

String Handling

22/7/15

In this chapter we will learn storing string data in JVM from a program & further performing several operations on this string data. The operations such as comparing, concatenating, replacing, changing case, trimming, retrieving characters, splitting, etc operations.

In this chapter we will learn below things

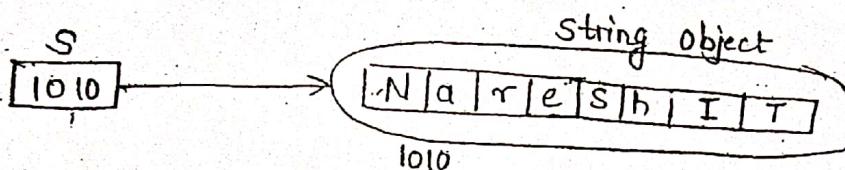
1. What is string?
 2. What is string handling?
 3. Different ways to store string data in JVM.
 4. Difference b/w String, StringBuffer, StringBuilder classes.
 5. Immutable and Mutable objects.
 6. ThreadSafe and NonThreadSafe objects.
 7. StringPooling
 8. String constant Pool Area.
 9. Garbage collection with String Object
 10. String class Constructors and methods
 11. StringBuffer class Constructors and methods.
 12. Programs
 13. Interview & SCJP Questions.
-

Scanned by Aman kumar

What is a String?

→ String is a sequence of characters which is placed in double quotes. It is an instance of a predefined class `java.lang.String`. That means every string literal is an object of String class by default. For e.g. "NareshIT" is an object of String class.

`String s = "NareshIT";`



→ The class `String` is used for storing string data in java program.

What is String Handling?

→ Performing different operations on string data is called String Handling.

→ To perform several operations on string data the reqd. methods are already given by sun in the classes `String`, `StringBuffer`,

→ As a programmer we just need them to access to perform reqd. operations.

For e.g.

- If we want to compare two string having same characters or not, we must call a method 'equals' available in `String` class.

For e.g.

`String s1 = "NareshIT";`

`String s2 = "Naresh IT";`

```
" if (s1.equals(s2)){
```

```
    System.out.println("Hi");
```

```
}
```

```
else{
```

```
    System.out.println("Hello");
```

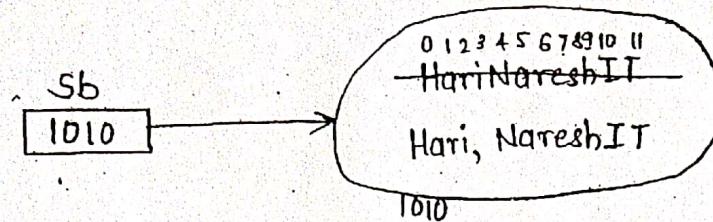
```
}
```

Comparing characters in s1 & s2.

- If we want to insert some characters in the middle of the string we must use a method called 'insert' from `StringBuffer` class.

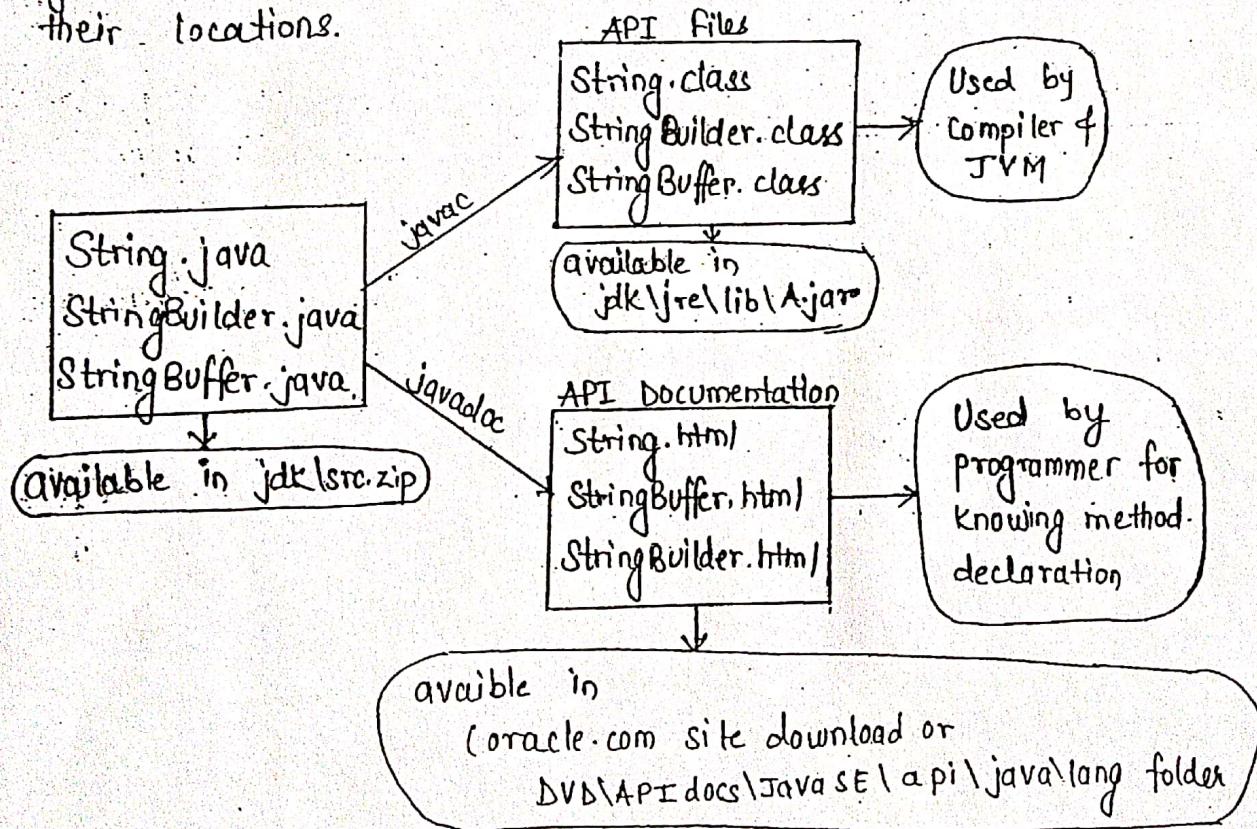
e.g.

```
StringBuffer sb = new StringBuffer("HariNareshIT");
sb.insert(4, ',');
```



How can we find or where can we find method declarations available in `String`, `StringBuffer` & `StringBuilder` classes?

→ We must use API Documentation files of these 3 classes. Below diagram shows above 3 classes java files, .class files, .html files & their locations.



What are the different ways we have to store string data in JVM?

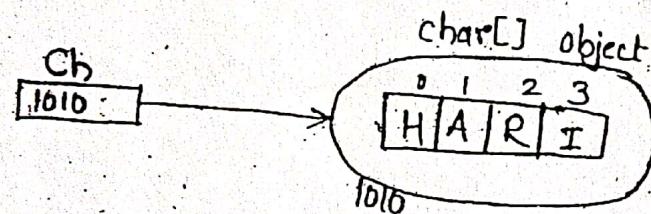
23/7/15

→ We have

1. Char[] object
2. String object
3. StringBuffer object
4. StringBuilder object

→ The actual way of storing string data is using char[] object.
for e.g.

char[] ch = { 'H', 'A', 'R', 'I' };

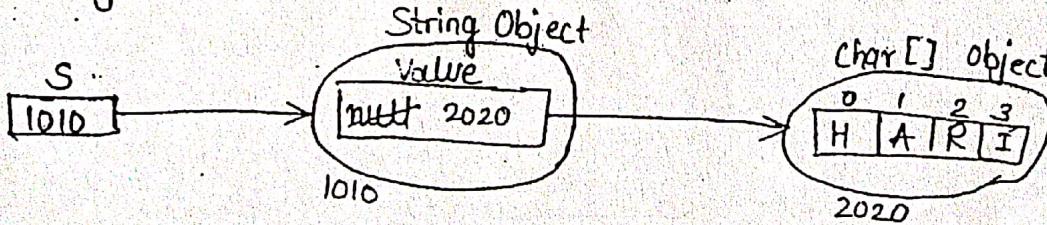


→ Inside String, StringBuffer, StringBuilder classes also char[] object is used for storing string characters.
for e.g.

String class looks like below:

```
public final class String {  
    private final char[] value;  
    ...  
}
```

String s = "HARI";



→ String, StringBuffer, StringBuilder objects has a char[] object for storing the given string characters.

When we have char[] object for storing string characters why String class is given?

→ char[] has two problems

1. Length is fixed

2. Operations reqd. inbuilt methods are not available.

→ To solve above two problems every programmer in projects must define their own class & methods.

→ Since this logic is same for every programmer, sun has decided to define a class with a name string with reqd. no. of methods to operate on char[] object.

→ String class is defined as immutable object that means for the modifications we are doing on string data result will be stored in another null string object, but not in current string object.

→ For e.g.

If you want to concat more characters to an existing string object characters we must use a method called 'concat'. This method internally concat argument string characters to current string characters, stores result in new string object, then returns its reference.

e.g.

```
String s1 = "HARI";
```

```
String s2 = s1.concat("NIT");
```

```
Sopln(s1);
```

```
Sopln(s2);
```

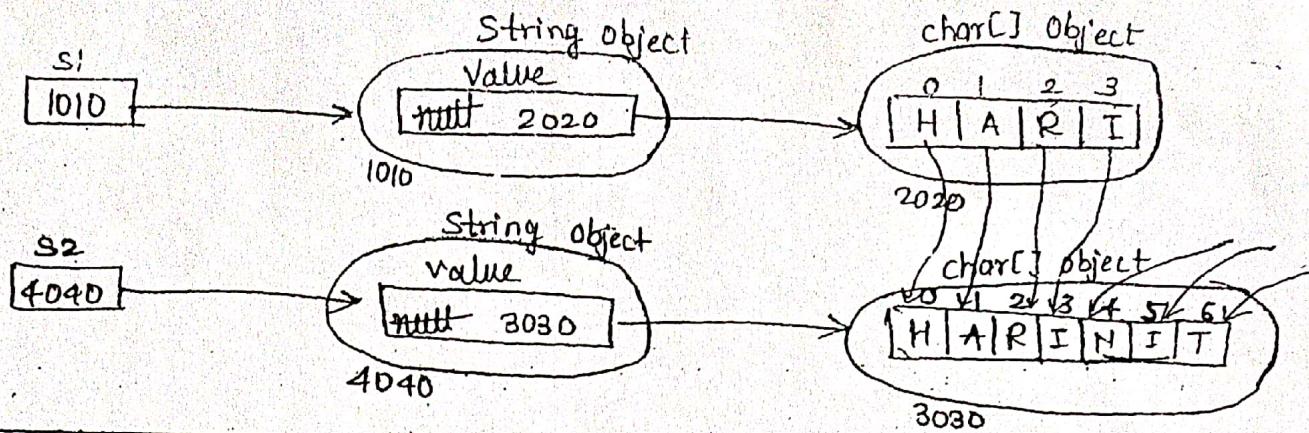
→ O/P HARI

→ O/P HARINIT

Memory diagram:



Memory Diagram:



Why String is given as immutable?

- For using String object as key in Map collection it is created as immutable.
- Once we store an object as key in Map, its data should not be modified. If its data is modified the value mapped to this object cannot be retrieved from this Map because other programmers will try to retrieve value using old data.
- In order to not stored modified modification result in current object, this object should be immutable.

Why StringBuffer class is given when we have string for storing string data?

- Since String is immutable, for every modification on String data new String object will be created. This leads to more memory consume, performance is reduced.
- To solve this problem StringBuffer class is given as a mutable object.
- That means the modifications we are doing on String data in StringBuffer object the result will be stored in the current StringBuffer object, new StringBuffer object will not be created. for e.g.

```
StringBuffer sb1 = new StringBuffer("HARI");
```

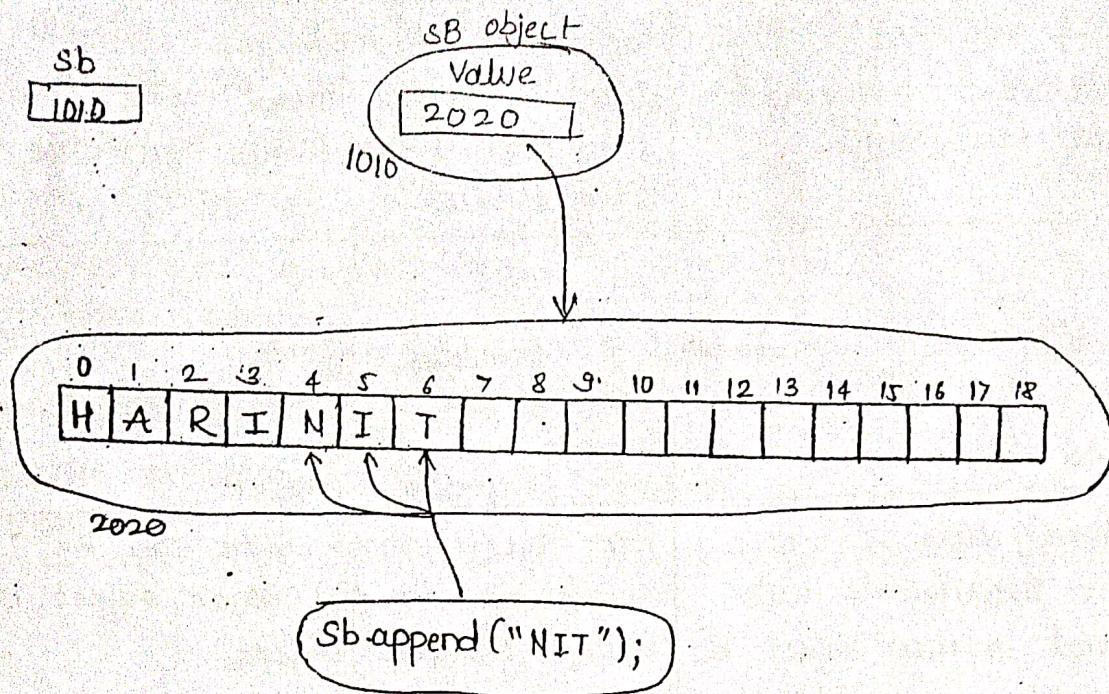
```
StringBuffer sb2 = sb1.append("NIT");
```

```
System.out.println(sb1);
```

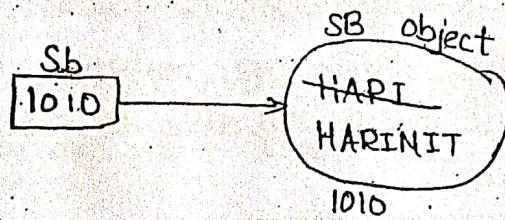
→ %P8: HARINIT

StringBuffer Memory Diagram

25/7/15



Shortcut Memory



Why `StringBuilder` class has define when we have `StringBuffer` class.

→ `StringBuffer` class is given as synchronized or threadsafe object. Due to this we will get less performance in single thread model of method local operations because locking & unlocking object is not necessary.

→ So `StringBuilder` class is given to have non-threadsafe mutable string.

When Should we use `String` & `StringBuffer`.

String

If we don't want to modify string data or if string data modified result want to store in new object, then we must use `String` object to store string data.

StringBuffer

If you want to modify string data in same object we must use `StringBuffer`.

When should we use StringBuffer & StringBuilder

StringBuffer

for storing & modifying string data in multithread environment we must use StringBuffer.

StringBuilder

for storing & modifying String data in single thread model app & in method local we must use StringBuilder.

What is the diff b/w Immutable & Mutable object.

Immutable object

An object whose data is not modifiable or modifiable but result is stored in new object is called Immutable.

for e.g. String, all wrapper classes, java.io.File classes are immutable means once data is stored in this object we cannot modify it.

Mutable object

An object whose data can be modifiable in the same object is called Mutable object.

for e.g. StringBuffer, StringBuilder, all collection objects are mutable. By default every variable is also Mutable.

What is the diff. b/w threadsafe & non-threadsafe objects.

threadsafe

An object which is not available to multiple threads at a time to modify its data concurrently is called threadsafe.

Non-threadsafe

An object which is available to multiple thread at a time to modify its data concurrently is called Non-threadsafe object. If we use Non-threadsafe object in multithread environment we will get wrong results. This object is recommended to use in single thread environment & in method local operations.

OR

An object which is available to multiple threads for modifying its data concurrently, but one thread modified result is not affecting to another thread, means result

If we create class as mutable object class without declaring its method as synchronized then it is non-threadsafe object.

s not stored in same object. We can develop a threadsafe object in two way

1. Declare all mutator methods as synchronized.
2. Create class as immutable Object class.

e.g. StringBuffer, Vector, Hashtable classes or threadsafe object classes.

By default every class is mutable non-threadsafe object classes.

for e.g. StringBuilder is non-threadsafe object.

→

Is the String object is threadsafe object?

→ Yes, it is threadsafe object because of Immutable.

→ All wrapper classes, java.io.file classes are also threadsafe. Because they are also Immutable.

Imp

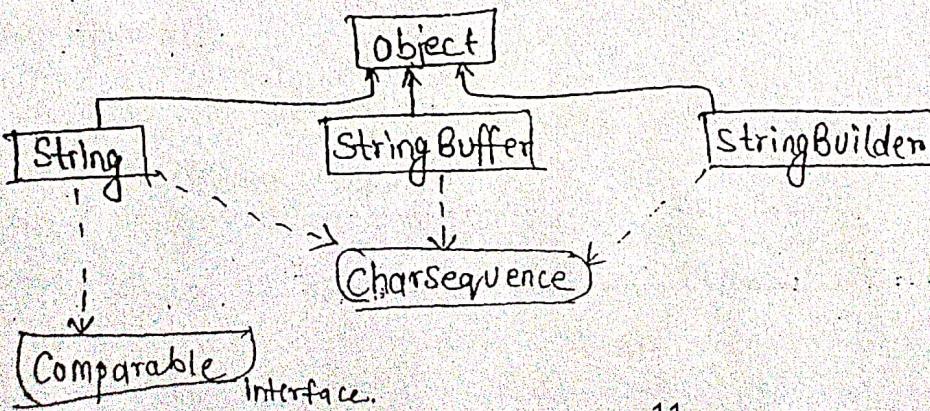
What is String, StringBuffer, StringBuilder?

→ String is immutable sequence of characters. It is a predefined final class.

→ StringBuffer is an mutable, threadsafe sequence of characters. It is also predefined final class.

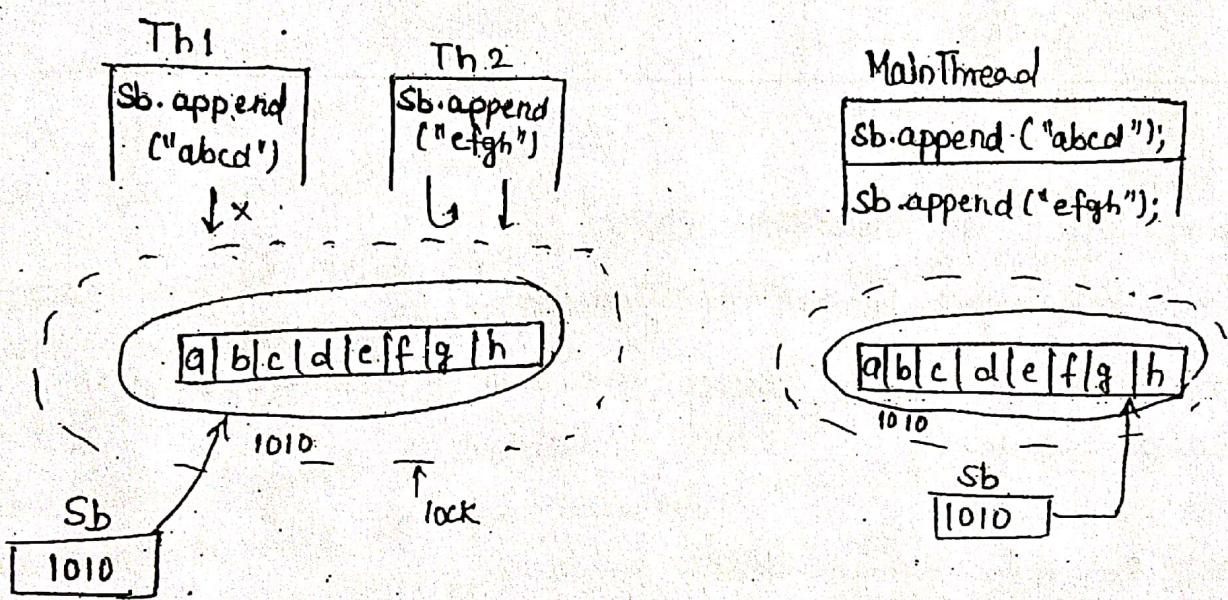
→ StringBuilder is an mutable, non-threadsafe sequence of characters. It is also predefined final class.

What is the relation b/w above 3 classes.



- These 3 objects are siblings, they didn't have inheritance relation because behaviour is different.
- These 3 classes are subclasses of `java.lang.Object` class & `java.lang.CharSequence` interface.
- `String` is also subclass of `Comparable` Interface so we can store its objects inside `Treeset` or `TreeMap` collections.
- We cannot store `StringBuffer` & `StringBuilder` objects in `Treeset` & `TreeMap` collections because they are not Comparable type objects.

Note:-



String Object Creation :-

→ We can create string objects in two ways

1. Using string literal
2. Using new keyword constructor.

for e.g.

```
String s1 = "abc";
```

```
String s2 = new String ("abc");
```

Note: Any data we place in double quotes is a string class object for the double quote JVM will create string object, stores these literal characters inside this object automatically.

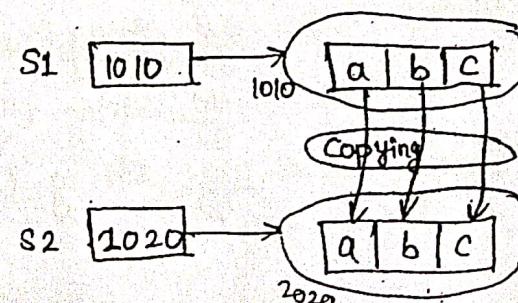
→ The actual way of string object creation approach is string literal approach. Using new keyword & constructor we cannot create the very first object.

→ We must new keyword & constructor approach only for copying operation that means we have already one string has existed, we want to create new string object with the data available in previous string object. Then we must use new keyword constructor approach.

for e.g.

```
String s1 = "abc";
String s2 = new String(s1);
```

assignment
copy



What are the diff b/w these two approaches.

→ We have two differences in creating string object in the above two approaches.

1. No. of objects creation.
2. String Pooling.

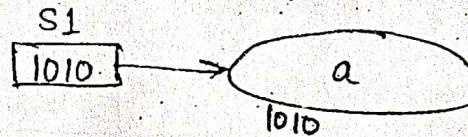
Let us understand first one.

→ In Literal approach only 1 object is created.

→ In new Keyword approach 2 objects are created.
for Eg.

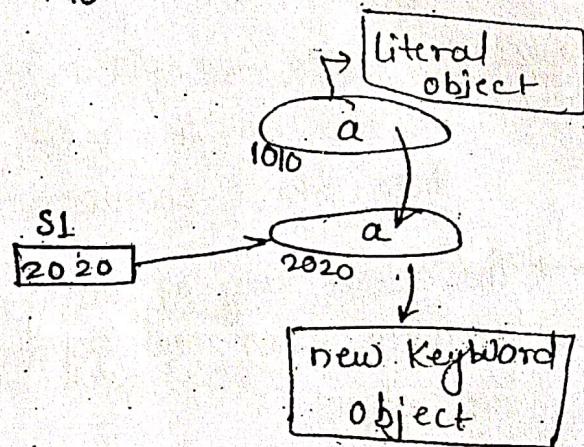
Approach #1:

String s1 = "a";
↓
1 object



Approach #2:

String s1 = new String("a");
↓
2nd object
1st object



find out how many objects are created in below statements.

No. of objects

String s1 = "a"; → 1

String s2 = "a"; → 0

String s3 = new String("a"); → 1

String s4 = new String("a"); → 1

String s5 = new String ("b"); → 2

String s6 = new String ("b"); → 1

6

String Pooling

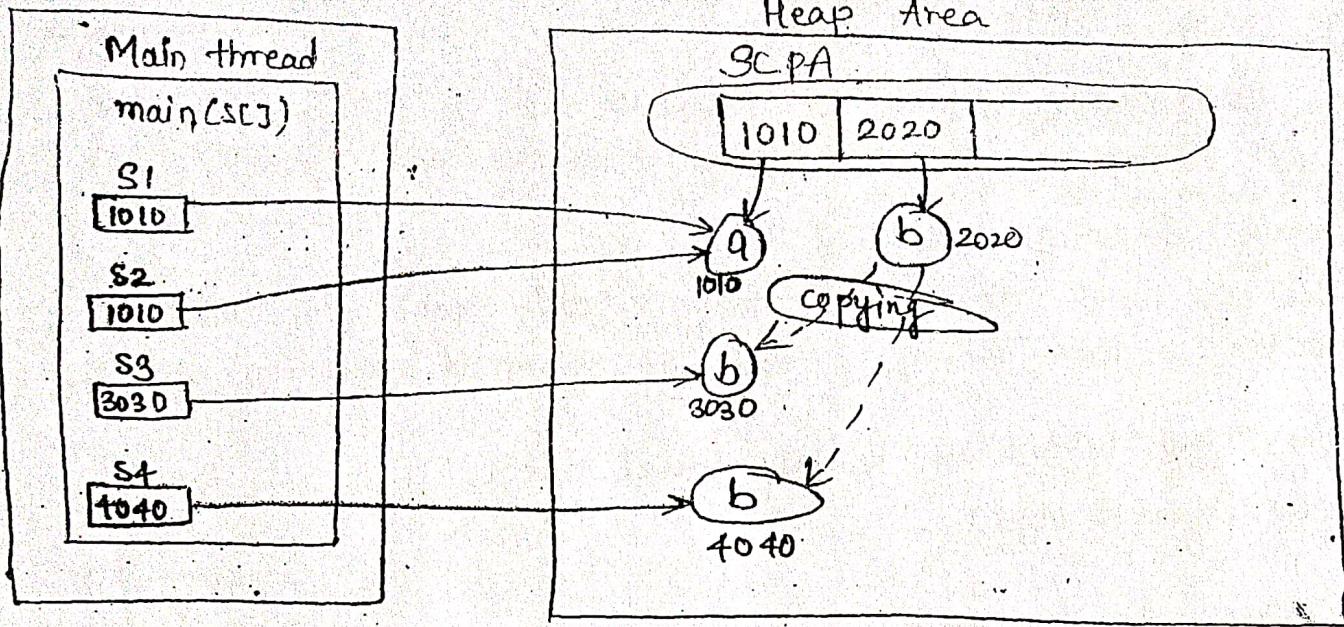
- String Pooling means "grouping String Literal Objects, reusing them in the next line of the code without recreating new objects".
- Advantage of String Pooling:-
 - We can avoid creating new String Literal object with same content so that memory & time will be saved, execution is fast.
- note: String Pooling is applied to only literal object, it is not applied to new keyword object.

String Constant Pooled Area :-

- For implementing String Pooling, inside JVM one special object is used. It is called as String Constant pooled Area (SCPA).
- This Object is nothing but Collection Object Vector/Hashtable
(I cannot prove it, it is simple common sense. That means as many String Literal objects are creating in the program those many String Literal objects should be stored in SCPA without size limitation. The only object that can store other object without size limitation is collection. The collection available in I-O is vector.)
- Below diagram shows String object creation in two ways, storing them in pool pooled Area.

```
String s1 = "a";  
String s2 = "a";  
String s3 = new String("b");  
String s4 = new String("b");  
SopIn(s1==s2); → True  
SopIn(s3==s4); → false
```

JSA



→ SCPA is created one per JVM. Then the String Literal Objects created from different classes loaded in this JVM are created in this same SCPA object this JVM.

→ If a String Literal is repeated in different classes, one object created in pool, shared to all these classes.

Below program will prove SCPA is created 1 per JVM.

A.java

```
class A {
    static String s1 = "a";
}
```

B.java

```
class B {
    static String s2 = "a";
}
```

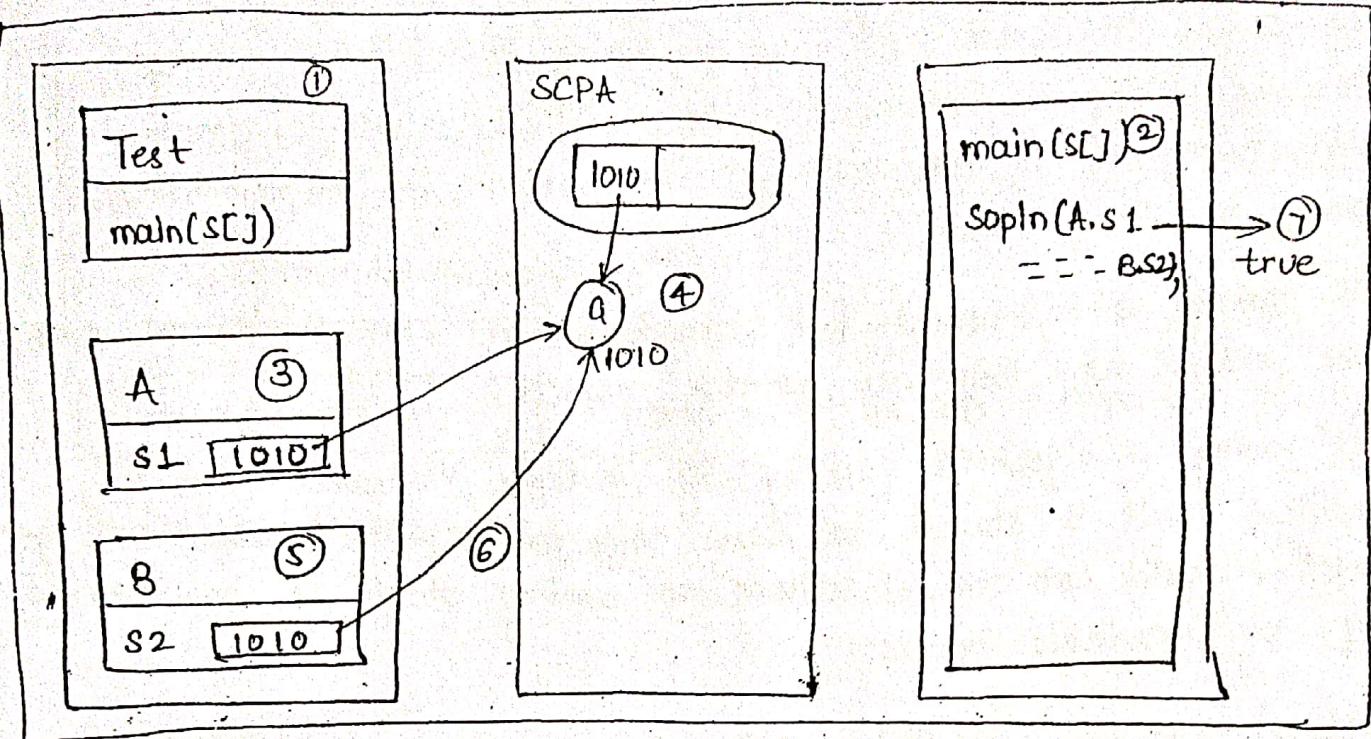
Test.java

```
class Test {
    public static void main(String[] args) {
        System.out.println(A.s1 == B.s2);
    }
}
```

O/P:-true

s1 & s2 variables from A & B classes are pointing to same string literal object creating in pool as shown in below diagram.

java Test



Because of string Pooling String object is pointing by multiple variables
then dnt we have problem when we modify String object using one variable?

→ No, we dnt have problem because string is immutable object, result is stored in new string object then with another referenced variable we will get original value.

for e.g

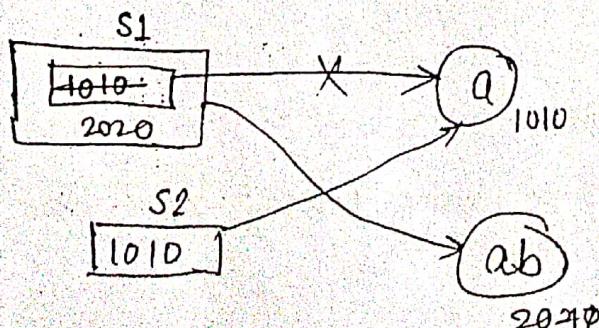
```
String s1 = "a";
```

```
String s2 = "a";
```

```
s1 = s1 + "b";
```

`sopln(s1);` → ab

`sopln(s2);` → a



Why String is Immutable?

→ Because for storing String object as a key in Map Collection

Why pooling is implemented only to String Object?

Because String is immutable, the modified result is stored in different object, so there will not be problem in implementing pooling in String object.

If pooling is implemented for StringBuffer or StringBuilder we will have problem as they are mutable objects.

In pooling, if a object is pointing by multiple variables then if modified result is stored in same object other variables will get modified result not original values. so pooling should be implemented only for immutable objects.

Garbage Collection in String Objects?

Literal String objects are not eligible for Garbage Collection because it is referencing from SCPA always.

String object created using new keyword & constructor it is eligible for garbage collection. when it is unreferenced, because it is not pointing by SCPA.

Below program proves literal object is not eligible for GC. In the below prg we have displayed reference of # String literal object before & after GC. The same value is displayed before & after GC this means string literal is not eligible for GC.

// TestGC.java

```
class TestGC{
```

```
    public static void main(String args){
```

```
        String s1="a";
```

```
        System.out.println(system.identityHashCode(s1));
```

```
        s1=null;
```

→ Unreferencing String Literal object from prg. 18

`System.gc();`

Requesting JVM to start GC.

```

try {
    Thread.sleep(10000);
}
catch(InterruptedException e) {}
```

Passing main thread to allow GC to execute.

`String s2 = "a";`

:: Literal is not GC same the previous object reference is assigned, same reference is displayed.

`sopln(System.identityHashCode(s2));`

O/P:
javac TestGC.java
java TestGC

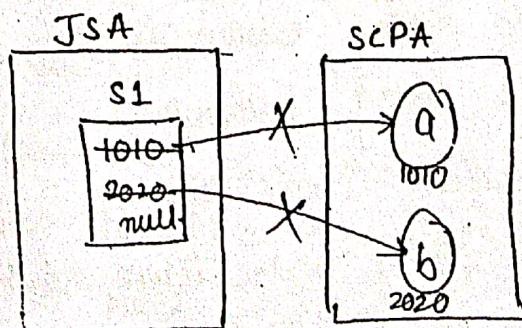
21559264 → Reference Value
21559264

Is String reference variable immutable?

→ No, String referenced variable is mutable that means we can assign another string object reference or null.

for e.g.

```
String s1 = "a";
s2 = "b";
s1 = null;
```

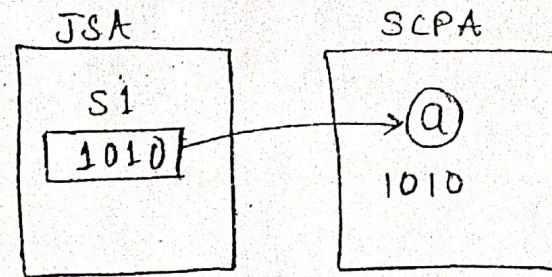


→ How can we make String Variable also Immutable?

→ By declaring it as final

Ex:

```
final String s1 = "a"; ✓  
// s2 = "b"; //ICE  
s2  
// s1 = null; //ICE
```



→ String is Immutable so the modifications we are doing on string data result is not stored in current object for every modification result is stored in new object.

→ String class has several method for verifying string data, retrieving characters from string object, modifying characters in string object.

→ We have four methods for modifying string data these four methods internally creates new String object with a result string data return this new string object reference.

→ These methods will never modify data in current object.
These methods are:

1. for Concatenation

```
public String concat(String s)
```

2. for changing case

```
public String toLowerCase()
```

```
public String toUpperCase()
```

3. for replacing char/substring

replace {
 all occurrences }
 public String replace(char oldchar, char newchar)
 public String replace(CharSequence oldString, CharSequence newString)
 public String replaceAll(String oldString, String newString)

replacing {
 only first occurrence }
 public String replaceFirst(String os, String ns)

4. for removing begin & end spaces
(public String trim()) 20

Q: - What is immutable and mutable object?

- A object which doesn't allow modifications on its data and if it allows modification but result is stored in different new object not in the current object memory & called immutable object.
- A mutable object allows modifications to be stored in current object memory itself. ~~The predefined~~ ~~in Java~~
- The predefined immutable objects are String and all wrapper classes.
- String allows modifications but stores result in new object.
- Wrapper classes doesn't allow modification.
- StringBuffer and StringBuilder are mutable objects.

Q: - Can we develop custom immutable object?

- Yes, we can develop.

Procedure:

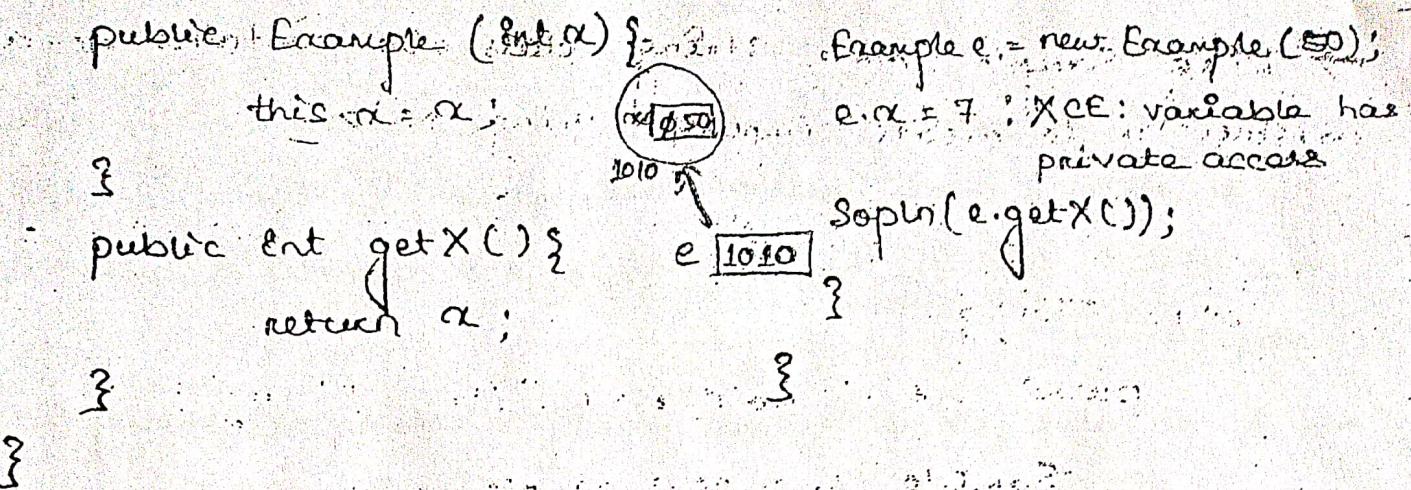
- i) Define a class by declaring all its variable as private.
- ii) Define a parameterized constructor for initializing object variable at least once.
- iii) Don't define setter method, so data will not be modified. If we want to allow modification define setter method but store ~~obj.~~ result in new object.
- iv) Define getter method to read values.

Below class shows developing immutable object that doesn't allow modifications.

class Example {

class Test {

21



Below class shows developing immutable object by allowing modifications but result is stored in new object.

class

```

class Sample {
    private int x;
    public sample () { }
    public sample (int x) {
        this.x = x;
    }
    public sample setX (int x) {
        Sample s = new Sample ();
        s.x = x;
        return s;
    }
    public int getX () {
        return x;
    }
}
```

class Test {
 public static void main (String [] args) {
 Sample s1 = new Sample (50);
 Sample s2 = s1.setX (7);
 Sopln (s1.getX ());
 Sopln (s2.getX ());
 }
}

→ For comparison all wrapper classes are developed by Example class (No setter method)

→ Like Sample class String class also contains no-arg constructor to create "empty" string object.

Q:- What is the output we get from below lines of code?

class Test {

 public static void main (String [] args {

 Sample s₁ = new Sample ("");

 case-1
 s₁.setX (7);

 System.out.println (s₁.getX()); → 5

case-2

 s₂ [3030]

 case-2

 Sample s₂ = s₁.setX (8);

 System.out.println (s₂.getX()); → 5

 System.out.println (s₂.getX()); → 8

 3030

 case-3
 s₁ = s₁.setX (9);

 System.out.println (s₁.getX()); → 9

 }

}

→ In above program 5 methods are executing.

i) main ()

ii) Sample ()

iii) Sample (s)

iv) s.getX ()

v) setX ()

→ So 5 stack frames are created.

s₁.setX (7) call :

→ Here setX (7) method stack frame is created.

→ Parameter 's' variable created with argument 7. 23

→ Local object created under non-modifiable class

- object & created with 'x' variable with default value
- storing argument value in the local object.
- Returning local object reference to main() method and $s_1.set()$ is replaced with the reference 2020.
- Since we are not storing this reference in any destination var the object is lost. Finally conclusion is current object s_1 is not modified.
- So $s_1.get()$ method returns 2020.

Conclusion :-

- If we hold the returned object we can have original value and also modified value.
- In this case we assign returned object reference in the current pointing object referenced variable.
- Now s_1 is pointing to new object and old object & references hence we lost old data.

String handling methods

13/8/15

String

- for storing string object in JVM we must create String type referenced variable.
- String class internally uses char[] object for storing string literal characters.
- So, the characters available in string literal will contain index starts zero.

String Handling

- Below are the operations we can perform on String data & their appropriate methods.

String class methods

- 01.0 find string is empty or not

`public boolean isEmpty()`

- This method returns true if string doesn't have characters
- Returns false if atleast 1 character available in this string object.

- 02.0 finding string length

`public int length()`

- It returns no. of characters available in this string.
- returns zero if no character available.

03. Displaying string characters (content/dat)

`public String toString()`

- This method is overridden in String class for returning current string object's content to display on console.

04. Retrieving hashCode of a String Object

`public int hashCode()`

- This method is overridden in String class for generating hashCode based on its string content.

05. Comparing two string objects for equality

public boolean equals(Object o)

- It also overrides in String class for comparing two string objects with their content.
- It compares characters in both string objects by considering their case.

public boolean equalsIgnoreCase(String s)

- This method specially defined in String class for comparing two string objects with their content, without considering case.

06. Comparing two string objects lexicographically

means comparing two string objects & returning their difference in no.

public int compareTo(String s)

- It is an implementing method of Comparable interface.
- It compares two string content by considering case. returns the a no. representing diff b/w the two string.

public int compareToIgnoreCase(String s)

- It is string class specially defined method for comparing content of two string objects content without considering case.

07. Retrieving a character from the given index

public char charAt(int index)

- This method returns a character from the given index.
- If the given index not found, this method throws StringIndexOutOfBoundsException.

Q8. Finding a index of a given character / substring

```
{ public int indexOf(char ch) }  
{ public int indexOf(String s) }
```

→ These methods return the first occurrence of given of char / string.

```
public int LastIndexOf(char ch)  
public int LastIndexOf(String s)
```

→ These methods will return last occurrence of given char / String.

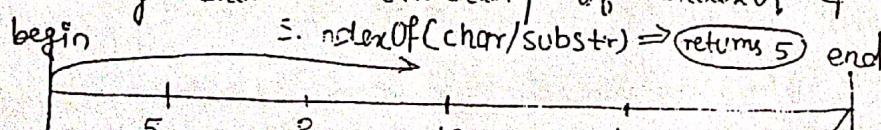
```
public int LastIndexOf(char ch, int fromIndex)  
public int indexOf(char String s, int fromIndex)
```

```
public int LastIndexOf(char ch, int fromIndex)  
public int LastIndexOf(String s, int fromIndex)
```

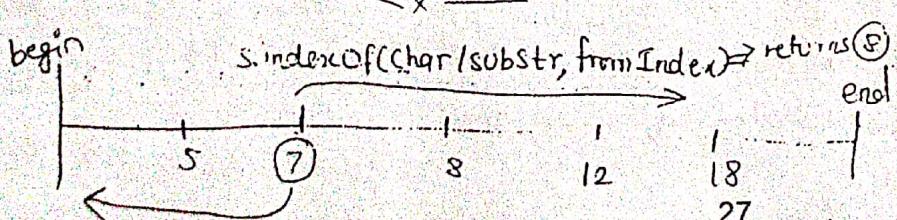
→ These two methods will return the first occurrence of given char or substring after from index inclusive.
the given

→ These two methods will return index of given char or substring before the given from index first occurrence.

Working Functionality of IndexOf & LastIndexOf()



S.lastIndexOf(char/str) ⇒ returns 18



09. Searching character / Substring in ^{This} given string in same sequence.

public boolean contains(String s)

10. Verifying given this string startsWith given substring

public boolean startsWith(String s)

11. Verifying this string endsWith given substring

public boolean endsWith(String s)

12. Retrieving Substring from this string

public String substring(int fromIndex)

public String substring(int fromIndex, int endIndex)

→ In the returning substring fromIndex character is included
endIndex character is not included.

Rule:- The given index should be b/w 0, s.length() - 1 &
also fromIndex should be less than or equals to endIndex

Rule: If fromIndex & endIndex are equal then empty string
will be returned bcoz endIndex character is exclusive.

13. Converting primitive value or object into String

public static String valueOf(XXX values)

int, double, char[], Object, ...

Note:- If you pass null directly to this method it matched to char[]
parameter method, then it internally throws ~~NO~~ NPE.

SopIn(String.valueOf(null)); ✓ NOCE ✗ RF; NPE

If we pass null with other referenced type cast operator it is
matched with Object parameter method, it doesn't throw NPE,
directly null is displayed.

for e.g.

SopIn(String.valueOf((String)null)), $\xrightarrow{\text{U/P}}$ null

SopIn(String.valueOf((Integer)null)); $\xrightarrow{\text{28}}$ null

Eg :-

```
class Student {  
    int sno;  
    public String toString() { (3)  
        return String.valueOf(sno);  
    }  
}
```

(2)

```
class Test {
```

```
    public void main(String[] args) {
```

```
        (1) Student s1 = new Student();
```

```
        System.out.println(s1);
```

```
    }
```

(4)

"101"

(5)

"101"

14. Converting String to char[] (char Array)

```
public char[] toCharArray()
```

15. Converting String to byte Array

```
public byte[] getBytes()
```

Example for charArray.

```
interface A {
```

```
    char[] getFWD(int uid);
```

```
class B implements A {
```

```
    public char[] getFWD(int uid) {
```

```
        String fwd = " ";
```

```
        return (fwd.toCharArray());
```

```
}
```

[NIT] char[] : =

uid = 1

UID - 1

NIT

uid	uname	fwd
1	HK	NIT
2	BB	Acting

→ So far we have learnt methods for either retrieving characters or for comparing characters.

→ Below methods are meant for modifying string data.

16. Concatenating two string characters

public String concat(String s)

17. Changing characters case in of this string.

public String toUpperCase()

public String toLowerCase()

18. Replacing char/substring with new char/substring

public String replace(char oldChar, char newChar)

public String replace(CharSequence oldstr, CharSequence newstr)

public String replaceAll(String oldstr, String newstr)

public String replaceFirst(String oldstr, String newstr)

19. Removing began & end spaces (trim)

public String trim()

String Operations performed by using String class Method :-

→ Sun Microsystems defined multiple methods in String class to perform different operations on string data.

Q:- Find how many characters available in below ~~the~~ String or what is the output from the below program:

class StringLength {

 public static void main (String [] args) {

 String s1 = args [0];

 System.out.println (s1.length());

> javac StringLength.java

> java StringLength TC HariKrishna

1. 4

2. 11 → Complete string is read by args[0] as there is no space.

3. 12 no space in middle

4. 7

> java StringLength TC Hari Krishna

1. 4

2. 11

3. 12

4. 7

Q:- If we use ~~BufferedReader~~ to read String data from keyboard, what is the output in the above two cases?

→ Case - 1 :- O/P - 11 Because ~~br.readLine()~~ reads complete line

→ Case - 2 :- O/P - 12

Q:- If we use ~~Scanner~~.nextLine() to read String data from keyboard, what is the output in the above two cases?

→ Case - 1 :- O/P - 11

Case - 2 :- O/P - 12

Because ~~Scanner~~.nextLine() reads complete line

Q:- If we use ~~Scanner~~.next() to read data from keyboard, what is the output in the above two cases?

→ Case - 1 :- O/P - 11

Case - 2 :- O/P - 4

→ If we call `scn.next()` method again in the same `TRY`, it reads next token(word) in the same string. If all words are read, then it prompts user for entering new string.

Q: - What is the difference between length property and length parentheses method?

→ Length property is used for finding length of the Array Object means no. of elements stored in array.

→ Length parentheses method is used for finding length of the String Object means no. of characters available in the String.

→ We cannot use them vice-versa. It leads to CE: cfs.

Q: Identify CE in the below lines?

String s2 = "abc";

System.out.println(s2.length()); O/P - 3

System.out.println(s2.length()); XCE: cfs

String[] sa = new String[10];

System.out.println(sa.length()); XCE: cfs

System.out.println(sa.length()); O/P - 10

Limitations Of Array Object:-

→ String Limitation - String is immutable. So for every modification of current object, a new string object is created with result and returned.

→ String class has 4 mutator operation methods.

i) `concat()`

ii) `toLowerCase()` .. `toUpperCase()`

→ "So as a result of above methods if current String object data is modified, result is not stored in the current object, a new String Object is created and returned with the result. If the current String object data is not modified, current String object reference is returned.

Q :- Assume we called a mutable method on an immutable object, and then it definitely return a new object with result. Then what is the output we get in below case?

- Case # 1 : Returned object is not assigned to any variable
- Case # 2 : Returned object is assigned to new references variable
- Case # 3 : Returned object is assigned to same current object variable

→ Case # 1 : Result object is lost, old object is still pointing to current object (old).

→ Case # 2 : Result object is held in the program. So we can use original value and new value both.

→ Case # 3 : We can use only new object and old object is lost.

Concatenation :-

→ Placing argument String object characters at end of current String object characters and storing result in new String object and returning its reference is called concatenation.

→ We can perform concatenation operation in 2 ways.

i) using '+' operator

E) Concatenation using concat():

prototype - public String concat (String s)

Write a test program to concat "bbc" with "abc".

class TestConcat {

public static void main (String [] args) {

String s1 = "abc";

String s2 = s1.concat ("bbc");

System.out.println (s1); → o/p - abc

System.out.println (s2); → o/p - abcbbc

}

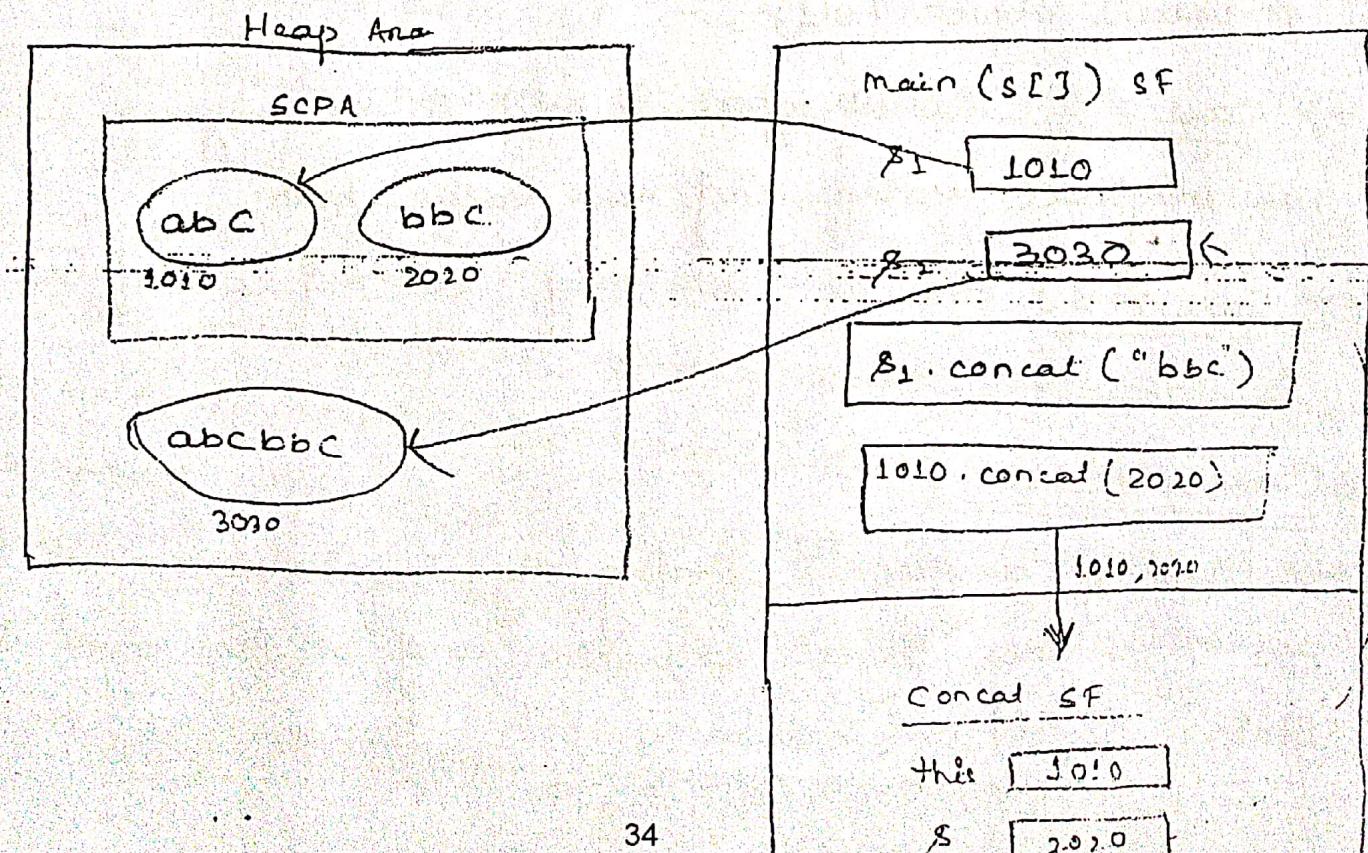
}

* In the above program 3 objects are created.

i) "abc" in p-SCPA .

ii) argument String object "bbc" in SCPA .

iii) Result String object "abcbbc" in head area .



Written Test Questions

String s₃ = "a";

s₃.concat("b");

SopIn(s₃); → a

String s₄ = s₃.concat("c");

SopIn(s₃); → a

SopIn(s₄); → ac

SopIn(s₃ == s₄); // false

String s₅ = s₃.concat("");

SopIn(s₃); → a

SopIn(s₅); → a

SopIn(s₃ == s₅); → // true

s₃ = s₃.concat("d");

SopIn(s₃); → ad

* Total 8 objects are created in the above program.

Here 2 objects created b and ab.

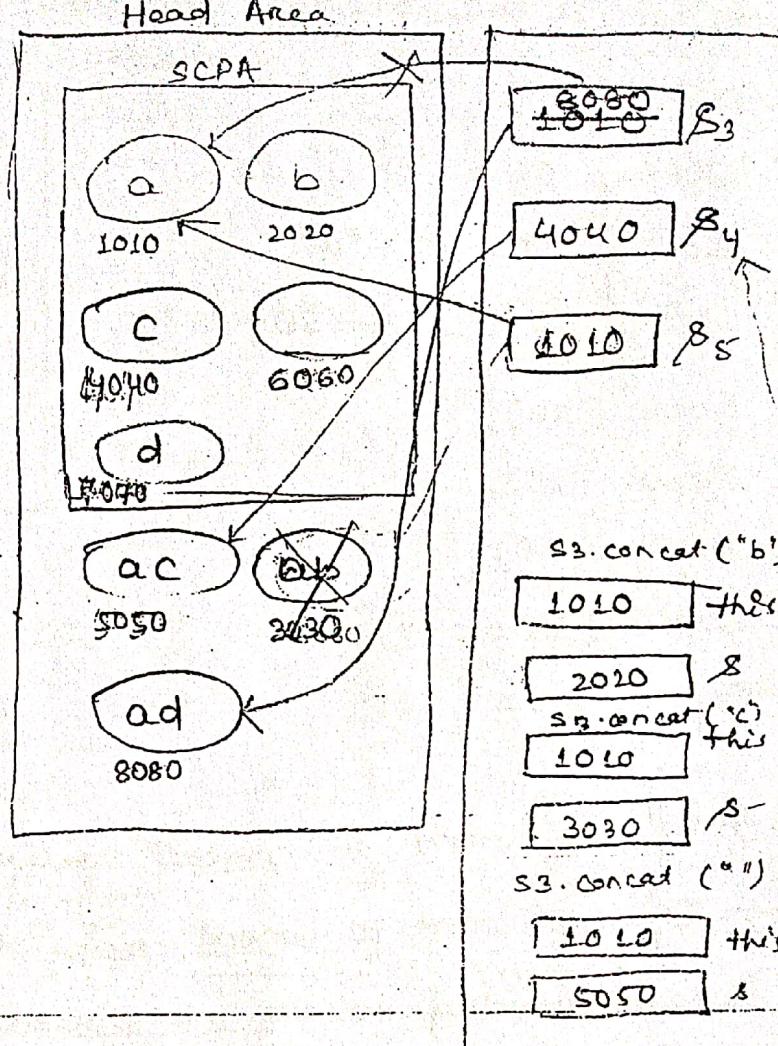
Here 2 objects created c and ac.

Here 1 object created "".

Here 2 objects created d and ad

Correlation with '+' operator:-

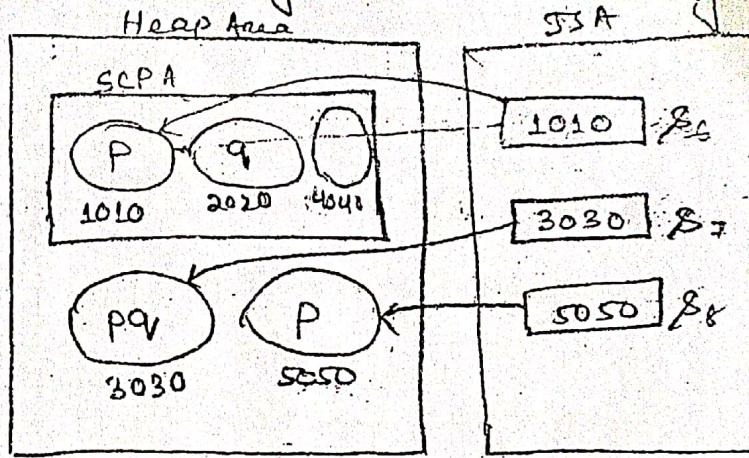
→ Unlike concat() method, '+' operator always return new object irrespective of current object be modified or not. Because we don't have current String object and argument String object in '+' operator to do length calculations.



s ₃ .concat("b")	
1010	this
2020	s
s ₃ .concat("")	this
1010	s-
3030	s-
s ₃ .concat("")	
1010	this
5050	s

What is the output from the below program and how many objects are created?

```
String s6 = "P";  
String s7 = s6 + "q";  
String s8 = s6 + " ";  
Sopn(s6);  
Sopn(s7);  
Sopn(s8);  
Sopn(s6 == s7);  
Sopn(s6 == s8);
```



→ If we want to concat only two strings we must use concat

→ If we want to concat more than two string in single line
is recommended to use '+' operator, using concat() method is not readable.

→ For example:

```
String s9 = s1 + s2 + s3 + s4;
```

```
String s10 = s1.concat(s2).concat(s3).concat(s4);
```

Sopn(s9); → Method call chaining.
Sopn(s10);

→ When we call a method with a method we should search for the called method in the first method return type class. And it is executed with first method returned object data.

→ In concatenation operation using '+' operator if an expression contains literals only then it is evaluated by compiler and replace expression with the result String. Then JVM creates only one String object with the result data 36

→ If an expression contains variable then that expression is calculated by JVM and it creates every string literal as object and then it evaluates.

How many objects are created in the below program?

Class StringLiteralExprTest {

public static void main (String [] args) {

String s1 = "ab"; → 1

String s2 = "bb"; → 1

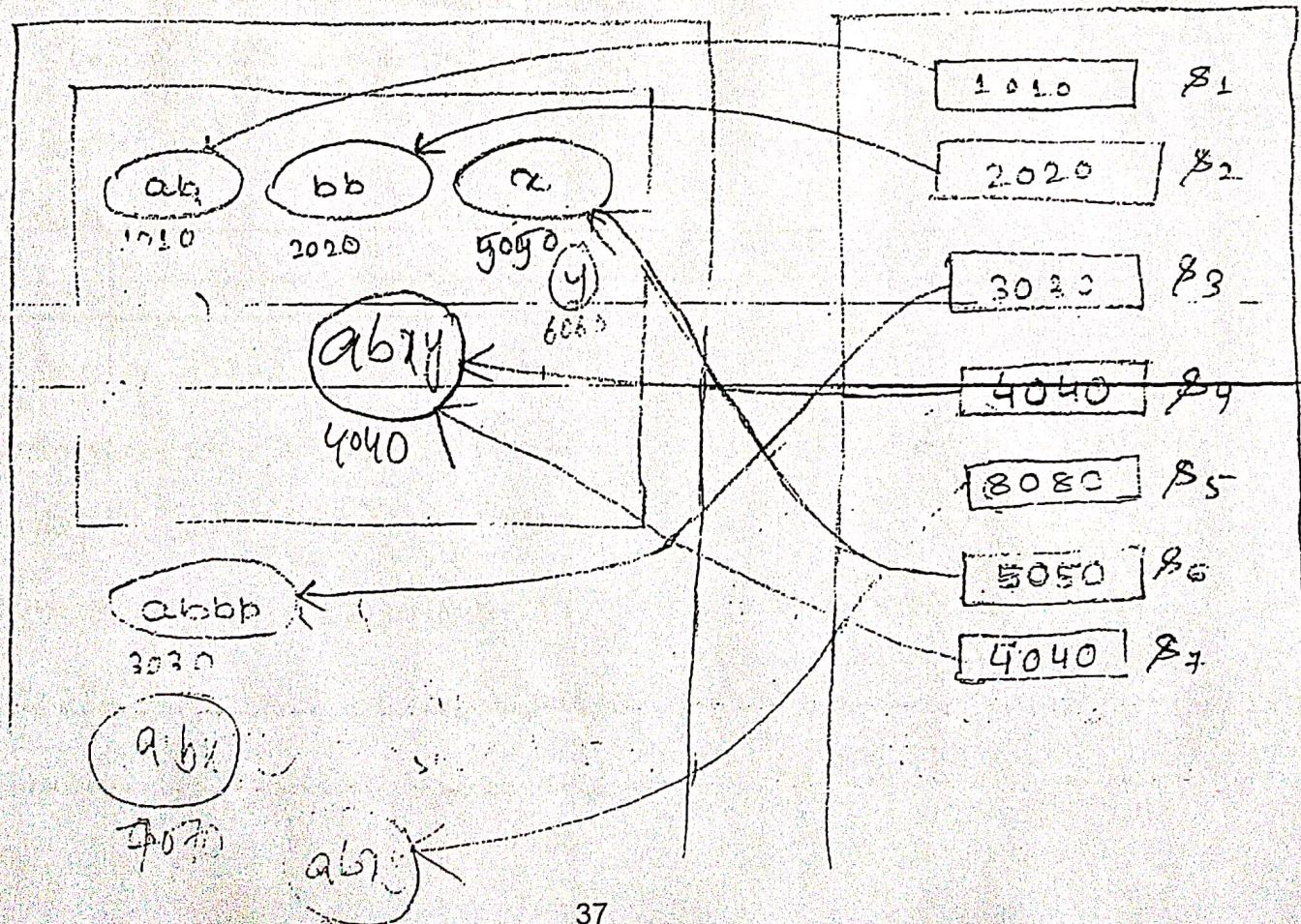
String s3 = s1 + s2; → 1

String s4 = "ab" + "x" + "y"; → 1

String s5 = s1 + x + y; → 3

String s6 = "x"; → 0

String s7 = "abxy"; → 1



Write a program to read first name and last name of the customer and print his fullname on the console with wish "Hi"

Enter fname : Hari

Enter lname : Krishna

O/P - :--: Hi Hari Krishna

Changing case Of the String

// public String toLowerCase()

// public String toUpperCase()

→ Above methods changing the case of alphabets in the current string.

→ If it has only numbers or special characters there is no effect in calling these methods i.e. no change in the current string. As a result of these method calls if at least one character case is changed result String is returned with new object else return same object.

→ Below program shows the change in case of the String

on't change String s1 = "abc";

use in s1 s1. toUpperCase();

returns new Sopln(s1); → abc

String s2 = s1.toLowerCase();

String s3 = s1.toUpperCase();

Sopln();

Sopln(s1); → abc

Sopln(s2); → abc

Sopln(s3); → ABC

Sopln(s1 == s2); // true

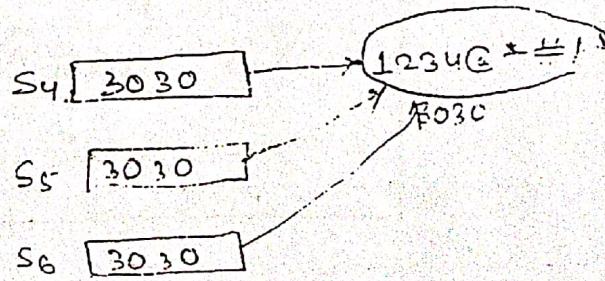
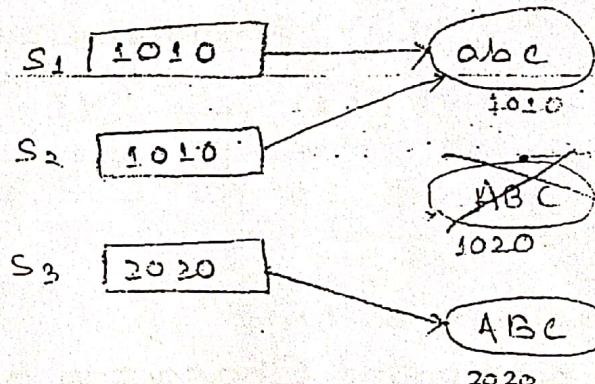
Sopln(s1 == s3); // false

String s4 = "1234@*#!";

String s5 = s4.toLowerCase();

String s6 = s4.toUpperCase();

Sopln();



`Sopn(s5 == s6); // true`

`String s7 = "XYZ";`

`Sopn();`

`Sopn(s7.toLowerCase());`

`Sopn(s7); → XYZ`

`String s8 = s7.toUpperCase();`

`String s9 = s7.toLowerCase();`

`Sopn(s7); → XYZ`

`Sopn(s8); → XYZ`

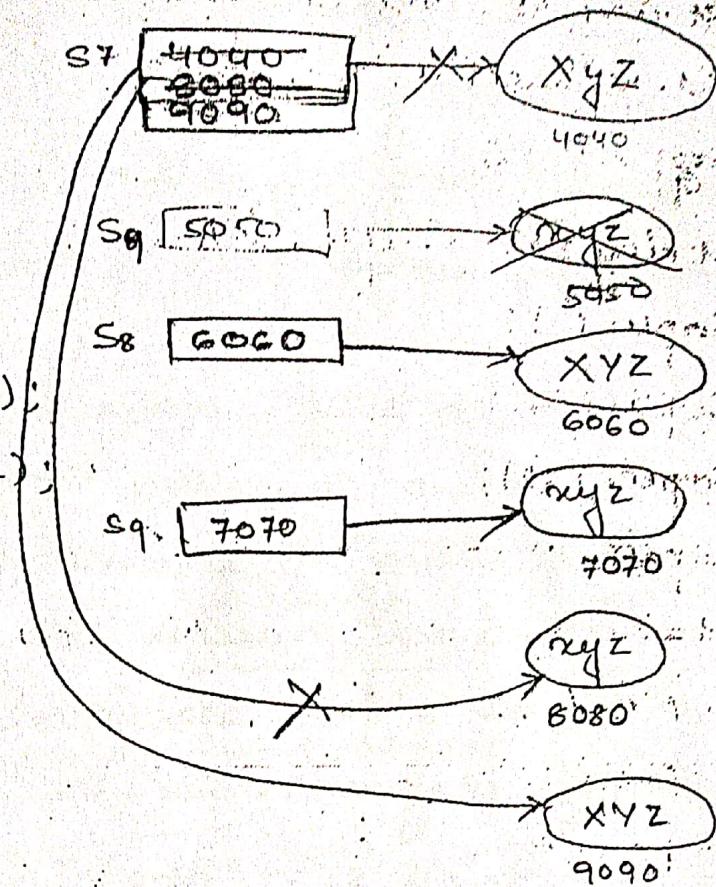
`Sopn(s9); → xyz`

`s7 = s7.toLowerCase();`

`Sopn(s7); → xyz`

`Sopn(s7 = s7.toUpperCase()); → XYZ`

`Sopn(s7); → XYZ`



Here s7 pointing objects characters are changed to UpperCase and stored etc reference in s7 variable and then printed. So in the next line we got same output.

Replacing Character or SubString in Current String :-

→ If current String is modified all below methods returns new String object with result.

Prototype - public String replace(char oldChar, char newChar)
public String replace(CharSequence oldString, CharSequence newStr);
public String replaceAll(String regex, String replacement)
public String replaceFirst(String regex, String replacement)

String s1 = "abc abc abc";

s1.replace('a', 'b'); → (doesn't replace 'a' in s1 pointing object)

Sopn(s1); → abc abc abc

String s2 = s1.replace('A', 'b');

Sopn(); → java Es case sensitive. So a ≠ A

Sopn(s1); → abc abc abc

Sopn(s2); → abc abc abc

Sopn(s1 == s2); // true

String s3 = s1.replace("ab", "cb");

Sopn();

Sopn(s1); → abc abc abc

Sopn(s3); → cbc cbc cbc

Sopn(s1 == s3); // false

String s4 = s3.replaceFirst("cb", "bb");

Sopn();

Sopn(s4); → bbc bbc bbc

Sopn(s3); → cbc cbc cbc

s3 [3030]

cbc cbc cbc

3030

Replace all the occurrences of "ab".

s4 [4040]

bbc cbc cbc

4040

Replace only the first occurrence of "cb".

Q:- Create a program with string object "Java". Replace its characters in below order and print result.

"Java" → Kava → Kaka → KeKa

→ class ReplaceCharacterTest

```
public static void main (String [] args) {
```

```
    String s1 = "Java"
```

```
    String s2 = s1.replace('J', 'K');
```

```
    System.out.println (s2);
```

String s3 = s2.replace('v', 'k');

System.out.println(s3);

String s4 = s3.replaceFirst("a", "o");

System.out.println(s4);

}

}

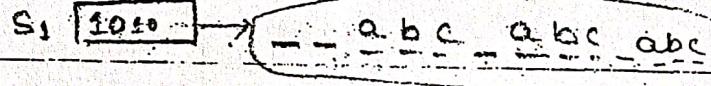
Trim():

→ It is used for removing leading and trailing spaces (beginning and end) spaces).

prototype :- public String trim();

String s1 = " -- abc -- abc -- abc -- ";

Sopn(s1.length()); → 16



String s2 = s1.trim();

Sopn();

Sopn(s1.length()); → 16

Sopn(s2.length()); → 11

String s3 = "abc--abc--";

s3.trim();

Sopn(s3.length()); → 9

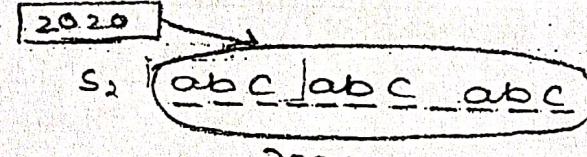
Sopn();

String s4 = s3.trim();

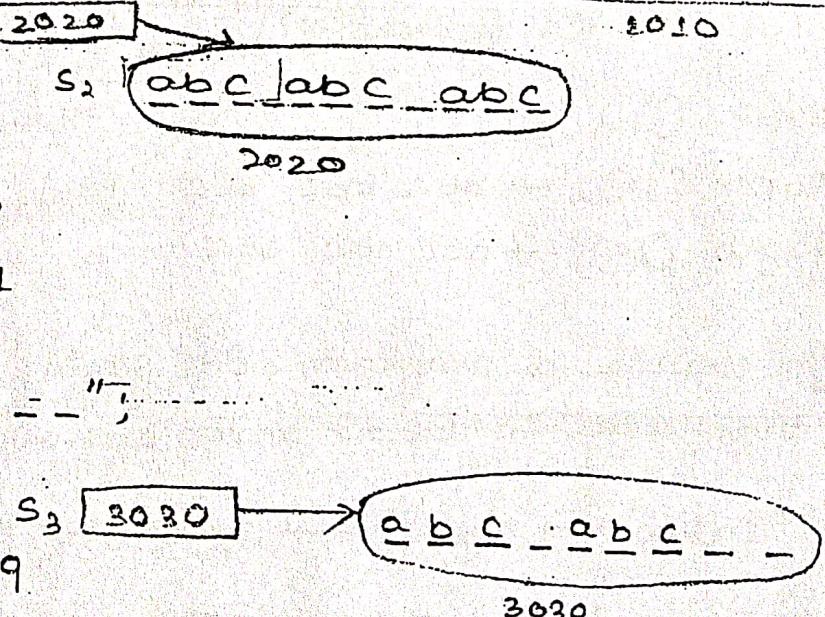
Sopn(s3.length()); → 9

Sopn(s4.length()); → 7

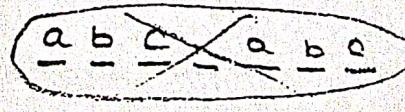
Sopn(s3 == s4); // false



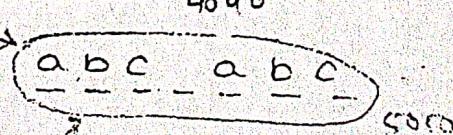
2020



3030



4040



5050

String ss = s4.trim();

Sopln(s4.length()); → 7

Sopln(ss.length()); → 7

Sopln(s4 == ss); // true

⇒ Trim(). removes only begin and end space.

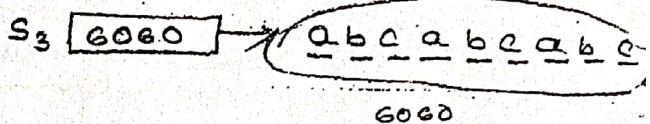
So how can we remove the middle spaces?

⇒ We must use replace() method to remove middle spaces.

String s3 = s1.replace(" ", " ");

String s4 = s1.replace(" ", " ");

Sopln();



Sopln(s1.length()); → 16

Sopln(s3.length()); → 9

Sopln(s4.length()); → 12

Importance of trim() methods:

* In projects often we are reading the data from HTML form text fields, we must read the data by removing starting and trailing spaces. Here user enter unknowingly. And then further the result data is stored in database.

Ex:- Reg.html

first Name	Hari
Last Name	Krishna
User Name	Harikrishna
password	Naresh@t
email	hari.krishna.hk@gmail.com
<input type="button" value="Submit"/>	

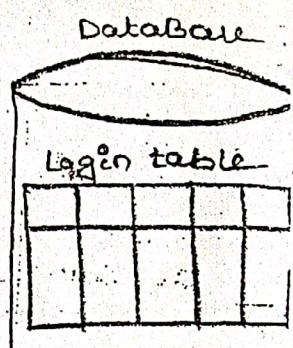
Req

Servlet

```
String fname = req. getParameter("fn").trim();  
String lname = req. getParameter("ln").trim();  
String user = req. getParameter("user").trim();  
String pwd = req. getParameter("pwd").trim();  
String email = req. getParameter("email").trim();
```

text fields name
given in Reg. HTML
files

JDBC

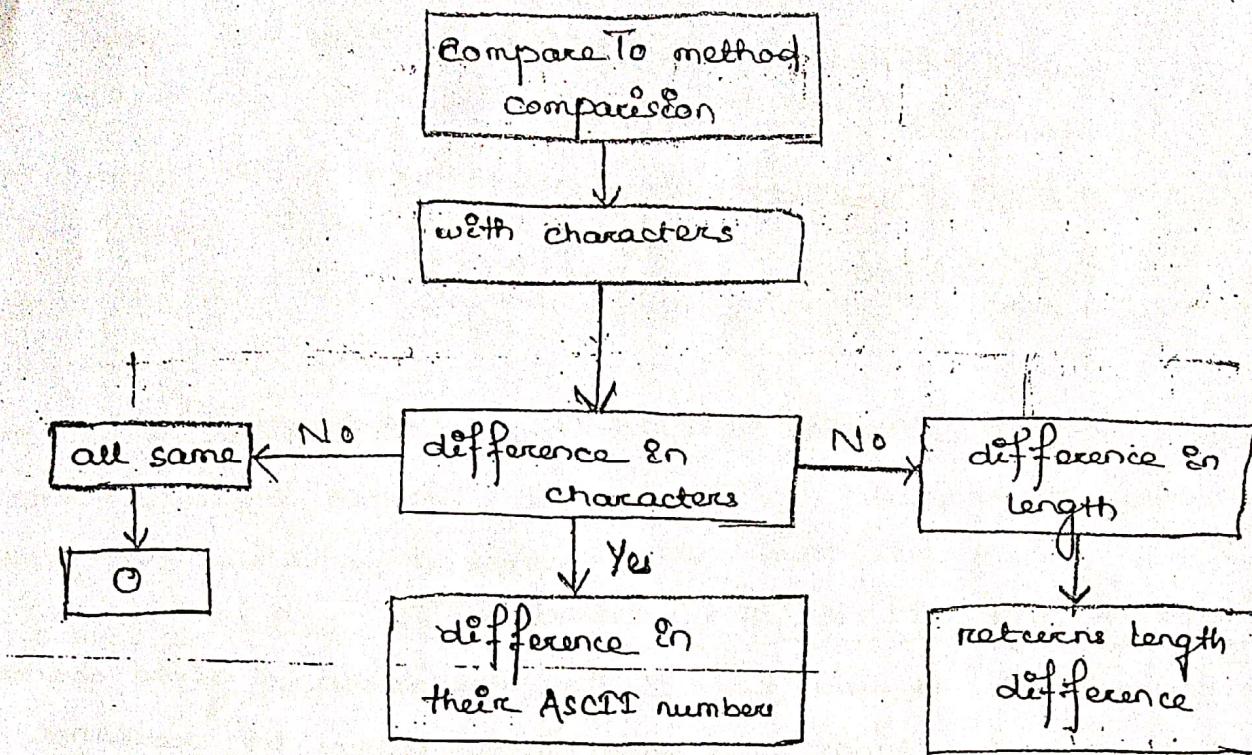


Equals and EqualsIgnoreCase

- In projects username are compared without case and password are compared with case.
- So we must use equalsIgnoreCase() method to compare username and equals() method to compare passwords.

CompareTo() Algorithm

- compareTo() compares between two string by subtracting each character in both string and return the ASCII number difference.
- It continues the subtraction comparison till non-zero number occurred.
- If no. of characters in both string are
- All characters in case equal but no. of characters is different
it returns, current-string object.length - argument String object length.
- Equals() method Internal logic uses compareTo() method.
- In projects equals() method is used for checking purpose, gives input & right or wrong.



Char charAt (int index) :

prototype : public string charAt (int index);

String s1 = "abcd";

Sopln (s1.charAt(0));

Sopln (s1.charAt(2));

Sopln (s1.charAt(4));

Q :- Write a program to read a string from the keyboard and print each character with its position as o/p "abc".

O/p - character 1 : a

2 : b

3 : c

→ import java.util.*;

class CharacterReader {

public static void main (string [] args) { .. }

Scanner scn = new Scanner (System.in);

```
SopIn( "Enter String: ");
```

```
String data = scnr.nextLine();
```

```
Int no of char = data.length();
```

```
For (Int i=0 ; i < no of char ; i++)
```

```
{
```

```
SopIn( "Character" + (i+1) + " is : " + data.charAt [i]);
```

```
}
```

3

IndexOf and LastIndexOf () methods :

→ `IndexOf()` is used for finding the position of given character of substring from begin or `Ende()` and returns 1st occurrence.

→ `LastIndexOf()` returns the last occurrence of the given character or substring

→ Also we have overloaded forms of these two methods to find the position of character or substring from begin to end, `Ende`.

```
String s1 = "Java programming language";
```

```
s1.IndexOf('a'); → 1
```

begin
↓

end

```
s1.IndexOf('a', int fromIndex);
```

begin fromIndex

5
→ 10

```
s1.LastIndexOf('a'); → 32
```

begin
↑

end

```
s1.LastIndexOf('a', int fromIndex);
```

begin fromIndex

5
→ 3

Prototypes:

```
public Int IndexOf (Int ch)
```

```
public Int IndexOf (String s)
```

```
public Int LastIndexOf (Int ch)
```

```
public Int LastIndexOf (String s)
```

```
public int lastIndexOf (char ch, int fromIndex)
public int lastIndexOf (String s, int fromIndex)
```

→ Above ~~two~~ methods returns '-1' if the given char or string is not found in the given string.

→ Note that both methods searching for given char or substring in the given case.

→ If we want to search for the string or character without case we must convert given String all characters to Lower case and then we have to search in the modified String.

Ex:- String s2 = "JavaHari";

```
SopIn (s2.indexOf ("hari"));
```

```
String s3 = s2.toLowerCase();
```

```
SopIn (s3.indexOf ("hari"));
```

→ Above methods ~~return~~ returns '-1' if it doesn't find any given character or string what we should do if we want to boolean value true/false.

→ We should write our own code as shown below:-

Q:- Write a program to read string find "Hari" substring available in the string, if it is available true else returns false. In main method print message ("Hari is available") if it returns true, else print message "Hari is not available".

```
import java.util.*;
```

```
class TestIndexOf {
```

```
static boolean searchIn (String s) {
```

```
if (s.indexOf ("Hari") != -1) {
```

```
    return true;
```

```
}
```

```
else {
```

```
    return false;
```

```
}
```

```
return (s.indexOf("Hari") != -1);
```

```
}
```

```
public static void main (String [] args) {
```

```
Scanner scn = new Scanner (System.in);
```

```
System.out ("Enter String : ");
```

```
String s1 = scn.nextLine();
```

```
boolean bo = searchIn (s1);
```

```
If (bo == true) {
```

```
System.out.println ("Hari is available");
```

```
}
```

```
else {
```

```
System.out.println ("Hari is not available");
```

```
}
```

```
}
```

Case-1.

I/P - javaHari

O/P - Hari is available

Case-2

I/P - JavaHari

O/P - Hari is not available

Q:- How can we get is available in case-2?

→ We must change the given string to Uppercase in SearchIn method as below.

```
static boolean searchIn (String s) {
```

```
s = s.toUpperCase();
```

```
return (s.indexOf ("Hari") != -1);
```

```
}
```

→ In Java SE Sun MicroSystem gives a special method called contains in String class for finding substring or character in

Prototype:

```
public boolean contains(CharSequence s)
```

- In our searchIn method argument is the actual String and subString we hard coded in this method.
- In our contains method argument is the subString for which we have to search and actual

Rule:

- contains() -> We cannot call this method by passing char in '' because it's parameter is CharSequence. So we must pass character with "".

Ex: String s1 = "Java Programming Language";
System.out.println(s1.contains('a')) ; // false
System.out.println(s1.contains("a")) ; // true

'startsWith()' and 'endsWith()':—

Q:- Write a program to count how many text files available in a given array?

→ class TestEndsWith {

```
public static void main (String [] args) {
    String [] fileArray = {"1.txt", "2.doc", "3.txt", "4.pdf",
    "5.xls", "6.txt"};
    for (int i = 0; i < fileArray.length; i++) {
        if (fileArray[i].endsWith(".txt")) {
            count++;
        }
    }
}
```

```
System.out.println (count + " text files are available");
```

prototype : public boolean endsWith (String s)

public boolean startsWith (String s)

Ex:

```
String s1 = "abc bbc cbc";
```

```
Sopln (s1.startsWith ("abc")); // true
```

```
Sopln (s1.endsWith ("cbc")); // true
```

```
Sopln (s1.endsWith ("bbc cbc")); // true
```

```
Sopln (s1.startsWith ("abc bbc cbc")); // true
```

```
Sopln (s1.endsWith ("abc bbc cbc")); // true
```

```
Sopln (s1.startsWith ("bbc cbc")); // false
```

Difference between equals(), contains(), startsWith() and endsWith():—

→ equals() method compares current String character and argument string character completely. It returns true if all are equal.

→ contains() method search for given string as substring.

in current string. It returns true if its characters are available in sequence in the current string at any position.

→ StartsWith() and endsWith() methods are also searching given String as substring.

→ startsWith(). returns true if this given string characters are available in the begin index.

→ endsWith() returns true if the given string characters are available at the end.

substring(): —

prototype: public String substring (int index)
public String substring (int start, int end)

Ex:-

String s1 = "Java Programming Language"

Sopn(s1.substring(5)); // Program

Sopn(s1.substring(5, 12)); // Program

Sopn(s1.substring(5, 11)); // Progra

Sopn(s1.substring(5, 5)); // "

Sopn(s1.substring(12, 5)); // CE:

String s2 = "Java Object-oriented Programming Language";

int startIndex = s2.indexOf("Program");

int endIndex = startIndex + 7;

Q:- Write a program to check "Hari" substring is available in given string.

If available print "Hari" word in the given case with

→ Class

```
Import java.util.*;
```

```
Class TestSubString {
```

```
public static void main (String [] args) {
```

```
Scanner scn = new Scanner (System.in);
```

```
while (true) {
```

```
System.out.print ("In Enter String : ");
```

```
String s1 = scn.nextLine();
```

```
String s2 = s1.toLowerCase();
```

```
boolean available = s2.contains ("Hari");
```

```
If (available) {
```

```
int startIndex = s2.indexOf ("Hari");
```

```
int endIndex = startIndex + 4;
```

```
String Hari = s1.substring (startIndex, endIndex);
```

```
System.out.println (Hari + " is available");
```

```
}
```

```
System.out.println ("Hari is not available");
```

```
}
```

```
}
```

Enter String : JavaHariKrishna

Hari is available.

Enter String : JavaHariKrishna

Hari is available.

Enter String : Hari

Hari is available.

Enter String : HariKrishnaJava

Hari is not available.

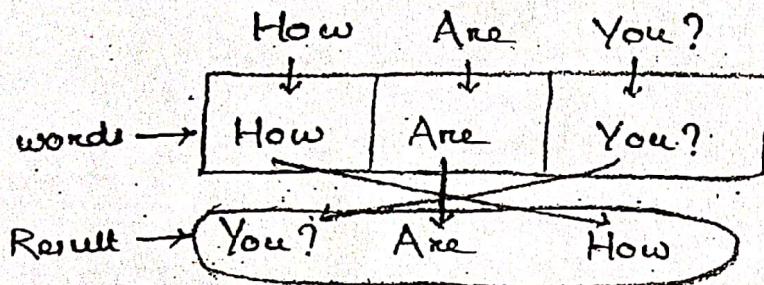
split() :-

Q:- Write a program to reverse words in given String

I/P - How Are You?

O/P - You? Are How

A) Procedure:



class StringReverse {

 static String reverseWords (String s) {

 String [] words = s.split (" ");

 String result = " ";

 for (int i = (words.length)-1; i>=0; i--) {

 result = result + words [i] + " ";

}

 return result.trim();

}

class TestSubString {

 public static void main (String [] args) {

 Scanner scn = new Scanner (System.in);

 System.out.println ("Enter String :");

 String s1 = scn.nextLine();

 System.out.println ("You entered : " + s1);

 String result = StringReverse.reverseWords (s1);

 System.out.println ("Result : " + result);

}

}

valueOf():

- It is a overloaded method used for converting any Java data type value or object to String type.
- This method is used internally by compiler and JVM.
- If we concat any other data type value with the string using '+' operator.

Prototype: | public static String valueOf(XXX type)

(Any Java data type is allowed)

- It is a overloaded method has 9 forms with parameters int, long, float, double, char, char[], boolean, (char[]), String, Object.
- Byte and short parameter methods are not there because there is no literals subtype byte and short.

⇒ Ex:-

Sopln ("data:" + 10); 10 is converted as string by compiler using valueOf() method as shown below.

Sopln ("data:" + String.valueOf(10));
→ Sopln ("data:" + " 10");
→ Sopln ("data: 10");
→ data: 10

String Handling using StringBuffer Object: —

- A buffer means precast memory that means it creates some default capacity memory prior to store.
- StringBuffer object is internally maintains a char[] object by default 16 location.

- So we can say StringBuffer object default capacity is 16 buffers (i.e. 16 locations).
- It's capacity is incremented after when it's size is crossing it's capacity.
- Here size means no. of characters we have stored (let us say 4 characters).
- Here capacity means no. of characters we can store (16 by default).
- StringBuffer class has below 2 methods to find capacity and length.

```
public int capacity()
public int length()
```

for details read
String Handling chapter in
Vol - 2

Tip to read name from the Keyboard. If entered name is empty display error message, prompt message 'Enter Name Again'.
If name is entered correctly display message 'Hi name'
If spaces only entered also throw Error.

14/7/15

→ To develop this project we must use isEmpty() method of also trim()
method from String class.

Welcome to Naresh IT'

//isEmptyTest.java

import java.util.*;

public class IsEmptyTest{

 public static void main(String args){

 Scanner scn = new Scanner(System.in);

 while(true){

 System.out.println("Enter name");

 String name = scn.nextLine();

 if(name.isEmpty()) {

 System.out.println("Name is required");

 Continue;

 }

 else {

 String resName = name.trim();

 if(resName.isEmpty()) {

 System.out.println("Name is required");

 Continue;

 }

 System.out.println("Hi " + resName);

 System.out.println("Welcome to Naresh IT");

 break;

 }

 }

}

⇒ Real time example verifying textfield is empty or not

Project 2

Read password from the keyboard. Verify its length is b/w 8 to 16 if yes display message 'Registration successful' else throw error 'password length should be b/w 8-16'

→ // PasswordLengthTest.java

```
import java.util.*;  
  
public class LengthPasswordTest {  
    public static void main(String[] args) {  
        Scanner scn = new Scanner(System.in);  
        while (true) {  
            System.out.println("Enter Password:");  
            String pwd = scn.nextLine();  
            if (pwd.isEmpty()) {  
                System.out.println("Password should not be empty");  
                continue;  
            } else {  
                int len = pwd.length();  
                if (len < 8 || len > 16) {  
                    System.out.println("Password length should be 8 to 16");  
                    continue;  
                } else {  
                    System.out.println("Registration completed successfully");  
                    break;  
                }  
            }  
        }  
    }  
}
```

WAP to validate mobile number length. If 10 digits not existed
throw error message. In this project we must implement totally 4
validations. 1. Empty or not . 2. All all digits are not
3. Ten digits exists or not 4. Is it really mobile number or not ^{send}
_{sms} (OTP)

// MobileValidator.java

```
import java.util.*;
```

```
public class MobileValidator{
```

```
    public static void main(String[] args){
```

```
        Scanner scn = new Scanner(System.in);
```

```
        while(true){
```

```
            System.out.println("Enter Number: ");
```

```
            String mobile = scn.nextLine();
```

```
            if(mobile.isEmpty()) {
```

```
                System.out.println("Mobile no. is Mandatory");
```

```
                continue;
```

```
            } else {
```

~~long mn = Long.parseLong(mob~~~~try {~~

```
long mn = Long.parseLong(mobile);
```

```
if(mobile.length() != 10) {
```

```
                System.out.println("Mobile no. should be 10 digits");
```

```
                continue;
```

```
}
```

```
else {
```

```
                System.out.println("Activation Key has send to your no.");
```

```
                System.out.println("Enter Activation Key:");
```

```
            } break;
```

```
}
```

```
Catch(NumberFormatException nfe){
```

```
                System.out.println("Mobile no. should contain only digits");
```

```
                continue;
```

```
}
```

Develop a program to find no. of vowels & consonants available in the given String. Read string from keyboard.

→ // CheckVC.java

```
import java.util.*;
```

```
public class CheckVC {
```

My code

```
P S V m(S[] args) {
```

```
String s1
```

```
P & checkVC {String s1,
```

```
{
```

```
this.s1String = s1;
```

```
}
```

```
public toString () {
```

```
return
```

```
import java.util.*;
```

```
public class VowCons {
```

```
P S V m (S[] args) {
```

```
Scanner scn = new Scanner((System.in))
```

```
System.out.println("Enter a string");
```

```
String s1 = scn.nextLine();
```

```
String s2 = s1.toLowerCase();
```

```
int vCount = 0;
```

```
int cCount = 0;
```

```
for (int i=0; i < s2.length(); i++) {
```

```
char ch = s2.charAt(i);
```

60

```

if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
    vCount++;
}
else {
    cCount++;
}
}

```

```

System.out.println("Available vowels are: " + vCount);
System.out.println("Available consonants are: " + cCount);
}

```

Develop a program to register a user account in website. In this project you develop code for validating password criteria.

1. password should contain atleast 1 uppercase Alphabet.
2. password should contain 1 digit.
3. password should contain 1 special character.
4. password length should be betn 8, 16.

→ // PasswordPatternValidator (sir)

// LoginPage.java

import java.util.*;

public class LoginPage {

public static void main(String[] args) {

Scanner scn = new Scanner(System.in);

// System.out.println(" Enter Username : ");

// while(true){

String usn = scn.nextLine();

System.out.println(" Enter Password : ");

String pwd = scn.nextLine();

// if (pwd.isEmpty()) {

// System.out.println(" Enter Password : ");

// }

MineCode

```
if (pwd.isEmpty()) {  
    System.out.println("Continue");  
}  
else if (  
    if (char ch >= 65 && ch <= 90) {  
        upperfound = true;  
    }  
    else if (char ch >= 97 && ch <= 122) {  
        lowerfound = true;  
    }  
    else if (Character.isDigit(ch)) {  
        digitfound = true;  
    }  
    else if (Character.isPunctuation(ch)) {  
        spCharfound = true;  
    }  
}
```

```
//PasswordPatternValidator.java  
import java.util.Scanner;  
public class PasswordPatternValidator {  
    public static void main(String[] args) {  
        Scanner scn = new Scanner(System.in);  
        boolean upperfound = false;  
        boolean digitfound = false;  
        boolean spCharfound = false;  
  
        while (true) {  
            System.out.print("Password: ");  
            String pwd = scn.nextLine();  
            int len = pwd.length();  
  
            if (len < 8 || len > 16) {  
                System.out.println("Password should contain 8-16 characters");  
            }  
            else {  
                for (int i = 0; i < len; i++) {  
                    char ch = pwd.charAt(i);  
                    if (Character.isLetter(ch)) {  
                        if (Character.isUpperCase(ch)) {  
                            upperfound = true;  
                        }  
                        else if (Character.isLowerCase(ch)) {  
                            lowerfound = true;  
                        }  
                    }  
                    else if (Character.isDigit(ch)) {  
                        digitfound = true;  
                    }  
                    else if (Character.isPunctuation(ch)) {  
                        spCharfound = true;  
                    }  
                }  
                if (upperfound && lowerfound && digitfound && spCharfound) {  
                    System.out.println("Valid Password");  
                }  
                else {  
                    System.out.println("Invalid Password");  
                }  
            }  
        }  
    }  
}
```

```

        else {
            SpCharfound = true;
        }

    } // for close

    if(upperfound && digitfound && spcharfound) {
        Sopln("Registration success");
        break;
    }
    else {
        Sopln("Password should contain");
        Sopln("1. One uppercase alphabet");
        Sopln("2. One digit");
        Sopln("3. One special character");
    }
}

} // else close
} // while
} // main
} // class

```

~~Index~~

KIAP to read a string from the keyboard. If this string contains substring "hari" then display 'String hari available' else display 'String hari is not available'.

~~KIAP~~

Rewrite above program to display the substring hari is available even though characters h,a,r,i are available at diff. places in given string.

Rewrite above program substring hari is available even though character h,a,r,i available in diff. places but the character sequence should be "hari"

Validate the given email is a email or not.

Validation: This email should contain '@' & '.' character, '@' should come after @.

WAP to read a string from the Keyboard. If this String contains substring "hari" then display 'substring hari available' else display 'substring hari is not available'

→ // SearchString

```
import java.util.Scanner;  
public class SearchString {  
    public static void main(String args){  
        Scanner scn = new Scanner(System.in);  
        System.out.println("Enter String");  
        String s1 = scn.nextLine();  
        String s2 = "Hari";  
        for(int i=0; i<s1.length(); i++) {  
            char ch = s1.charAt(i);  
        }  
        public int indexOf(String s2){  
            if(s2.equals(s1.substring(i, i+s2.length()))){  
                System.out.println("Substring Hari available");  
            } else {  
                System.out.println("Substring Hari not available");  
            }  
        }  
    }  
}
```

though characters b,a,r,i are available at diff. places in given string.

```
// SubString  
import java.util.Scanner.*;  
Public class Substring {  
    P S v m (S[] args) {  
        Scanner scn = new Scanner(System.in);  
        SopIn("Enter a String: ");  
        String s1 = scn.nextline();  
        for (int i=0; i < s1.length(); i++) {  
            char ch = s1.charAt(i);  
            if (ch == 'b' || ch == 'a' || ch == 'r' || ch == 'i') {  
                SopIn(" Available");  
            }  
            else {  
                SopIn(" Not Available");  
            }  
        }  
    }  
}
```

Rewrite above program substring han is available even though character h; a, r, i available in diff places but the character sequence should be 'han'

→ //

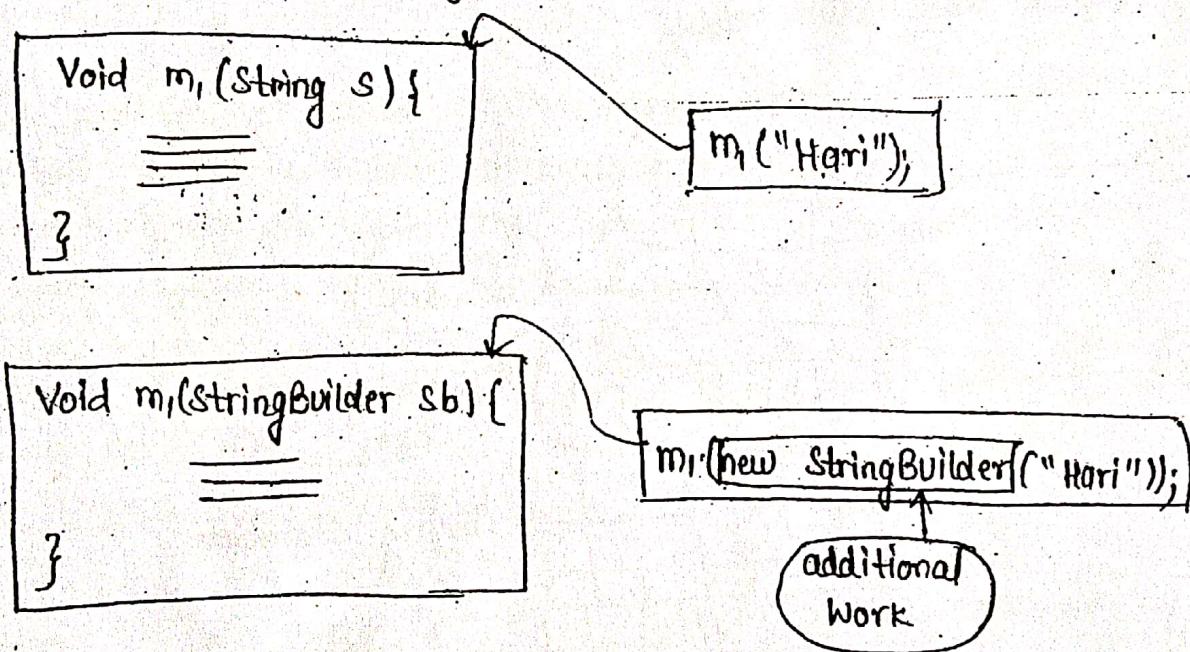
```
import java.util.Scanner.*;  
public class  
P S V m{S[ ] args}{
```

Email Validator program :- (Q in back page)

```
→ // EmailValidator.java  
import java.util.*;  
public class EmailValidator {  
    public static void main(String args) {  
        Scanner scn = new Scanner(System.in);  
  
        System.out.println("Email:");  
        String email = scn.nextLine();  
        int atIndex;  
        if ((atIndex = email.indexOf('@')) != -1) &&  
            (email.indexOf('.', atIndex) != -1)) {  
            System.out.println("Valid email");  
        }  
        else {  
            System.out.println("Invalid email");  
        }  
    }  
}
```

What is the right parameter for taking String data as argument,
returning string data from a method? 26/7/15

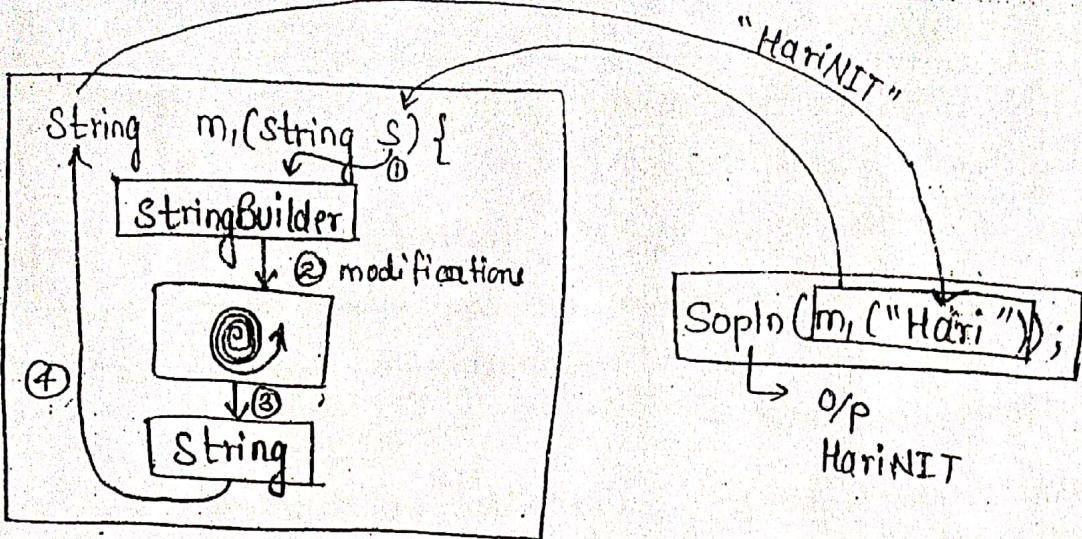
- Parameter type & return type of the method should be `java.lang.String`.
- If we take String as a parameter we can pass string literal directly without using new keyword & constructor. Even we can read string data from keyboard.
- If we take `StringBuffer` or `StringBuilder` as parameter for passing String data we must create object using new keyword & constructor explicitly. It increases more lines of code burden to method call.
- Below code shows creating a method with `String` & `StringBuilder` as the parameter & calling them.



→ If we take a `String` as parameter, if you modify string data stored in the `String` object for every modification new string object is created with result. How should we solve this performance issue. Inside the method convert `String` object to `StringBuilder` object.

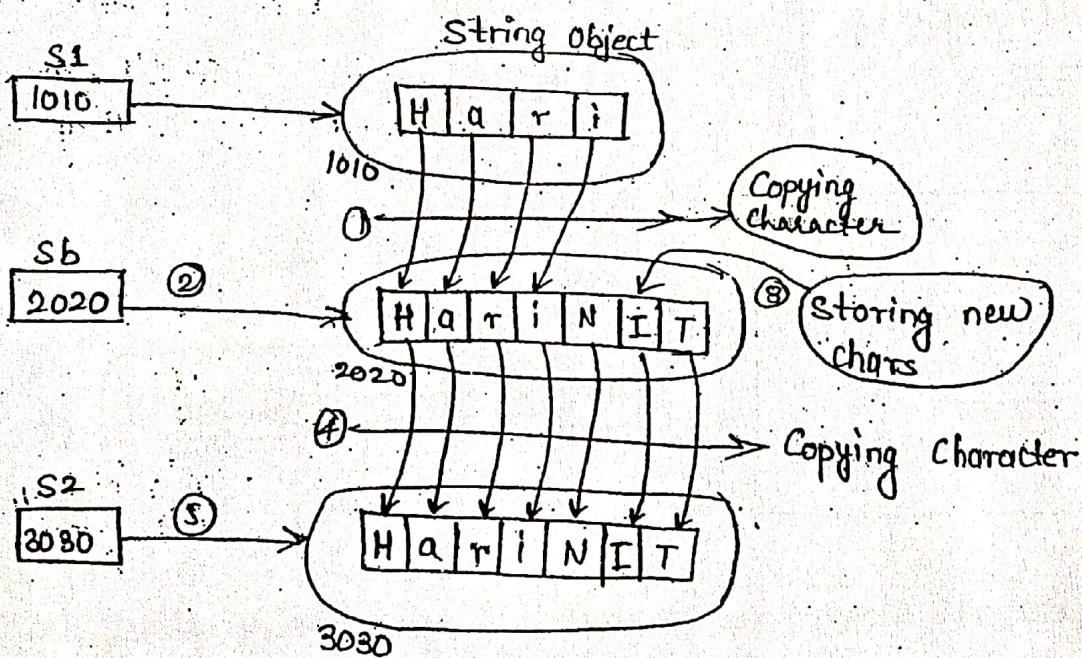
→ Do reqd. modification on `StringBuilder` object.

→ Convert the result `StringBuilder` object to `String` object, return this `String` object to method caller as shown below. →



String & StringBuilder Objects are incompatible then how can we convert String to StringBuilder & stringBuilder to string?

→ Here converting means not casting string to stringBuilder rather copying characters from String object to stringBuilder & stringBuilder to string object as shown below.



What is the API we must use to convert String to stringBuilder & viceversa?

→ In String class & stringBuilder class we have constructor for copying characters from string object to stringBuilder object & stringBuilder object to string object.

class String {

String (StringBuilder sb) {}

String (StringBuffer sb) {}

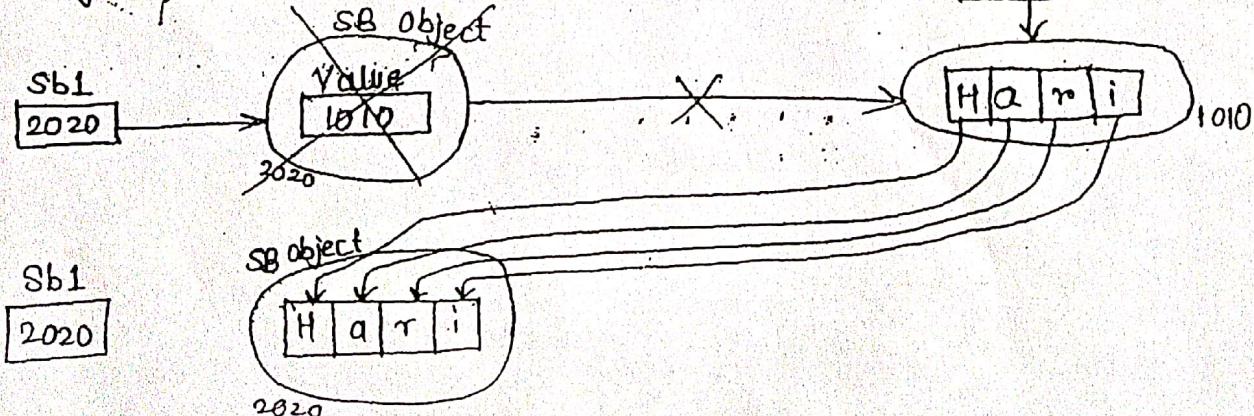
class StringBuilder {

StringBuilder (String s) {}

The diff b/w normal class constructors to String & StringBuilder constructors in normal class constructors stores given argument data in current object whereas String, StringBuilder class constructors will copy given argument object characters into current object. They will not store arguments as shown below.

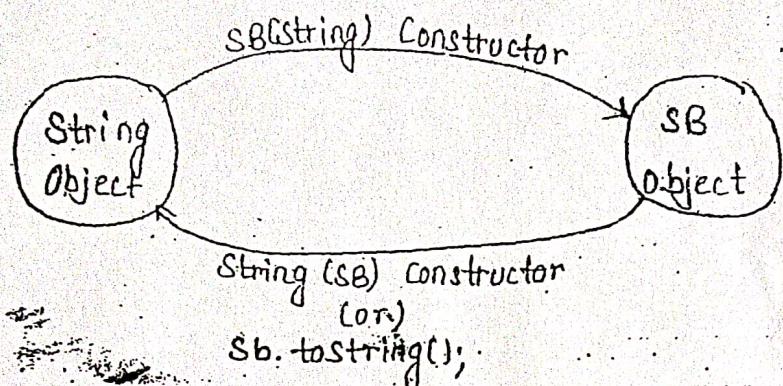
String s1 = "Hari";

StringBuilder sb1 = new SB(s1);



We can convert StringBuffer to String object in two ways

1. Using Constructors
2. Using toString method



Develop a program to take string data as argument say "Hari", append NareshIT characters each character individually to the argument string data, return result string.

→ class StoSBConversion {

```
    public static void main(String[] args) {  
        String s1 = "Hari";  
        String s2 = modify(s1);  
        System.out.println("s1: " + s1); // → Hari  
        System.out.println("s2: " + s2); // → HariNareshIT.  
    }
```

//Converting String to StringBuilder

```
    static String modify(String s1) {  
        StringBuilder sb = new StringBuilder(s1); ① S to SB conversion  
        String s2 = "NareshIT";  
        for(int i=0; i < s2.length(); i++) {  
            sb.append(s2.charAt(i)); ② Modifications in StringBuilder object  
        }  
        return sb.toString(); ③ Modifications in StringBuilder object  
    } ④ Converting SB to String
```

} (4) Returning String