Sheetal"));
```
Output
true
false
true

22. startsWith()

This method in Java's String class is used to determine whether a string starts with the specified prefix. If yes it return 'true' otherwise it returns 'false'.

→ public boolean startsWith(String variableName)

→ Calling syntax : String_variable_Name.startsWith("prefix");

23. CompareTo()

This method in Java's String class is used to compare two strings lexicographically. It returns an integer value which can be positive, negative or zero. It compares strings on the basis of the unicode value of each character in the strings.

→ If the invoking string (the string calling the method) is lexicographically less than the string passed as an argument.	Negative integer.
→ If the invoking string is lexicographically greater than the string passed as an argument.	Possitive integer
→ If both the string are equal, it return	zero

→ public int compareTo(String String_variable)

→ Lexicographically refers to the order in which words or strings are sorted alphabetically. Also called alphabetical order or dictionary order.

→ lexicographic ordering is case sensitive and uppercase letters typically come before lowercase letters.

→ Comparison also may contain characters such as digits or special symbols, based on their Unicode values.

→ We can also compare empty string, there are three cases.

case1 → calling string is empty	Negative
case2 → argument string is empty	Positive
case3 → both string's are empty	zero

24. lastIndexOf()

This method will returns the index of the last occurrence of the specified character in the string.

→ public int lastIndexOf(int variable)

→ public int lastIndexOf(int variable, int fromIndex)

→ public int lastIndexOf(String substring)

→ public int lastIndexOf(String substring, int fromIndex)

- if not found it will return -1.
- fromIdex → it start searching from backward direction.

```java
class StringClassMethods
{
public static void main(String args[])
{
//charAt(index)
System.out.println("\n 1.CharAt() method");
String s1="Sheetal Verma";
char ch=s1.charAt(0);
System.out.println("Character at 0 th index in 'Sheetal Verma' "+ch);

//length()
System.out.println("\n2.length() method");
int length=s1.length();
System.out.println("Length of Sheetal Verma is : "+length);

//substring(index)
System.out.println("\n3.substring method");
String s2=s1.substring(2);
System.out.println("Substring from index (2 to end) in Sheetal Verma : "+s2);
String s3=s1.substring(0,5);
System.out.println("Substring from index (0 to 4)index Sheetal Verma : "+s3);

//contain()
System.out.println("\n4.contains() methods");
```

StringClassMethods.java

```
23   //contain()
24   System.out.println("\n4.contains() methods");
25   boolean b= s1.contains("sheetal");
26   System.out.println("Is 'Sheetal Verma' contain 'sheetal' : " +b);
27   System.out.println("Is 'Sheetal Verma' contain 'Sheetal' : "+s1.contains("Sheetal"));
28
29   //replace()
30   System.out.println("\n5.replace() methods");
31   String s6="sheetal verma";
32   String s5=s6.replace('e','i');
33   System.out.println("After replacing 'e' with 'i' in 'sheetal verma' "+s5);
34   String s7="My name was Sheetal ";
35   String s8=s7.replace("was","is");
36   System.out.println("After replacing 'was' with 'is' in 'My name was Sheetal '");
37   System.out.println("New string : "+s8);
38
39   //concat()
40   System.out.println("\n6.concat() methods");
41   String Name="Sheetal";  String surname=" Verma"; String father=" Tarachand";
42   String fullname=Name.concat(surname);
43   System.out.println("After concating 'Sheetal' and 'Verma' : "+fullname);
44   String completeName = Name.concat(father).concat(surname);
45   System.out.println("After concating Sheetal and Verma and Tarachand  "+completeName);
```

StringClassMethods.java

```java

    //indexof()
    System.out.println("\n7.indexof() methods");
    String line="Hello everyone sheetal verma Hello";
    //int index=line.indexOf(97);//ascii of 'a'
    int index=line.indexOf('e');
    System.out.println("index of 'e ' in 'Hello everyone sheetal verma hello '"+index);

    int index1=line.indexOf('e',5);
    System.out.println("index of 'e ' in 'Hello everyone sheetal verma hello ' after index 5 is :"+index1);

    int index2=line.indexOf("Hello");
    System.out.println("index of 'Hello' in 'Hello everyone sheetal verma hello '"+index2);

    int index5=line.indexOf("Helloo");
    System.out.println("index of 'Helloo' in 'Hello everyone sheetal verma hello '"+index5);

    int index3=line.indexOf("hello");
    System.out.println("index of 'hello' in 'Hello everyone sheetal verma hello ' "+index3);

    int index4=line.indexOf("Hello",20);
    System.out.println("index of 'Hello' in 'Hello everyone sheetal verma hello ' after index 20 is :"+index4);

```

StringClassMethods.java

```java
//toupperCase
System.out.println("\n8.toUpperCase() methods");
String ss="Sheetal VERMA";
System.out.println("Sheetal VERMA in upper case :"+ss.toUpperCase());
System.out.println("Sheetal VERMA in lower case :"+ss.toLowerCase());
//trim()
System.out.println("\n9.trim() methods");
String t="  Hello Sheetal Verma  ";
System.out.println("Before trim() method");
System.out.println(t);
System.out.println("After trim() method");
System.out.println(t.trim());

//valueOf()
System.out.println("\n10.valueOf() methods");
int i=20,j=40;
System.out.println("When i is integer i+j=60");
System.out.println(i+j);
System.out.println("when we convert i in string i+j=2040");
System.out.println(String.valueOf(i)+j);//now i become string
System.out.println("Array to string");
char arr[]={'j','a','v','a'};
System.out.println("Array {'j','a','v','a'} as 'java'");
System.out.println(String.valueOf(arr));
```

StringClassMethods.java

```java
//codePointAt()
System.out.println("\n11.codePointAt() method");
String s24= "Sheetal";
int result = s24.codePointAt(0);
System.out.println("Unicode value at 0 :"+result);

//endsWith()
System.out.println("\n12.endswith()method");
String s15= "sheetal";
System.out.println("Is 'sheetal' ends with 'l' : "+s15.endsWith("l"));
System.out.println("Is 'sheetal' ends with 'le': "+s15.endsWith("le"));
System.out.println("Is 'sheetal' ends with 'sheetal' : "+s15.endsWith("sheetal"));

//compareTo()
System.out.println("\n13.compareTo() method");
String s16="sheetal";
String s17="sheetal";
String s18="Sheetal";
String s19="";   String s20="";
System.out.println("Comparing sheetal and sheetal :"+s16.compareTo(s17));
System.out.println("Comparing sheetal and Sheetal :"+s16.compareTo(s18));
System.out.println("Comparing sheetal and empty string :"+s16.compareTo(s19));
System.out.println("Comparing Empty Strings "+s19.compareTo(s20));
```

StringClassMethods.java

```java
//compareTo()
System.out.println("\n13.compareTo() method");
String s16="sheetal";
String s17="sheetal";
String s18="Sheetal";
String s19="";   String s20="";
System.out.println("Comparing sheetal and sheetal :"+s16.compareTo(s17));
System.out.println("Comparing sheetal and Sheetal :"+s16.compareTo(s18));
System.out.println("Comparing sheetal and empty string :"+s16.compareTo(s19));
System.out.println("Comparing Empty Strings "+s19.compareTo(s20));

//lastIndexOf
System.out.println("\n14.lastIndexOf() method");
String s22="Java programming";
int ind=s22.lastIndexOf('J');
System.out.println("'J' index from last in Java Programming : "+ind);
//it will search area before 6
int i1=s22.lastIndexOf('m',6);
System.out.println("'m' index before index 6 in java programming : "+i1);

//isEmpty()
System.out.println("\n15.isEmpty() methods");
String s4="";
System.out.println("Is s4='' variable empty :"+s4.isEmpty());
```

StringClassMethods.java

```java

    //intern()
    System.out.println("\n17.intern() methods");
    //Create in Heap
    String s11=new String("sheetal");
    //Create in String Constant Pool
    String s12="sheetal";
    System.out.println("Before intern()method :");
    //false because s13 point to "sheetal" of heap and s12 is of scp
    String s14=s11;
    System.out.println("s14='sheetal' point to heap 'sheetal' and s12 point to scp 'sheetal'");
    System.out.println("So s14==s12 is :"+s14==s12);
    System.out.println("After intern()method :");
    //true because s13 and s12 both points to "sheetal of scp"
    String s13=s11.intern();
    System.out.println("s13='sheetal' and s12='sheetal' both point to heap 'sheetal'");
    System.out.println("So s13==s12 is :"+s13==s12);
    }
    }
```

StringClassMethods.java

```
124   int i1=s22.lastIndexOf('m',6);
125   System.out.println("'m' index before index 6 in java programming : "+i1);
126
127   //isEmpty()
128   System.out.println("\n15.isEmpty() methods");
129   String s4="";
130   System.out.println("Is s4='' variable empty :"+s4.isEmpty());
131
132   //split()
133   System.out.println("\n16.split() methods");
134   String week= "Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday";
135   String[] names=week.split(",");
136   System.out.println("Spliting with commas");
137   for(String element:names) {
138   System.out.println(element);}
139   //split() with limit
140   System.out.println(" spliting with space and limit is 3");
141   String s21="Hello everyone sheetal verma here";
142   for(String w:s21.split(" ",3))
143   {
144   System.out.println(w);
145   }
146
```

```
***  OUTPUT   ***

 1.CharAt() method
Character at 0 th index in 'Sheetal Verma' S

2.length() method
Length of Sheetal Verma is : 13

3.substring method
Substring from index (2 to end) in Sheetal Verma : eetal Verma
Substring from index (0 to 4)index Sheetal Verma : Sheet

4.contains() methods
Is 'Sheetal Verma' contain 'sheetal' : false
Is 'Sheetal Verma' contain 'Sheetal' : true

5.replace() methods
After replacing 'e' with 'i' in 'sheetal verma' shiital virma
After replacing 'was' with 'is' in 'My name was Sheetal '
New string : My name is Sheetal

6.concat() methods
After concating 'Sheetal' and 'Verma' : Sheetal Verma
After concating Sheetal and Verma and Tarachand  Sheetal Tarachand Verma

7.indexof() methods
index of 'e ' in 'Hello everyone sheetal verma hello '1
```

```
7.indexof() methods
index of 'e ' in 'Hello everyone sheetal verma hello '1
index of 'e ' in 'Hello everyone sheetal verma hello ' after index 5 is :6
index of 'Hello' in 'Hello everyone sheetal verma hello '0
index of 'Helloo' in 'Hello everyone sheetal verma hello '-1
index of 'hello' in 'Hello everyone sheetal verma hello ' -1
index of 'Hello' in 'Hello everyone sheetal verma hello ' after index 20 is :29

8.toUpperCase() methods
Sheetal VERMA in upper case :SHEETAL VERMA
Sheetal VERMA in lower case :sheetal verma

9.trim() methods
Before trim() method
  Hello Sheetal Verma
After trim() method
Hello Sheetal Verma

10.valueOf() methods
When i is integer i+j=60
60
when we convert i in string i+j=2040
2040
Array to string
Array {'j','a','v','a'} as 'java'
java
```

```
11.codePointAt() method
Unicode value at 0 :83

12.endswith()method
Is 'sheetal' ends with 'l' : true
Is 'sheetal' ends with 'le': false
Is 'sheetal' ends with 'sheetal' : true

13.compareTo() method
Comparing sheetal and sheetal :0
Comparing sheetal and Sheetal :32
Comparing sheetal and empty string :7
Comparing Empty Strings 0

14.lastIndexOf() method
'J' index from last in Java Programming : 0
'm' index before index 6 in java programming : -1

15.isEmpty() methods
Is s4='' variable empty :true

16.split() methods
Spliting with commas
Monday
Tuesday
Wednesday
```

```
'm' index before index 6 in java programming : -1

15.isEmpty() methods
Is s4='' variable empty :true

16.split() methods
Spliting with commas
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
 spliting with space and limit is 3
Hello
everyone
sheetal verma here

17.intern() methods
Before intern()method :
s14='sheetal' point to heap 'sheetal' and s12 point to scp 'sheetal'
false
After intern()method :
s13='sheetal' and s12='sheetal' both point to heap 'sheetal'
false
PS D:\JAVA>
```

Constructors of String class

1. String()
String() is a default constructor. It is use to create a empty string. It is also called as empty constructor

```
String variable = new String();
```

2. String(String variable)
It will create a string object in the Heap area using new keyword. And if we pass another string variable it will create copy of that variable in heap.

```
String variable = new String("literal");
or
String variable = new String(String variable1);
```

3. String(byte[] byteArray)
It will create a string from the byte array by decoding the given array of bytes that means decoding Ascii values into the characters. It is use System's default encoding character set.

```
String variable = new String(byteArray)
or
String variable = new String(byte arrayName)
```

4. String(byte[] byteArray, String charsetName)
It will convert the byte array into string using specified character encoding. If encoding is not supported. UnsupportedEncoding Exception is thrown.

```
import java.io.UnsupportedEncodingException;
public class constructor
{
public static void main(String) arg[]) throws UnsupportedEncodingEx
{
byte array[] = { 96, 97, 98, 99 };
String s = new String(array, "UTF-8");
S.o.f (s);
}}
```

→ "US-ASCII"
→ "UTF-8"

output → 'abc

5. String (byte array[], Charset c)
→ This constructor decodes an array of bytes into a string using the specified charset.

→ We can use charset class from 'java.nio.charset' package to encode and decode characters to and from byte array.

6. String (byte[] Name, int startIndex, int length)
Construct a new string from the bytes array depending on the start_index and length means number of characters from starting location.

7. String (byte[] Name, int start, int length, String charsetName)

8. String (char value[])
This will create a string using char sequence given.

9. String (char value[], int offset, int count)

10. String (int array[], int offset, int count)
Create a string from an array of Unicode code points as input. If any of code point are incorrect

it raises the IllegalArgumentException exception.

11. String(StringBuffer variable)

This will creates a string with the same characters sequence as the contents of the specified stringBuffer. class variable.

12. String(StringBuilder buffer)

```
StringBuilder s = new StringBuilder("Sheetal");
String s1 = new String(s);
S.O.P(s1);
```

Output → Sheetal

Extend

01/03/2024

DATE
PAGE

String Buffer

→ Why we need string buffer class.

String buffer class is used to create a mutable string object that means we need String Buffer class when we wants to create string object whose value may change frequently. We can modify the string without creating a new string object each time. This will make stringbuffer class memory efficient.

→ Thread safe operations

In multithreaded environments, "StringBuffer" provides synchronized methods, making it suitable for scenarios where multiple threads might access or modify the same string concurrently.

→ Compatibility

String Buffer is part of the Java Platform's core API and has been available since the early versions of Java.

Syntax of String Buffer class

```
public final class StringBuffer extends Object implements
Serializable, CharSequence
{
}
```

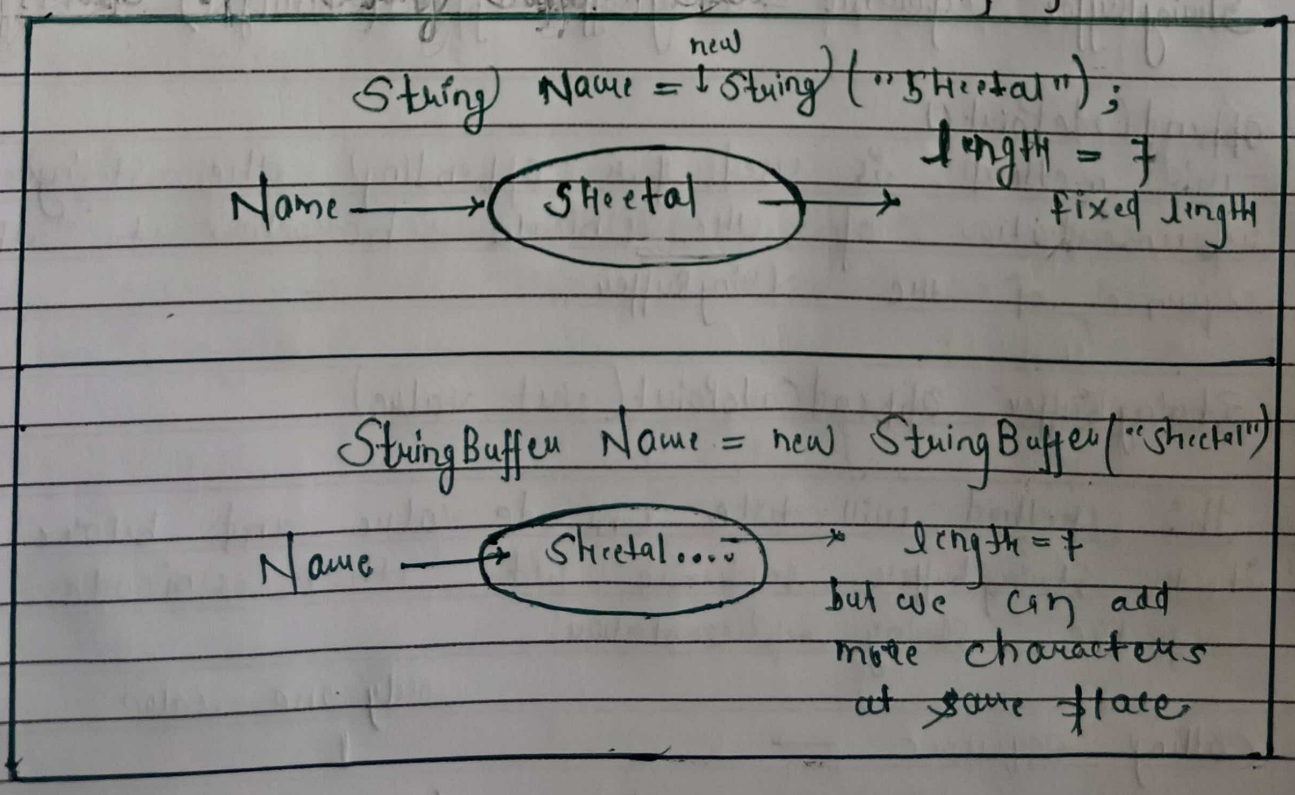
Creating a StringBuffer object

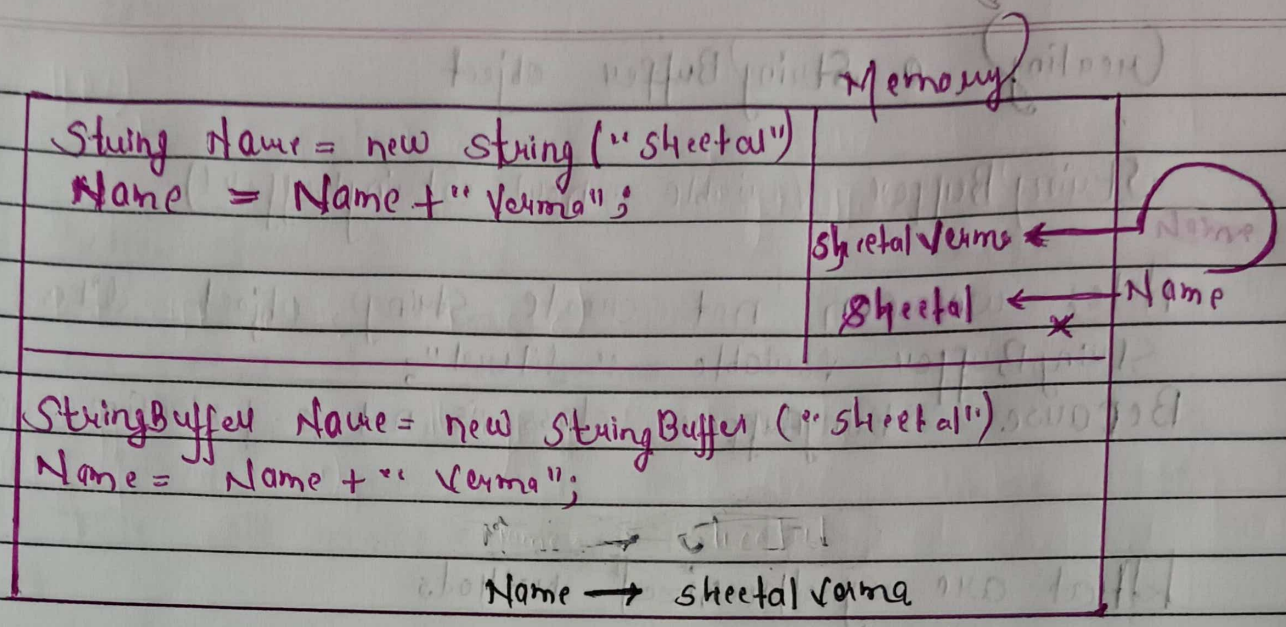
```
StringBuffer variable = new StringBuffer();
```

Note:- we can not create String object like this :-
StringBuffer variable = "literal";
Because StringBuffer class is not a primitive class.

What are synchronized methods

In Java, synchronized methods are methods that are declared with the 'synchronized' keyword. When a method is synchronized, only one thread can execute it at a time. Other thread that wants to execute the synchronized method must wait until the executing thread releases the lock.





Methods of StringBuffer class

1. append()

This method is used for appending the element of a particular datatype at the end of current sequence of the stringBuffer.

StringBuffer reference = StringBuffer.append(datatype)

2. appendCodePoint()

This method is used for appending the string representation of the unicode characters to the sequence of the stringBuffer.

StringBuffer appendCodePoint(int value)

This method will take unicode value and before appending it to StringBuffer sequence it that unicode into respective string representation.

Calling sequence → only one value

stringBuffer.appendCodePoint(unicodeValue)

3. capacity()

The capacity method() of StringBuffer class returns the current capacity of the buffer. The default capacity is 16 bytes. If the number of characters increases from its current capacity, it increases the capacity by (old capacity *2)+2 or (old capacity +1) *2.

Syntax
int capacity()

Calling syntax:
StringBuffer.capacity();

4. charAt()
This method is used to get the character present at given index. It returns the char value. If the index value is out of range, this causes an IndexOutOfException.

Char charAt(int index)

StringBuffer.charAt(index);

5. deleteCharAt()
This method is used to delete the character from given index. The value must be inside (0 to length-1).

StringBuffer deleteCharAt(int index)

StringBuffer.deleteCharAt(index);

6. setCharAt()
This method is used for inserting the character at the given index of the StringBuffer.

Void setCharAt(index, char c)

eg) StringBuffer s = new StringBuffer("shrital");
s.setCharAt(2, 'j');
S.O.p(s);

Output
shjital

7. reverse()
This method is used to reverse the String sequence.

StringBuffer reverse()

String.reverse();

8. length()
This method is used to find the length of string sequence.

int length()
StringBuffer.length();

9. indexOf()
This method is used for finding the first index of the given string in the sequence. If the string is not present in the sequence of the StringBuffer this method will return -1.

int indexof(string)

eg StringBuffer s = new StringBuffer("Hello sheetal");

s.o.p(s.indexof("sheetal")); output → 6

10. lastIndexOf()
This method is used to find the last occurence of given sequence. If the sequence not presence it returns -1, otherwise will return the starting index value.

int lastIndexof(string)

s.o.p(s.lastIndexof("sheetal")); output → 6

11. isEmpty()
Check whether a StringBuffer & reference points to string object or not. If yes it return true otherwise false.

boolean isEmpty()()
StringBuffer.isEmpty()()

12. substring()
This method is used for getting the substring from the StringBuffer sequence.

String substring(int startIndex)
or
String substring(int startIndex, int endIndex)
It will return a String not StringBuffer object.

13. **delete()**

This method is used for deleting the sequence from the buffer. This method takes two parameters, the starting index of the sequence and ending index of the sequence that need to be deleted from the buffer.

StringBuffer reference = StringBuffer.delete(startindex, endingindex+1);

means give one +1 index of the end of deleting sequence

eg StringBuffer s = new StringBuffer("Hey Sheetal");
0 1 2 3 4 5 6 7 8 9 10
s = s.delete(0, 4);
s.o.p(s);

→ Sheetal...

14. **Insert()**

This method of the StringBuffer class is used to insert a string representation of any type into the string buffer object at a specified position.

StringBuffer insert(int index, String str)
StringBuffer insert(int index, int num)
StringBuffer insert(int index, Object obj)

index value must be ≤ length of the stringbuffer object. Otherwise it will show index out of bound error.

eg s = Sheetal;
s = s.insert(0, "hey ");
s.o.p(s);

→ hey Sheetal

→ We can also pass StringBuffer reference variable or string

15. deleteCharAt()

This method will delete the character at specified index.

StringBuffer deleteCharAt(int index)

16. ensureCapacity()

→ This method is used to ensure that the capacity of the 'StringBuffer' is at least equal to the specified minimum capacity. This method will increase the capacity if it is less than specified capacity.

→ This method is useful when you know beforehand that you are going to append a large number of characters to a StringBuffer and wants to avoid memory overheading

public void ensureCapacity(int minCapacity)

StringBuffer.ensureCapacity(value);

StringBufferMethods.java

```java
public static void main(String args[])
{
//append() method
System.out.println("\n1. append()");
StringBuffer s1=new StringBuffer("Sheetal ");
StringBuffer s2=new StringBuffer("Verma");
s1=s1.append(s2);
System.out.println("After appending Verma with Sheetal :");
System.out.println("New String in s1 :"+s1);

//appendCodePoint() method
System.out.println("\n2. appendCodePoint()");
StringBuffer s3=new StringBuffer("abc");
s3=s3.appendCodePoint(100);
System.out.println("Adding 100='d' at the end abc : "+s3);

//capacity() method
System.out.println("\n3. capacity()");
StringBuffer s4=new StringBuffer();
int capacity=s4.capacity();
System.out.println("Default capacity of string s4 :"+capacity);
s4=s4.append("My name is sheetal verma ");
System.out.println("After appending 'My name is sheetal verma':");
System.out.println("New Capacity : "+s4.capacity());
```

StringBufferMethods.java

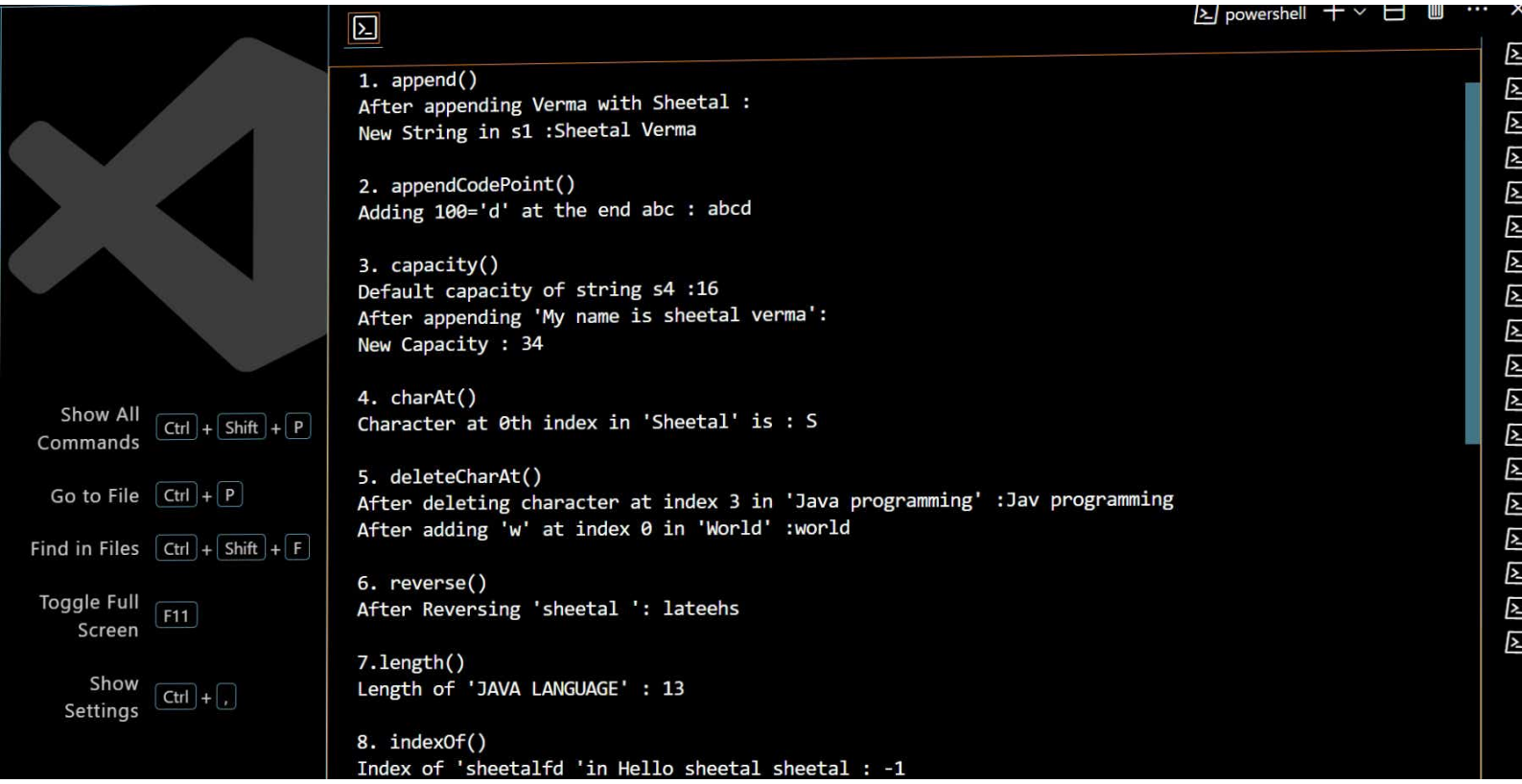
```
28    //charAt()
29    System.out.println("\n4. charAt()");
30    StringBuffer s5=new StringBuffer("Sheetal");
31    char c=s5.charAt(0);
32    System.out.println("Character at 0th index in 'Sheetal' is : "+c);
33
34    //deleteCharAt()
35    System.out.println("\n5. deleteCharAt()");
36    StringBuffer s6=new StringBuffer("Java programming");
37    s6=s6.deleteCharAt(3);
38    System.out.println("After deleting character at index 3 in 'Java programming' :"+s6);
39    StringBuffer s7=new StringBuffer("World");
40    s7.setCharAt(0,'w');
41    System.out.println("After adding 'w' at index 0 in 'World' :"+s7);
42
43    //reverse()
44    System.out.println("\n6. reverse()");
45    StringBuffer s8=new StringBuffer("sheetal");
46    System.out.println("After Reversing 'sheetal ': "+s8.reverse());
47
48    //length()
49    System.out.println("\n7.length()");
50    StringBuffer s9=new StringBuffer("JAVA LANGUAGE");
51    System.out.println("Length of 'JAVA LANGUAGE' : "+s9.length());
```

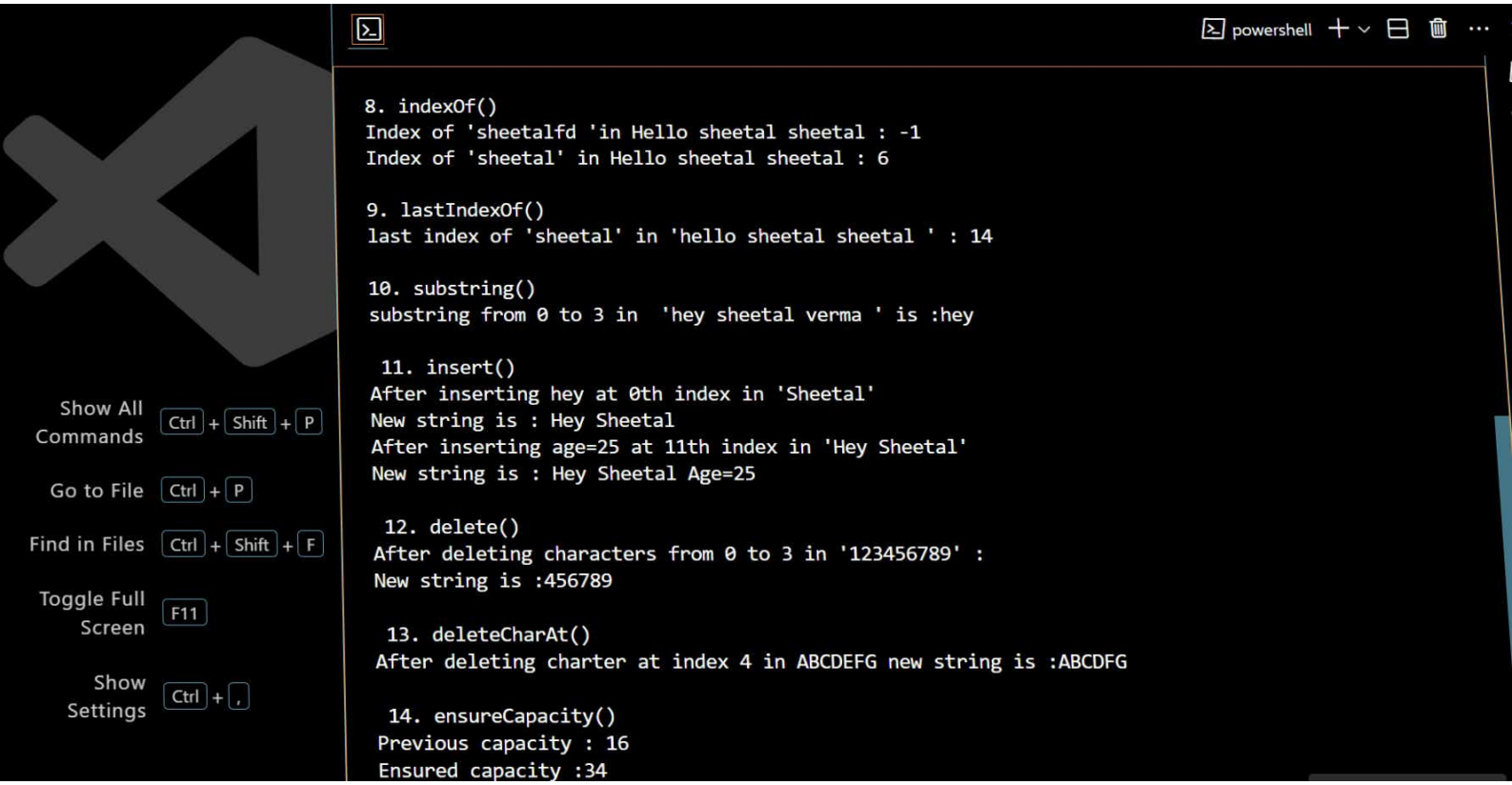
StringBufferMethods.java

```
53    //indexOf()
54    System.out.println("\n8. indexOf()");
55    StringBuffer s10=new StringBuffer("Hello sheetal sheetal");
56    System.out.println("Index of 'sheetalfd 'in Hello sheetal sheetal : "+s10.indexOf("sheetalfd"));
57    System.out.println("Index of 'sheetal' in Hello sheetal sheetal : "+s10.indexOf("sheetal"));
58    //lastIndexOf()
59    System.out.println("\n9. lastIndexOf()");
60    System.out.println("last index of 'sheetal' in 'hello sheetal sheetal ' : "+s10.lastIndexOf("sheetal"));
61
62    //substring ()
63    System.out.println("\n10. substring()");
64    StringBuffer s11=new StringBuffer("hey sheetal verma");
65    String s12=s11.substring(0,3);
66    System.out.println("substring from 0 to 3 in  'hey sheetal verma ' is :"+s12);
67
68    //insert()
69    System.out.println("\n 11. insert()");
70    StringBuffer s16=new  StringBuffer("Sheetal");
71    s16=s16.insert(0,"Hey ");
72    System.out.println("After inserting hey at 0th index in 'Sheetal' ");
73    System.out.println("New string is : "+s16);
74    StringBuffer s17=new StringBuffer(" Age=25");s16=s16.insert(11,s17);
75    System.out.println("After inserting age=25 at 11th index in 'Hey Sheetal' ");
76    System.out.println("New string is : "+s16);
```

StringBufferMethods.java

```
78   //delete()
79   System.out.println("\n 12. delete()");
80   StringBuffer s13=new StringBuffer("123456789");
81   s13=s13.delete(0,3);
82   System.out.println("After deleting characters from 0 to 3 in '123456789' : ");
83   System.out.println("New string is :"+s13);
84
85   //deleteCharAt()
86   System.out.println("\n 13. deleteCharAt()");
87   StringBuffer s14=new StringBuffer("ABCDEFG");
88   s14=s14.deleteCharAt(4);
89   System.out.println("After deleting charter at index 4 in ABCDEFG new string is :"+s14);
90
91   //ensureCapacity()
92   System.out.println("\n 14. ensureCapacity()");
93   StringBuffer s15= new StringBuffer();
94   System.out.println("Previous capacity : "+s15.capacity());
95   s15.ensureCapacity(18);
96   System.out.println("Ensured capacity :"+s15.capacity());
97   }
98   }
```


powershell
1. append()
After appending Verma with Sheetal :
New String in s1 :Sheetal Verma
2. appendCodePoint()
Adding 100='d' at the end abc : abcd
3. capacity()
Default capacity of string s4 :16
After appending 'My name is sheetal verma':
New Capacity : 34
4. charAt()
Character at 0th index in 'Sheetal' is : S
5. deleteCharAt()
After deleting character at index 3 in 'Java programming' :Jav programming
After adding 'w' at index 0 in 'World' :world
6. reverse()
After Reversing 'sheetal ': lateehs
7.length()
Length of 'JAVA LANGUAGE' : 13
8. indexOf()
Index of 'sheetalfd 'in Hello sheetal sheetal : -1
Show All Commands Ctrl + Shift + P
Go to File Ctrl + P
Find in Files Ctrl + Shift + F
Toggle Full Screen F11
Show Settings Ctrl + ,


powershell
8. indexOf()
Index of 'sheetalfd 'in Hello sheetal sheetal : -1
Index of 'sheetal' in Hello sheetal sheetal : 6
9. lastIndexOf()
last index of 'sheetal' in 'hello sheetal sheetal ' : 14
10. substring()
substring from 0 to 3 in 'hey sheetal verma ' is :hey
11. insert()
After inserting hey at 0th index in 'Sheetal'
New string is : Hey Sheetal
After inserting age=25 at 11th index in 'Hey Sheetal'
New string is : Hey Sheetal Age=25
12. delete()
After deleting characters from 0 to 3 in '123456789' :
New string is :456789
13. deleteCharAt()
After deleting charter at index 4 in ABCDEFG new string is :ABCDFG
14. ensureCapacity()
Previous capacity : 16
Ensured capacity :34
Show All Commands Ctrl + Shift + P
Go to File Ctrl + P
Find in Files Ctrl + Shift + F
Toggle Full Screen F11
Show Settings Ctrl + ,

Inherited Methods of StringBuffer class from Object class

1) HashCode()

→ The HashCode() method of the 'StringBuffer' class returns the hash code value for the String Object. The hash code is calculate based on the contents of the String buffer.

→ If two stringbuffer objects are equal according to equals() method, then calling 'HashCode()' on each of them return the some integer value.

→ public int HashCode()

→ It is inherited from Object class method.

2) equals()

→ This method is used to compare two objects of the StringBuffer class. Literally equals() method use the HashCode() method for comparison of two object. If the HashCode values for two objects are equal, this this method returns true otherwise returns false.

:- The HashCode value of every string object is unique even if they have same sequence if we create object using 'new' keyword.

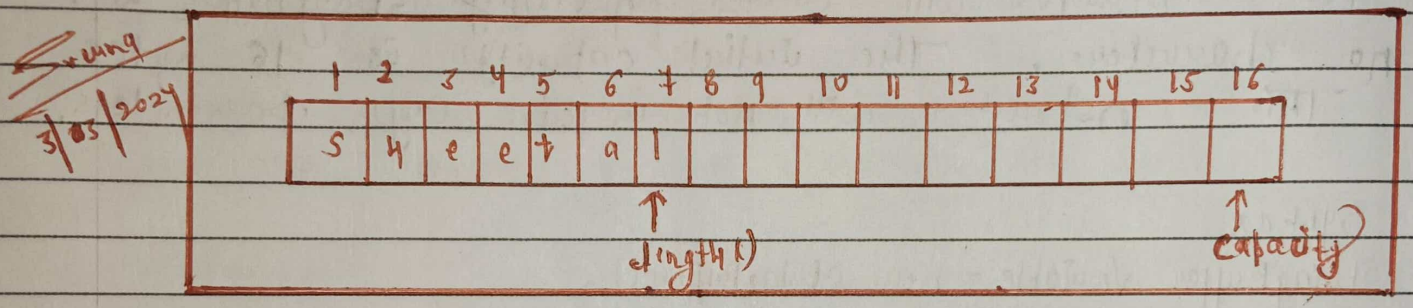
public boolean equals(object obj)

→ It will compare the contents stored into StringBuffer.

3) toString()
This method internally converts the StringBuffer sequence into String format and returns a String.

→ public String toString()

String Variable = StringBuffer.toString();



4) getClass()
The getClass() method in Java is inherited from the 'Object' class. This method returns the runtime class of an object.

→ public final classType getClass()
Or
Class<?>

Class<?> it return type dipendens on the type of class means (Userdefine class, String class etc).

eg StringBuffer S = new StringBuffer("Sheetal");
S.o.p(S.getClass()); → class java.lang.StringBuffer

Userdefineclass ob = new userdefineclass();
S.o.p(ob.getClass()); → class Userdefineclass

→ notify()
→ notifyAll()
→ wait()
→ wait(long timeout)
→ wait(long timeout, int nanos)

Constructors of StringBuffer class

1. StringBuffer()

This constructor will creates an empty stringBuffer with no characters. The initial capacity is 16 bytes. This constructor will not requaire any parameters.

→ Syntax
StringBuffer variable = new stringBuffer();

2. StringBuffer(int capacity)
This Constructor will create an empty StringBuffer with no characters. with the specified initial capacity.

→ StringBuffer variable = new StringBuffer(capacity);

3. StringBuffer(String str)
This constructor will create a string buffer with the specified string. It take one parameter that is string object inside double quotes.

→ StringBuffer variable = new StringBuffer(String);

StringBufferInheritedMethods.java

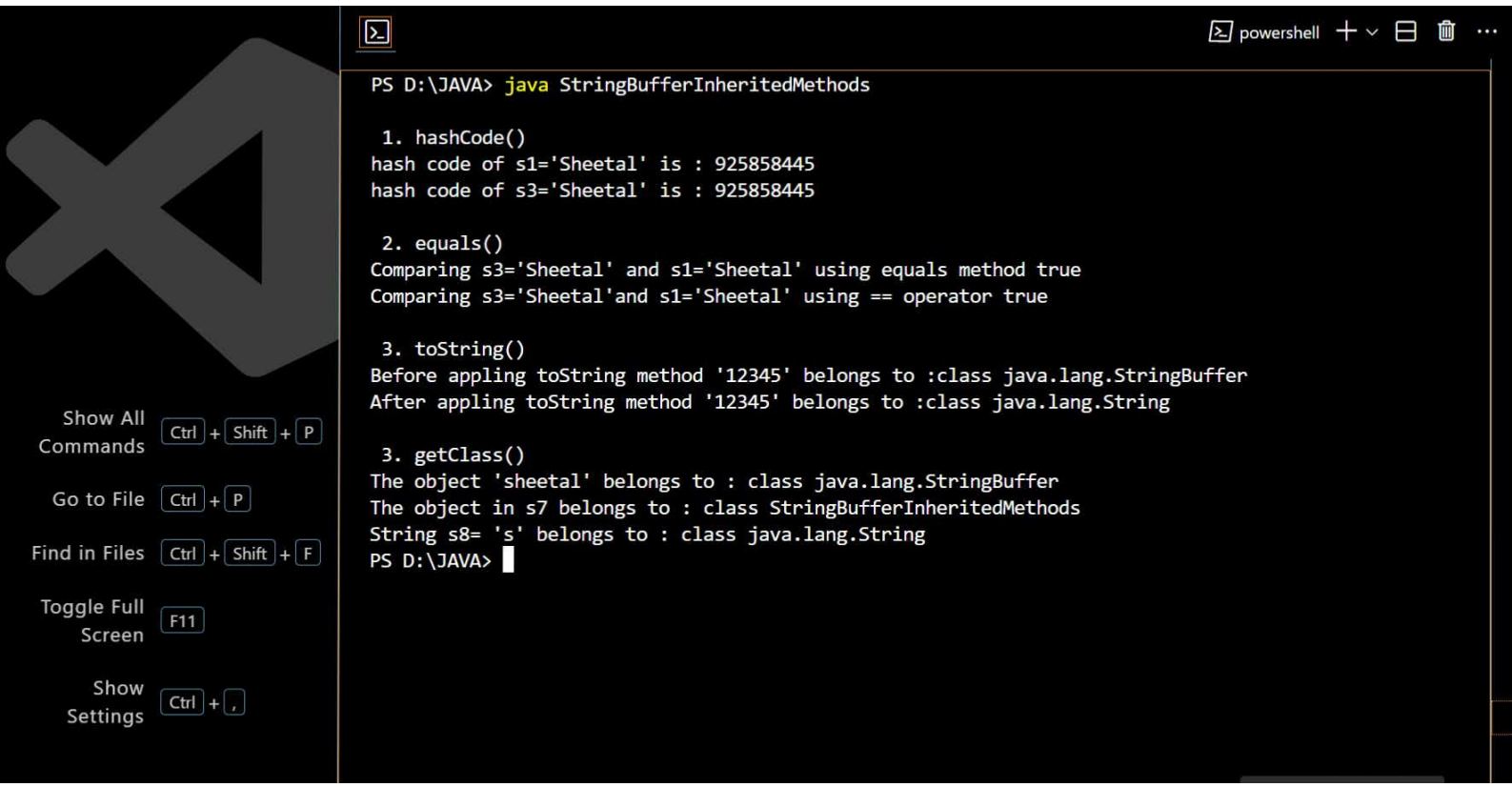
```java
class StringBufferInheritedMethods
{
public static void main(String arg[])
{
//hashCode() method
System.out.println("\n 1. hashCode()");
StringBuffer s1 = new StringBuffer("Sheetal");
StringBuffer s2 = new StringBuffer("Sheetal");
StringBuffer s3=s1;
int s3hash=s3.hashCode();
int s1hash=s1.hashCode();
//"here s1 and s3 holding same object 'Sheetal'"
//"So the hashcode of string inside them is same"
System.out.println("hash code of s1='Sheetal' is : "+s1hash);
System.out.println("hash code of s3='Sheetal' is : "+s3hash);

//equals() method
System.out.println("\n 2. equals()");
System.out.println("Comparing s3='Sheetal' and s1='Sheetal' using equals method "+s3.equals(s1));
System.out.println("Comparing s3='Sheetal'and s1='Sheetal' using == operator "+(s1==s3));
//return false because the hash code value is different
//System.out.println(s1.equals(s2));

//toString method
```

StringBufferInheritedMethods.java

```
21    //return false because the hash code value is different
22    //System.out.println(s1.equals(s2));
23
24    //toString method
25    System.out.println("\n 3. toString()");
26    StringBuffer s4 = new StringBuffer("12345");
27    System.out.println("Before appling toString method '12345' belongs to :"+s4.getClass());
28    String s5=s4.toString();
29    System.out.println("After appling toString method '12345' belongs to :"+s5.getClass());
30
31    //getclass() method
32    System.out.println("\n 3. getClass()");
33    StringBuffer s6=new StringBuffer("sheetal");
34    System.out.println("The object 'sheetal' belongs to : "+s6.getClass());
35    StringBufferInheritedMethods s7=new StringBufferInheritedMethods();
36    System.out.println("The object in s7 belongs to : "+s7.getClass());
37    String s8="s";
38    System.out.println("String s8= 's' belongs to : "+s8.getClass());
39    }
40    }
```


powershell
PS D:\JAVA> java StringBufferInheritedMethods
1. hashCode()
hash code of s1='Sheetal' is : 925858445
hash code of s3='Sheetal' is : 925858445
2. equals()
Comparing s3='Sheetal' and s1='Sheetal' using equals method true
Comparing s3='Sheetal'and s1='Sheetal' using == operator true
3. toString()
Before appling toString method '12345' belongs to :class java.lang.StringBuffer
After appling toString method '12345' belongs to :class java.lang.String
3. getClass()
The object 'sheetal' belongs to : class java.lang.StringBuffer
The object in s7 belongs to : class StringBufferInheritedMethods
String s8= 's' belongs to : class java.lang.String
PS D:\JAVA>
Show All Commands Ctrl + Shift + P
Go to File Ctrl + P
Find in Files Ctrl + Shift + F
Toggle Full Screen F11
Show Settings Ctrl + ,

String Builder

String Builder class in Java is used to create mutable strings. It is available since JDK 1.5.

Why we need StringBuilder class if we already have String and StringBuilder class?

1) Efficiency
Unlike StringBuffer, StringBuilder class is not synchronized. This means is not thread safe, but it can be more efficient in single-threaded environments. If safety is not your concern go for StringBuilder class.

2) Mutability
StringBuilder class allows us to create mutuable String objects

3) Memory Usage
Since StringBuilder class does not use any synchronization mechanism there the use of memory is efficient since no extra memory is required fore locks or mutex or monitors or not additional use of data structures to store inside the memory

4) Preferable in single threaded environments

```
public final class StringBuilder extends Object implements
        Serializable, CharSequence
{

}
```

StringBuilder class syntax

Methods of StringBuilder class

→ append()

public StringBuilder append(String variable)

→ We can append any datatype value.

→ insert()

public StringBuilder insert(int offset, String s)
public StringBuilder insert(int offset, char c)
public StringBuilder insert(int offset, object obj)
:

→ replace()

The replace() method is used to replace a sequence of characters within the StringBuilder with another sequence of characters.

public StringBuilder replace(int start, int end, String str)

start → the starting index of the sequence to be replaced.
end → the ending index of the sequence to be replaced.
str → the string that replace the specified sequence.
↳ if the end index is 3 it will replace till 2.

eg StringBuilder s = new StringBuilder("shital");
012345
s = s.replace(0, 3, "shee");
s.o.p(s)

output
sheetal

→ delete()
→ charAt()
→ reverse()
→ ensureCapacity()
→ capacity()
→ substring(int beginIndex, int endIndex)
→ CodePointAt(int index)
→ CodePointBefore(int index)

→ indexOf(String s)
→ indexof(String s, int fromIndex)
→ lastIndexOf(String s)
→ lastindexof(String s, int index)

→ subSequence(int start, int end)
→ toString()
→ trimToSize()
→ getChars()

→ getchars()
This method is used to copy characters from the String Builder Object into a Character array.

→ getchars(int srcBegin, int srcEnd, char[] dst, int dstBegin)

srcBegin → the starting index from where we need to get character.

srcEnd → the end index inside StringBuilder sequence.

char[] dst → Character array (dst) is name of array.

dstBegin → In character array) from which index you wants to store characters.

eg
```
StringBuilder s = new StringBuilder ("Hellosheetal");
char arr[] = new char[s.length];
s.getChars(5, s.length(), arr, 5);
for(int i=0; i<arr.length; i++)
{
s.o.p("index" + i + "..." + arr[i]);
}
```

output →
index 0
index 1
index 2
index 3
index 4
index 5 s → because we start from 5th index
index 6 h
index 7 e
index 8 e
index 9 t
index 10 a
index 11 l

→ trimToSize()
This method of StringBuilder class is used to trim the capacity of the StringBuilder instance to its current length. for every instance the default capacity is 16 bytes.

eg
```
s.o.p(s.capacity());   → 16 bytes
s.trimToSize();
s.o.p(s.capacity());   → 12 bytes
```

→ this method return nothing (void).

Constructors of StringBuilder class

1. StringBuilder()
This constructor is used for creating a StringBuilder object with no characters in it and initial capacity of 16 characters.

```
StringBuilder reference = new StringBuilder();
```

2. StringBuilder(int capacity)

3. StringBuilder(CharSequence s)
This constructor is used for creating a StringBuilder object with initial characters specified by the character sequence argument. The capacity is 16 + length of charSequence s.

```
eg CharSequence S = "Sheetal";
StringBuffer S1 = new StringBuilder(S);
S.O.P(S1)
```

4. StringBuilder(String s)

```
String name = "Sheetal";
StringBuilder S = new StringBuilder(name);
S.O.P(S);
```

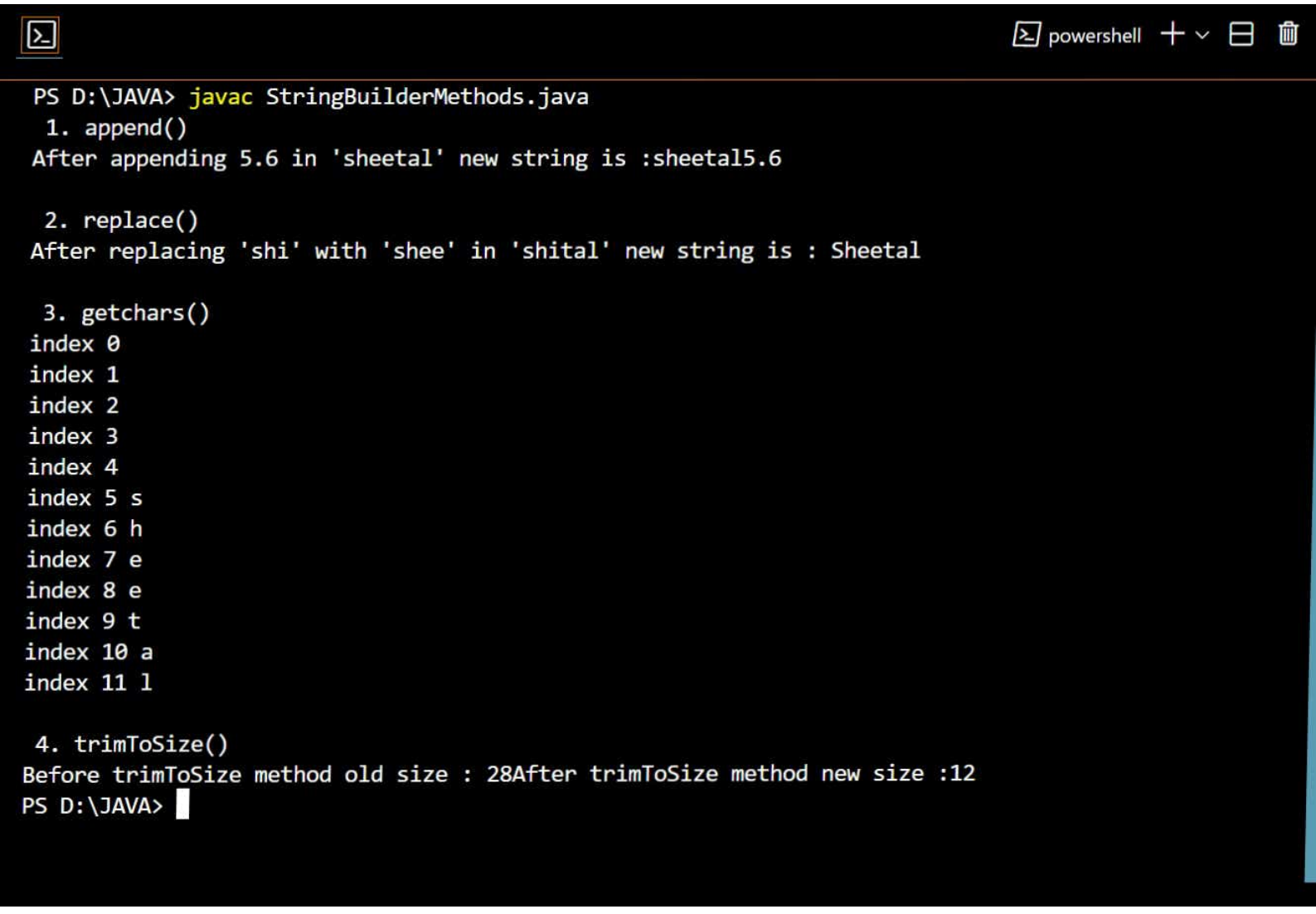
```java
class StringBuilderMethods
{
public static void main(String a[])
{
//append method
System.out.println("\n 1. append()");
StringBuilder s1=new StringBuilder("sheetal");
s1=s1.append(5.6);
System.out.println("After appending 5.6 in 'sheetal' new string is :"+s1);

//replace method
System.out.println("\n 2. replace()");
StringBuilder s2=new StringBuilder("shital");
s2=s2.replace(0,3,"Shee");
System.out.println("After replacing 'shi' with 'shee' in 'shital' new string is : "+s2);

//getchars() method
System.out.println("\n 3. getchars()");
StringBuilder s3= new StringBuilder("Hellosheetal");
char charArray[] = new char[s3.length()];
//here last 5 means i wants to store char from index 5 in array
s3.getChars(5, s3.length(), charArray, 5);
for(int i=0;i<charArray.length;i++)
{
```

StringBuilderMethods.java

```
16
17  //getchars() method
18  System.out.println("\n 3. getchars()");
19  StringBuilder s3= new StringBuilder("Hellosheetal");
20  char charArray[] = new char[s3.length()];
21  //here last 5 means i wants to store char from index 5 in array
22  s3.getChars(5, s3.length(), charArray, 5);
23  for(int i=0;i<charArray.length;i++)
24  {
25  System.out.println("index "+i+" "+charArray[i]);
26  }
27
28  //trimToSize();
29  System.out.println("\n 4. trimToSize()");
30  StringBuilder s4= new StringBuilder("Hellosheetal");
31  System.out.println("Before trimToSize method old size : "+s4.capacity());
32  s4.trimToSize();
33  System.out.println("After trimToSize method new size :"+s4.capacity());
34  }
35  }
```


powershell
PS D:\JAVA> javac StringBuilderMethods.java
1. append()
After appending 5.6 in 'sheetal' new string is :sheetal5.6
2. replace()
After replacing 'shi' with 'shee' in 'shital' new string is : Sheetal
3. getchars()
index 0
index 1
index 2
index 3
index 4
index 5 s
index 6 h
index 7 e
index 8 e
index 9 t
index 10 a
index 11 l
4. trimToSize()
Before trimToSize method old size : 28After trimToSize method new size :12
PS D:\JAVA>

Parameter	String	StringBuffer	StringBuilder
Mutable	Immutable	mutable	mutable
Synchronization	No need of synchronization because string class objects are immutable.	Need synchronization here multiple threads can not change single object. StringBuffer class is synchronized.	non-synchronized.
Added in	The string class is the fundamental class in Java, introduced with release of Java language.	StringBuffer was introduced in Java 1.0.	StringBuilder was introduced in Java 1.5.
Thread safety	Immutable and inherently thread-safe.	Thread safe due to internal synchronization.	Not thread safe. If multiple threads need to modify the same object.
Performance	Because string objects are immutable, every operation that appears to modify a string actually creates a new string object. lead performance issue.	more faster than string class but less than StringBuilder because it required Synchronization.	It is faster than and more efficient than string and StringBuffer because not need to create new object each time and no need of Synchronization.