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

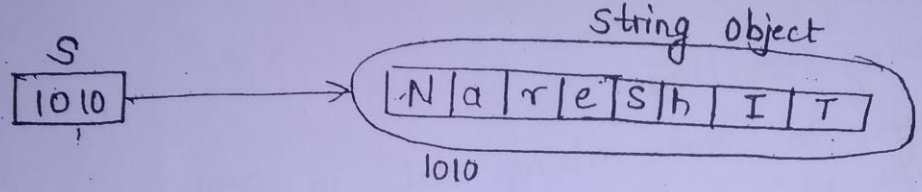
1. What is string?
2. What is string handling?
3. Different ways to store string data in JVM.
4. Difference b/t String, StringBuffer, StringBuilder classes.
5. Immutable and Mutable objects.
6. ThreadSafe and Non ThreadSafe objects.
7. StringPolling
8. String constant Polled Area.
9. Garbage Collection with String Object
10. String class constructors and methods
11. StringBuffer class constructors and methods.
12. Programs
13. Interview & SCJP Questions.

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?

→ Performance 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, StringBuilder.

→ 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 eg.

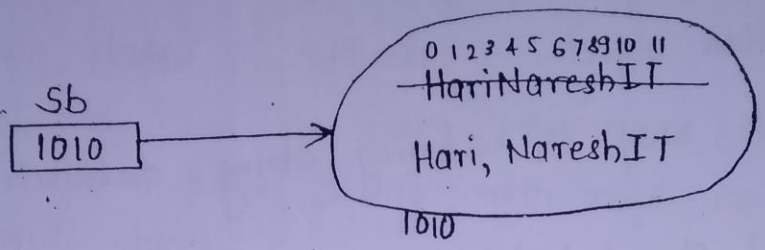
```
String S1 = "Naresh IT";
String S2 = "Naresh IT";
if (S1.equals(S2)){
    Sopln("Hi");
}
else{
    Sopln("Hello");
}
```

→ Comparing characters in S1 & S2.

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

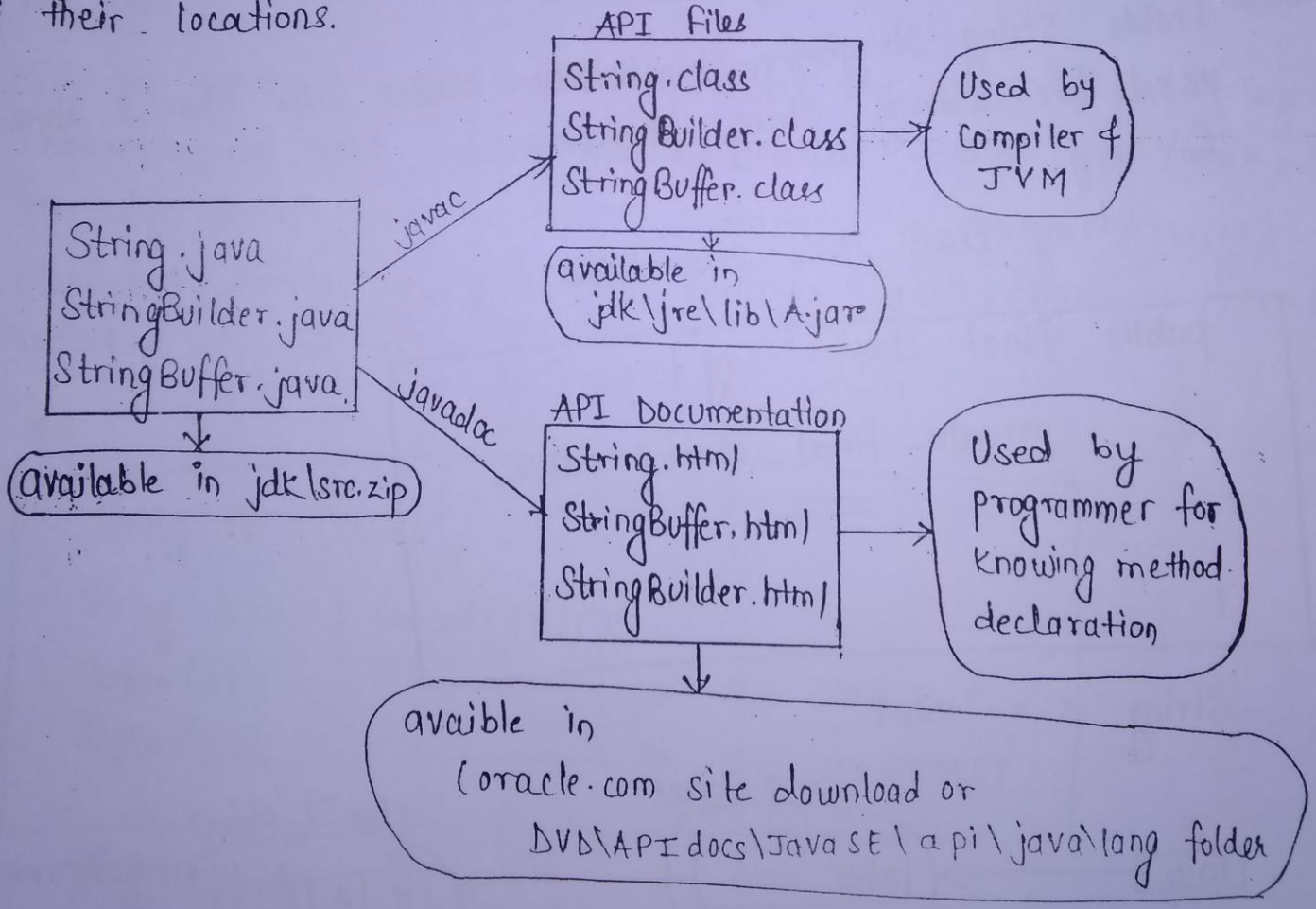
eg.

```
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?

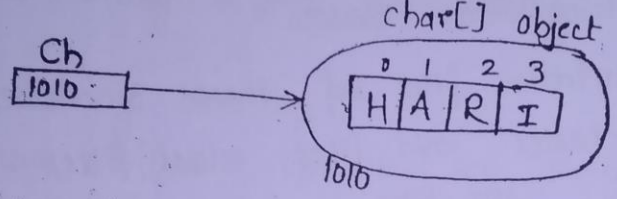
23/7/15

→ 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' };
```

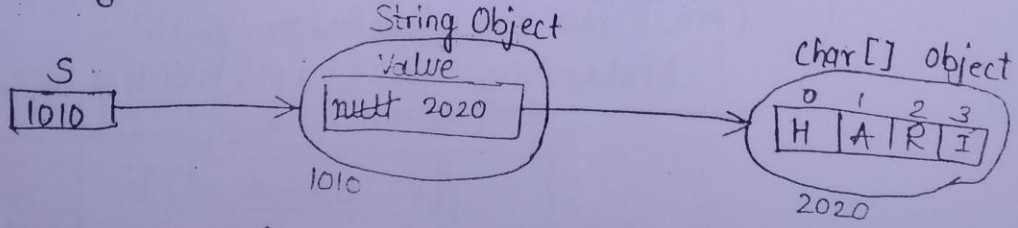


→ Inside String, StringBuffer, StringBuilder classes also char[] object is used for storing string characters.
for e.g.

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 oproblems 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 <u>immutable</u> 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.

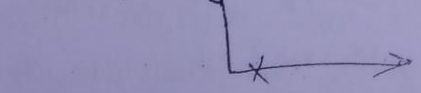
→ For e.g.

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.

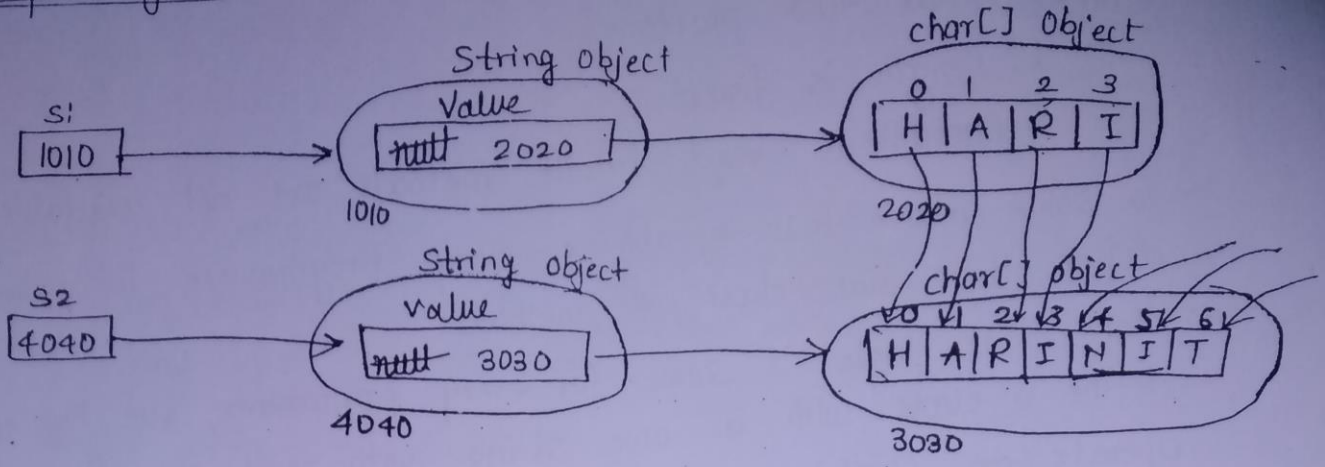
e.g.

```
String s1 = "HARI";
String s2= s1.concat("NIT");
Sopln(s1);          ------>  O/P HARI
Sopln(s2);          ------>  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 modify. If its data is modified the value mapped to this object cannot be retrieved from this Map because other programmer will try to retrieved 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.

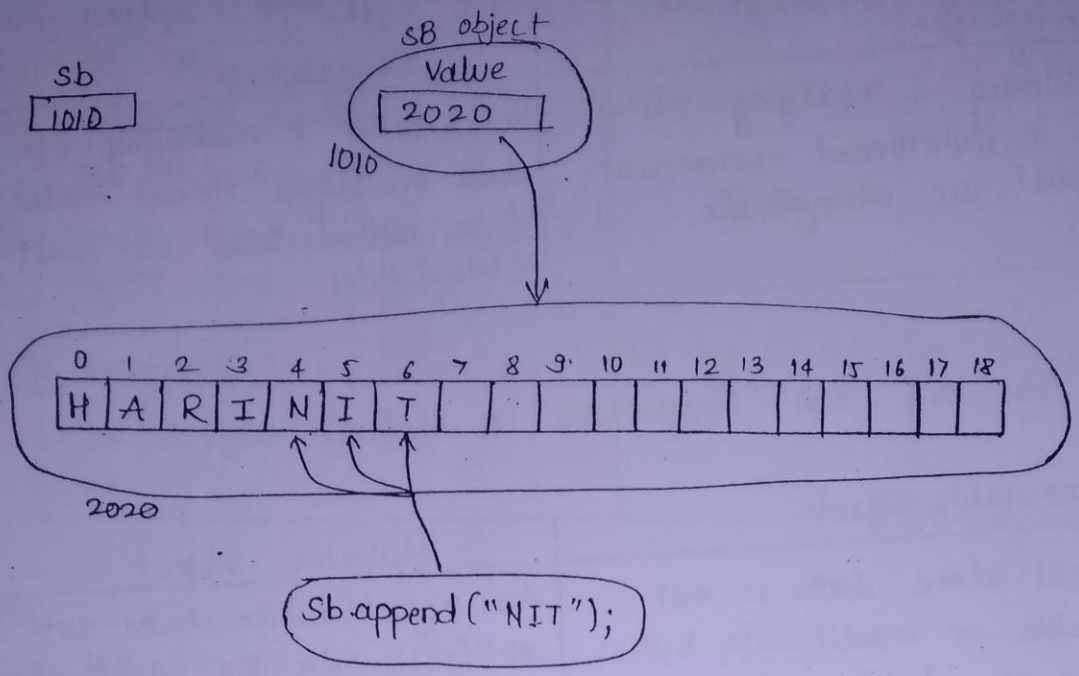
→ That means the modifications we are doing on string data in StringBuffer object the result will be stored in the current StringBuffer object, new StringBuffer object will not be created.

for e.g.

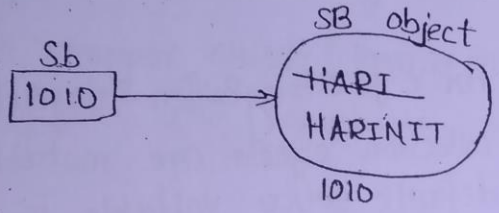
```
StringBuffer sb1 = new StringBuffer("HARI");
StringBuffer sb2 = sb1.append("NIT");
Sopln(sb1);  ——→ o/p : HARINIT
Sopln(sb2);  ——→ o/p : HARINIT
```

25/7/15

StringBuffer Memory Diagram



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 & 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	StringBuffer
If we d'nt 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.	If you want to modify String data in same object we must use ~~String~~ StringBuffer.

When should we use StringBuffer & StringBuilder

→

StringBuffer	StringBuilder
for storing & modifying string data in multithread environment we must use stringBuffer.	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	Mutable object
→ An object whose data is not modifyable or modifyable but result is stored in new object is called Immutable.	An object whose data can be modifyable in the same object is called Mutable object.
→ 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.	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	Non-threadsafe.
An object which is not available to multiple threads at a time to modify its data concurrently is called threadsafe.	An object which is available to multiple thread at a time to modify its data concurrently is called Non-threadsafe objects. 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	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.

is 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. for 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.

—x—

Imp

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

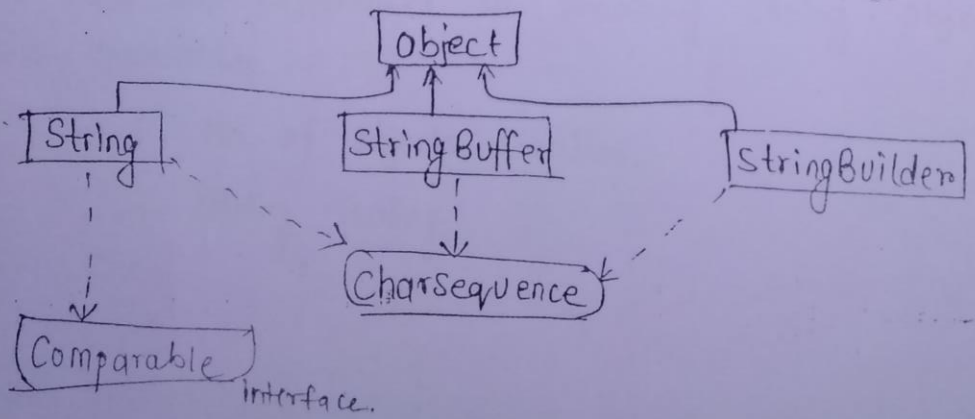
What is String, StringBuffer, StringBuilder?

→ 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.



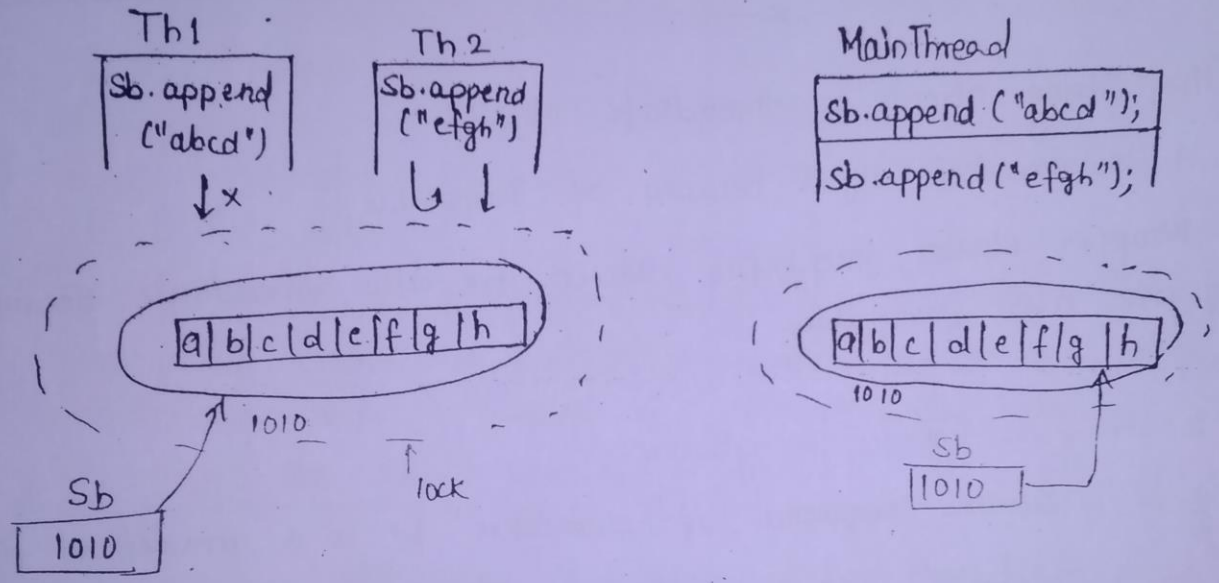
→ These 3 objects are siblings, they ~~did~~ d'nt have inheritance relation because behaviour is different.

→ 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 & 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.

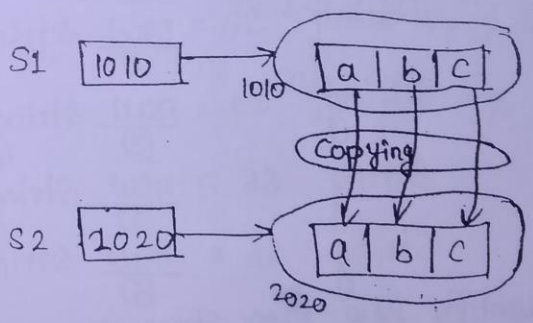
→ The actual way of string object creation approach is String Literal approach. Using new keyword & constructor we cannot create the very first object.

→ We must new keyword & constructors 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 use keyword constructors approach.

for e.g

```
String S1 = "abc";
String S2 = new String (S1);
```

(assignment, 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.

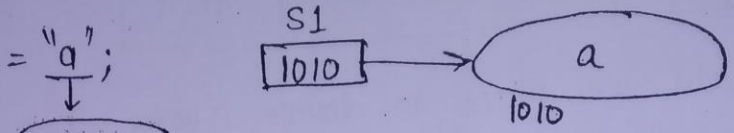
→ In Literal approach only 1 object is created.

→ In new Keyword approach 2 objects are created.

for Eg:

Approach #1:

String s1 = "a";

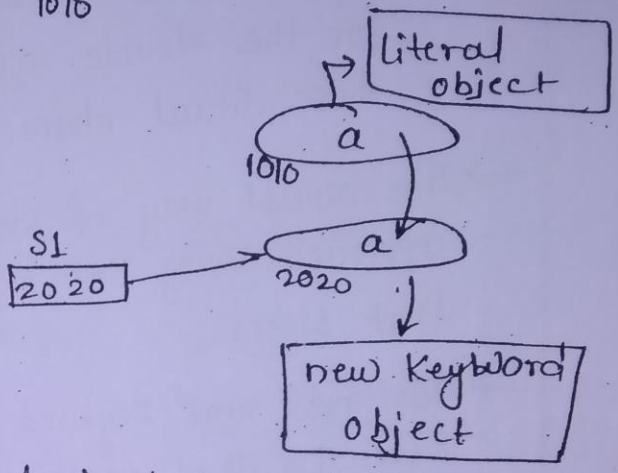


1 object

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	→ 1
String s2 = "a";	→ 0
String s3 = new ② String("a");	→ 1
String s4 = new ③ String("a");	→ 1
String s5 = new ⑤ String("b"); ④	→ 2.
String s6 = new ⑥ String("b");	→ 1
	6

javap -verbose Test

final String s18 = "p";

final String s19 = "q";

String s20 = s18 + s19;

String s21 = s18 + s19;

sopln(s20 == s21); true

1 obj is created

Identify how many String obj are created in below statements?

String s1 = "a";	1
String s2 = "a";	0
String s3 = "A";	1
String s4 = new String("a");	1
String s5 = new String("b");	1
String s6 = new String("b");	1
String s7 = new String(s6);	1
String s8 = s1.concat(s2);	1
String s9 = s1.concat(s2);	1
String s10 = s1 + s2;	1
String s11 = s1 + s2;	1
String s12 = "b";	0
String s13 = "ab"	1
String s14 = "a" + "b";	0
String s15 = s1 + s12;	1
String s16 = scn.nL();	1
String s17 = scn.nL();	1

Total 15 obj created

Compiler removes variable with final

String s20 = "p" + "q";

↑ Compiler takes it like this in both s20 & s21 statements.

based on this modified code only 1 obj is created.

* 2nd obj is not created if it is same data constant obj.
* If the string is not const. even though it is same data string obj, always 2nd obj is created.

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.

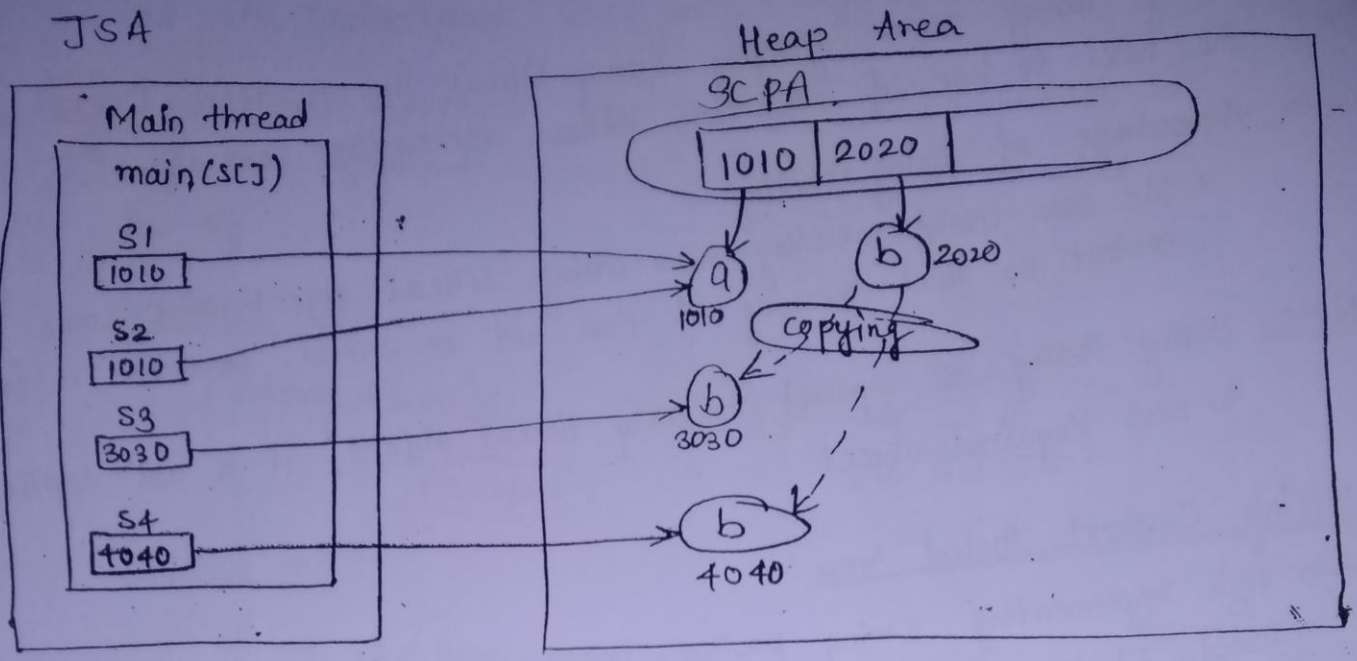
String Constant Pooled Area :-

→ 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 1.0 is Vector.)

→ Below diagram shows String object creation in two ways, storing them in pooled Area.

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



→ 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";
}
```

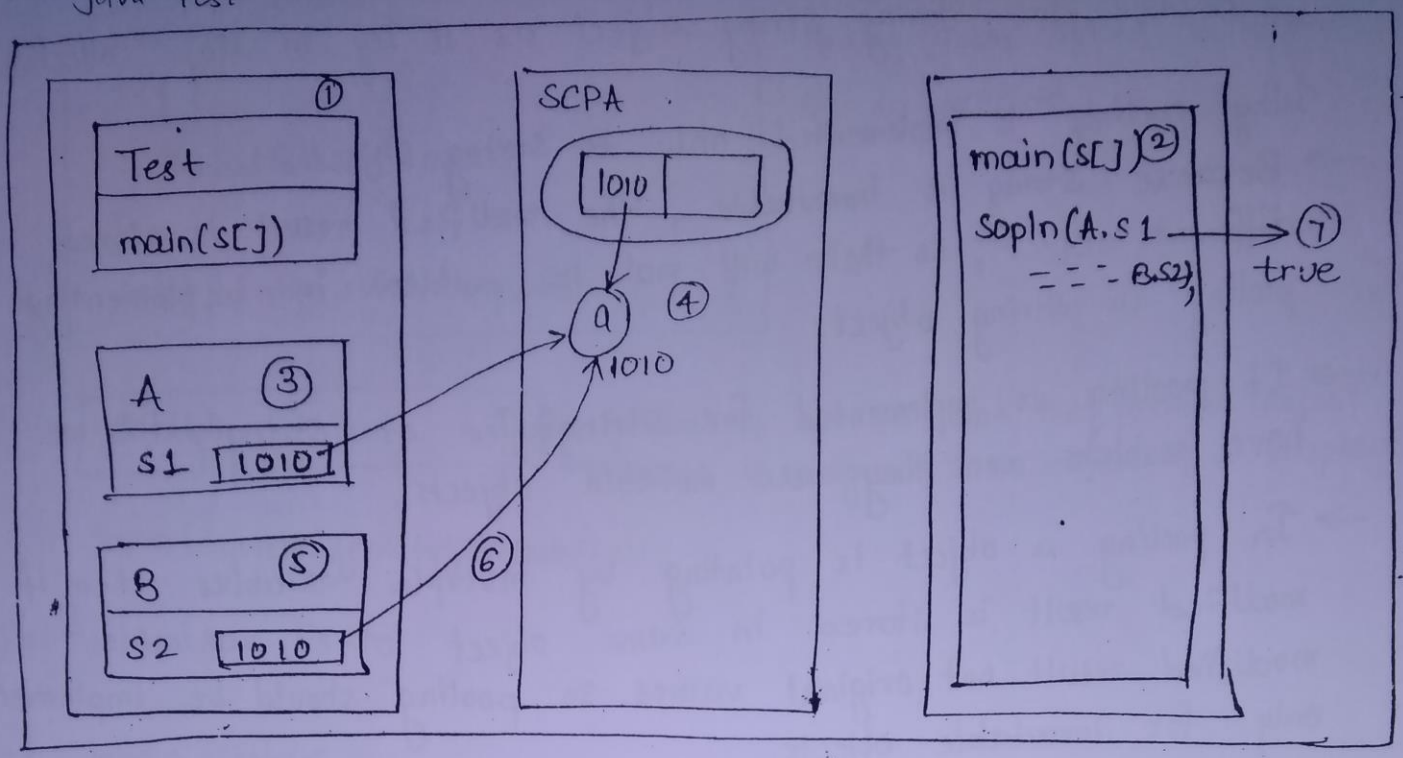
Test.java

```
class Test {
    P S V m(S[] args) {
        Sop ln(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

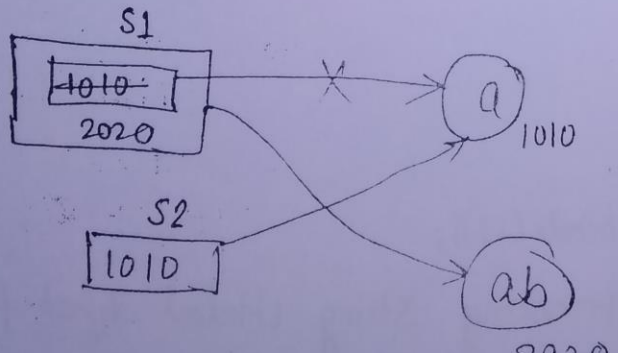


Because of string Pooling string object is pointing by multiple variables then d'nt we have problem when we modify String object using one variable?

→ No, we d'nt 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 their 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 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{
    p s v m(S[] args){
        String s1="a";
        Sopln(system.identityHashCode(s1));
        s1.null;  ⟶ Unreferencing String Literal object from prg.
```

`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

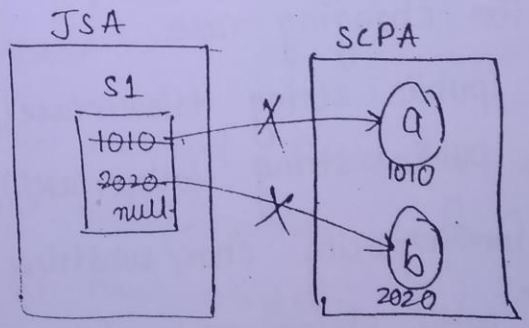
21559264

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;
```

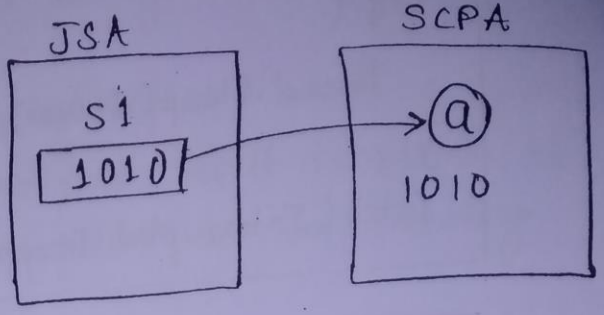


How can we make String Variable also Immutable?

→ By declaring it as final

Ex:

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



→ 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 a 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.

→ This methods will never modify data in current object.

Those methods are

1. for Concatenation

   public String concate (String s)

2. for changing case

   public string toLowerCase()

   public string toUpperCase()

3. for replacing char/substring

replace all occurances:

   public string replace (char oldchar, char newchar)

   public string replace (CharSequence oldstring, CharSequence newstring)

   public string replaceAll (String oldstring, string newstring)

replacing only first occurance:

   public string replaceFirst (String os, string ns)

4. For removing begin & end spaces

   public String trim()

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 immuta~~

→ 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.

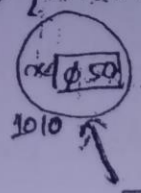
iv) Define getter method to read values.

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

```
class Example {
    private int a;
```

```
Class Test {
    P.S.V main (String [] args) {
```

```
public Example (int x) {
    this.x = x;
}
public int getX() {
    return x;
}
}
```




```
Example e = new Example(50);
e.x = 7 ; XCE: variable has
                private access
Sopln(e.getX());
}
}
```

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

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

```
class Test {
P S V main (String[] args)
Sample s1 = new Sample(50)
Sample s2 = s1.getX(7);
Sopln (s1.getX());
Sopln (s2.getX());
}
}
```

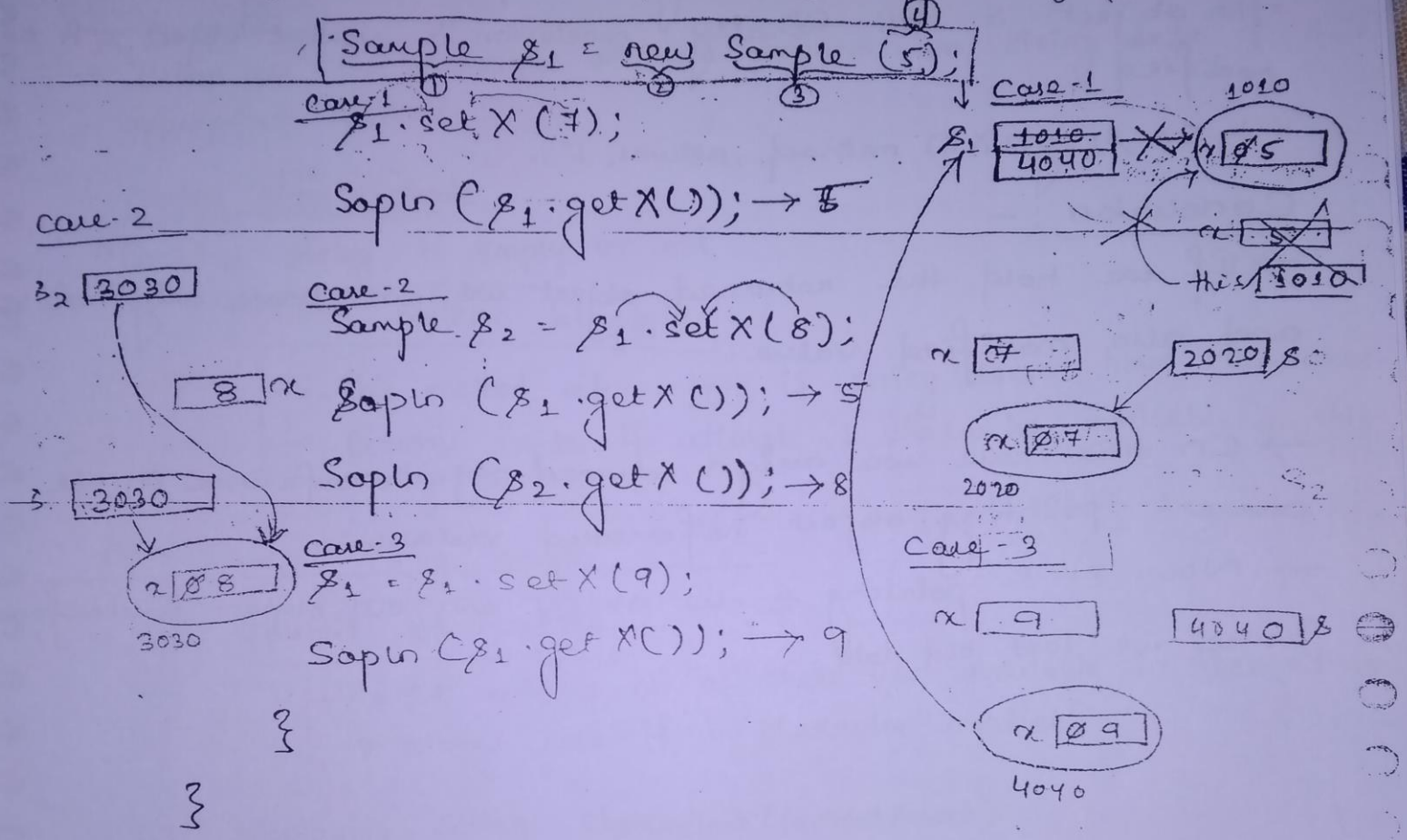
→ For comparision all wrapper classes are developed by Example class (No setter method)

→ String class is developed by Sample class (setter or mutator 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 (5);
        s1.setX (7);
        Sopln (s1.getX());
        Sample s2 = s1.setX (8);
        Sopln (s1.getX());
        Sopln (s2.getX());
        s1 = s1.setX (9);
        Sopln (s1.getX());
    }
}
```



→ In above program 5 methods are executing.

i) main ()
ii) Sample (5)
iii) setX ()
iv) Sample ()
v) getX ()

→ So 5 stack frames are created.

s1.setX (7) call :

→ Here setX (7) method stack frame is created.
→ Parameter 'x' variable created with argument 7.
→ Local object created using no-arg constructor. As part of logic

object & created with 'x' variable with default value.
→ Storing argument value in this local object.
→ Returning local object reference to main() method and s_1.setX is replaced with this reference 2020.
→ Since we are not storing this reference in any destination variable the object is lost. Finally conclusion is current object s_1 is not modified.
→ So s_1.getX() 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 it is pointing to new object and old object is unreferenced 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. 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. Finding string length

`public int lenght()`

→ 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 overriden is 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 overriden 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 overriden 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 StringIndexOutOfBoundsExceptions.

08. Finding a index of a given character / substring

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

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

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

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

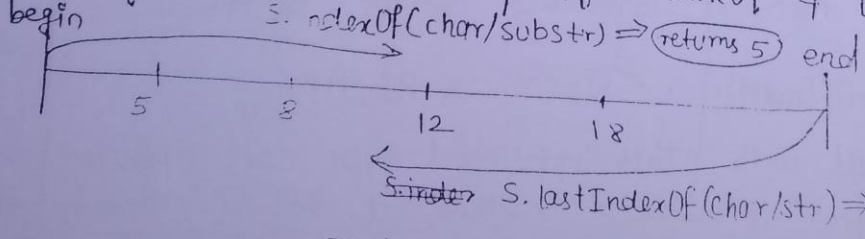
```
public int indexOf(char ch, int fromIndex)
public int indexOf(String s, int fromIndex)
```

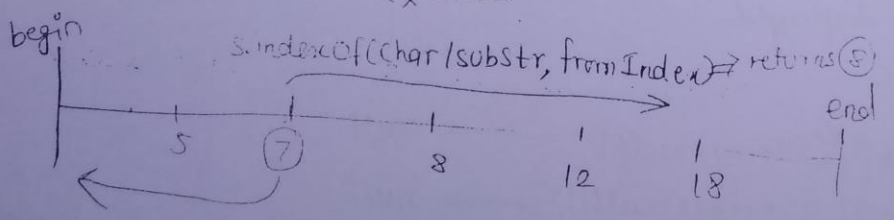
```
public int LastIndexOf(char ch, int fromIndex)
public int LastIndexOf(String s, int fromIndex)
```

→ These two methods will return the first occurance 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 occurance.

Working ~~Index~~ functionality of IndexOf & LastIndexOf()





S.lastIndexOf(char/substr, fromIndex) ⇒ returns 5

09. Searching character / Substring in this given string in same sequence

```
public boolean contains(String s)
```

10. Verifying this string startsWith given substring

```
public boolean startsWith(String s)
```

11. Verifying this string endsWith given subString

```
public bodean endsWith(String s)
```

12. Retriving 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 than 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.

Sopln(String.valueOf(null)); ✓ NOCE ✗ RE: NPE

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

for e.g:

Sopln(String.valueOf((String)null)); o/p → null

Sopln(String.valueOf((Integer)null)); → null

Eg:-

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

class Test {
    P S V m(S[] args) {
        Student s1 = new Student();
        Sopln(s1);
    }
}
```

② ③ ④ "101" ⑤ 101

14. Converting String to char[] (charArray)

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

15. Converting String to byteArray

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

Example for charArray

```
interface A {
    char[] getPWD(int uid);
}
```

uid = 1

```
class B implements A {
    public char[] getPWD(int uid) {
        String pwd = ;
        return pwd.toCharArray();
    }
}
```

NIIT

UID = 1

NIIT

Uid	Uname	PWD
1	HK	NIIT
2	BB	Acting

→ So far we have learnt methods for either retriving characters or for Comparing characters.

- - x - -

→ 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 Microsystem defined multiple methods in String class to perform different operations on String data.

Q:- Find how many characters available in the below 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 Hari Krishna

1. 4
2. 11 → Complete string is read by args[0] as there is
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 ~~BufferReader~~ br.ReadLine() to reade 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 scn.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 scn.nextLine() reads complete line

Q:- If we use scn.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

Because scn.next() reads only first word in the string

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

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

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

→ Length paranthesis 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); x CE: cfs
String[] sa = new String[10];
System.out.println(sa.length()); x CE: 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()

iii) replace(), replaceAll(), replaceFirst()

→ So as a result of above methods of 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 cases:

- Case #1: Returned object is not assigned to any variable
- Case #2: Returned object is assigned to new referenced 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.

Concatination:-

→ Placing arguement String object characters at end of current String Object characters and storing result in new String Object and returning it's reference is called concatinating.

→ We can perform concatination operation in 2 ways.

i) using '+' operator

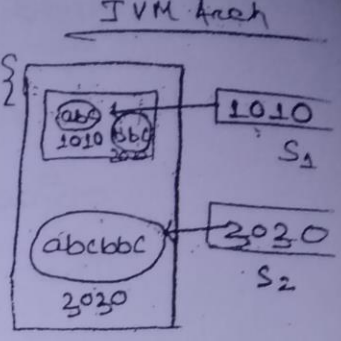
ii) using concat method

ii) Concation 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
    }
}
```

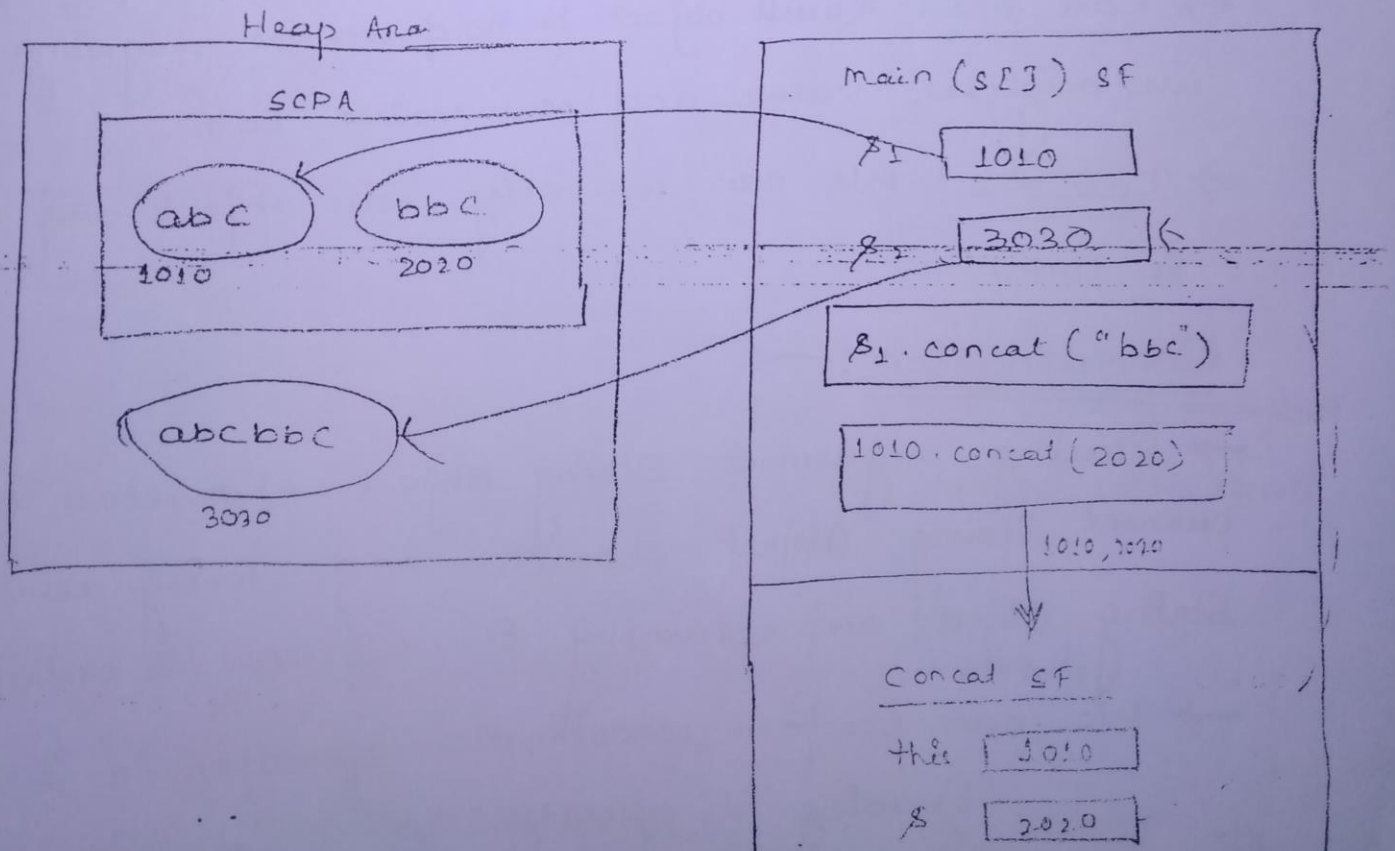


* In the above program 3 objects are created.

i) "abc" in p.SCPA.

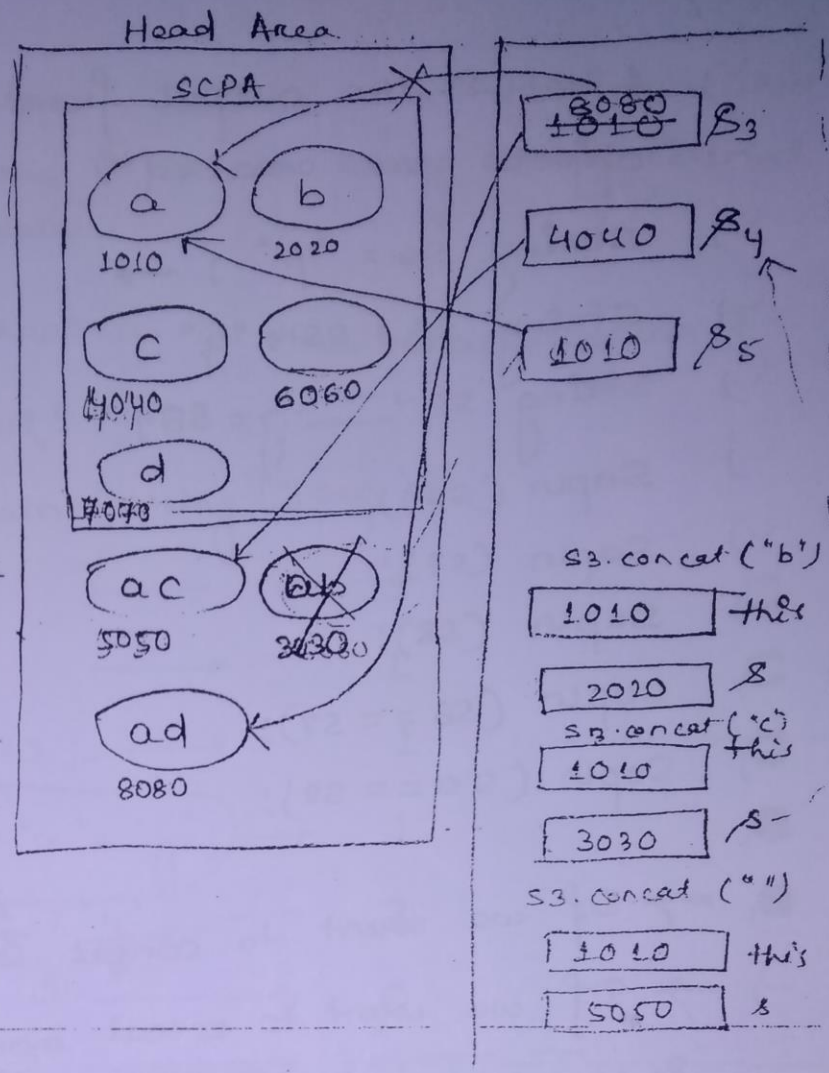
ii) argument String object "bbc" in SCPA.

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



Written Test Questions

```
String s3 = "a";
s3.concat("b");
Sopln(s3); → a
String s4 = s3.concat("c");
Sopln(s3); → a
Sopln(s4); → ac
Sopln(s3 == s4); // false
String s5 = s3.concat("");
Sopln(s3); → a
Sopln(s5); → a
Sopln(s3 == s5); → // true
s3 = s3.concat("d");
Sopln(s3); → ad
```



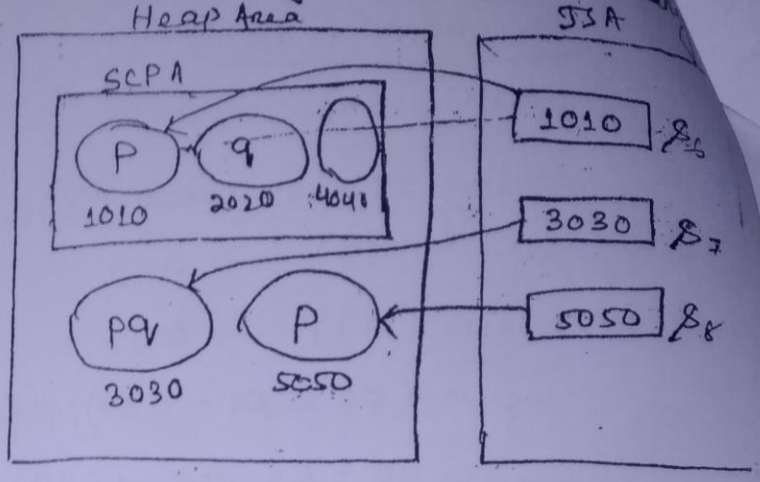
* Total 8 objects are created in the above program.
- Here 2 objects created b and ab.
- Here 2 objects created c and ac.
- Here 1 object created "".
- Here 2 objects created d and ad.

Concatination with '+' operator :=

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

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

```
String s6 = "p" ; →
String s7 = s6 + "q" ;
String s8 = s6 + "" ;
Sopln (s6);
Sopln (s7);
Sopln (s8);
Sopln (s6 == s7);
Sopln (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 it 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);
Sopln (s9);
Sopln (s10);
```

↳ Method call chaining

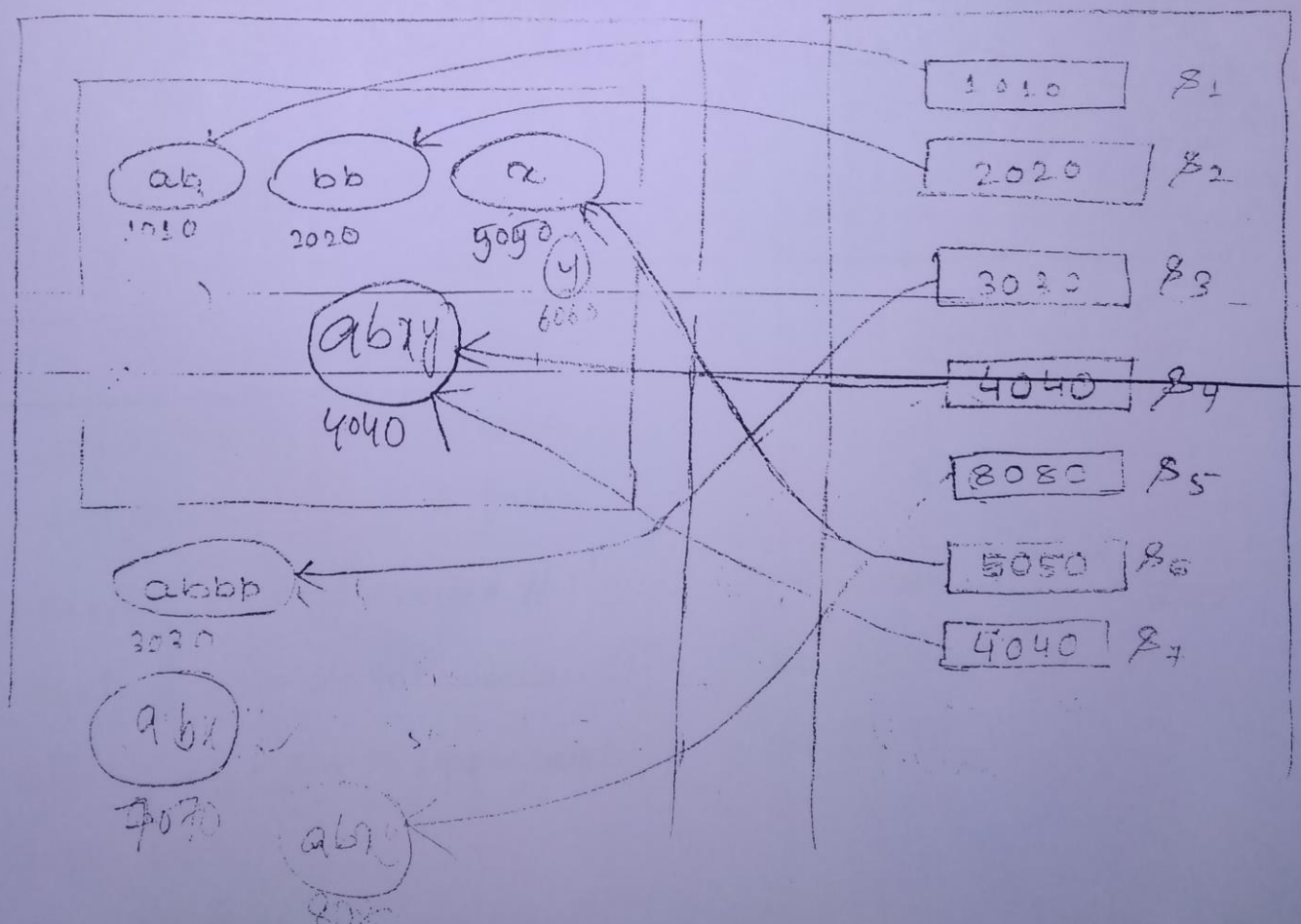
→ When we call a method with a method we should search for the called method in the first method return type class. And it is executed with first method returned object data.

→ 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 this result data.

→ 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 firstname : Hari

Enter Last name : Krishna

O/P - Hi HariKrishna

Changing case Of the String

// public String toLowerCase()

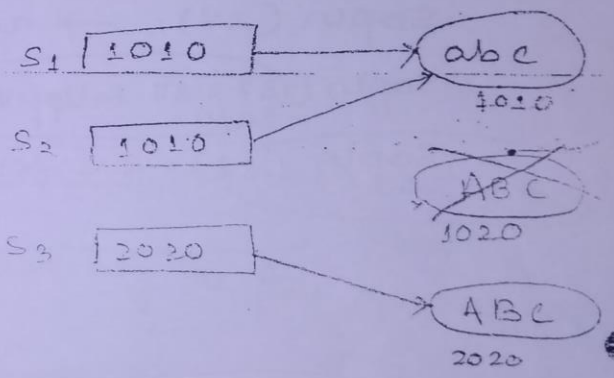
// public String toUpperCase()

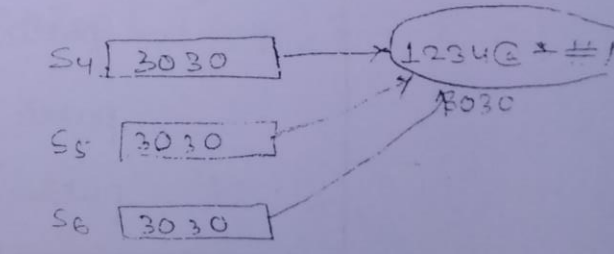
→ Above methods changing the case of alphabets in the current String.

→ 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.

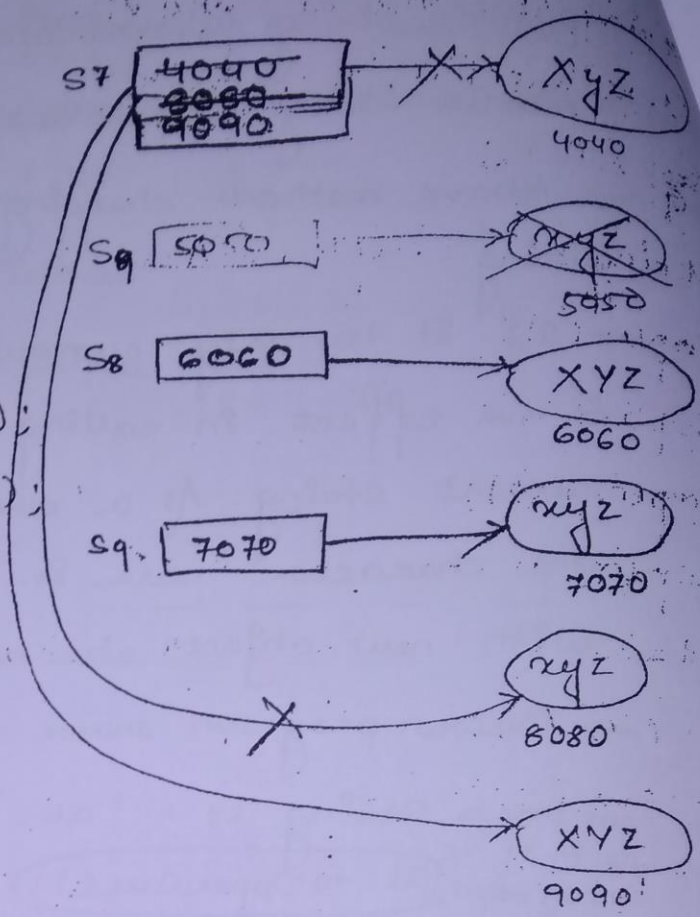
⇒ Below program shows the change in case of the String

```
String s1 = "abc";
s1 = s1.toUpperCase();
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();
Sopln(s4==s5); // true
```





```
Sopln(s5 == s6);  // true
String s7 = "XyZ";
Sopln();
Sopln(s7.toLowerCase());
Sopln(s7); → XyZ
String s8 = s7.toUpperCase();
String s9 = s7.toLowerCase();
Sopln(s7); → XyZ
Sopln(s8); → XYZ
Sopln(s9); → xyz
s7 = s7.toLowerCase();
Sopln(s7); → xyz
Sopln(s7 = s7.toUpperCase()); → XYZ
Sopln(s7); → XYZ
```



→ Here s7 pointing objects characters are changed to UpperCase and stores its 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 return 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 object) replace 'a' in s1 pointing

Sopln(s1); → abc abc abc

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

↳ java is case sensitive. So a ≠ A

Sopln();

Sopln(s1); → abc abc abc

Sopln(s2); → abc abc abc

sopln(s1==s2); // true

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

Sopln();

Sopln(s1); → abc abc abc

Sopln(s3); → cbc cbc cbc

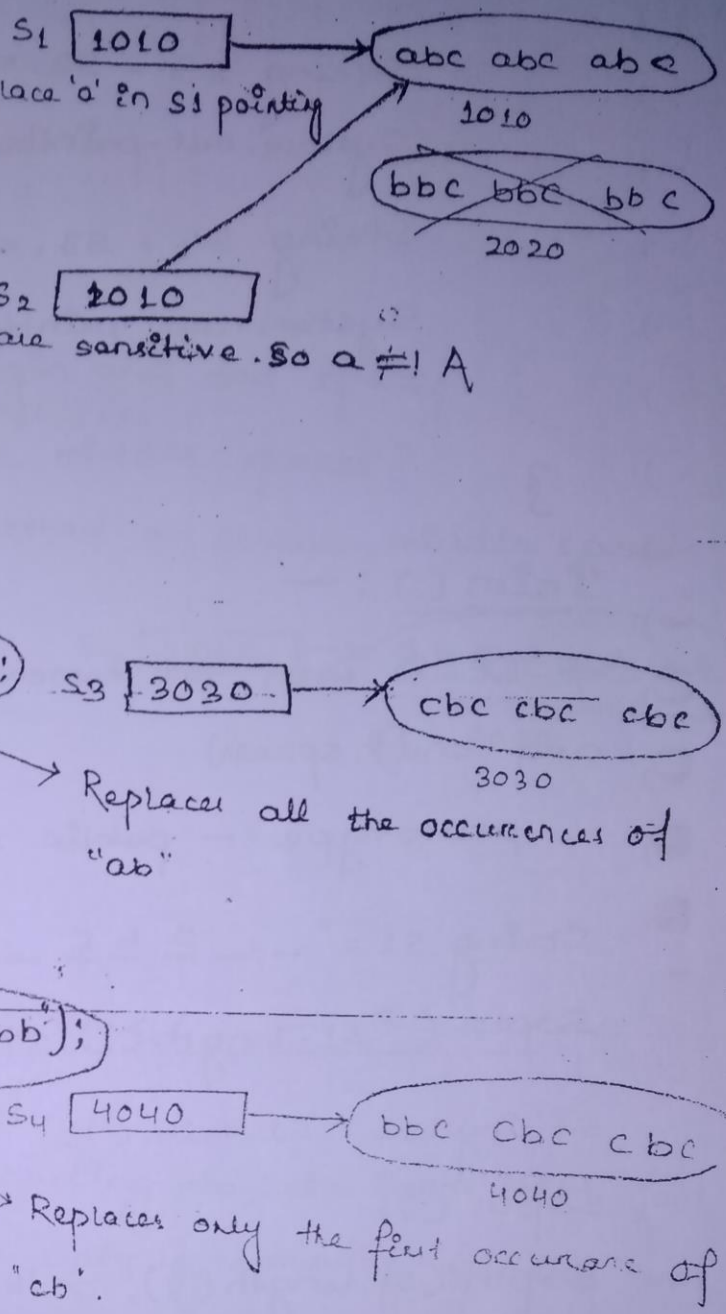
Sopln(s1==s3); // false

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

Sopln();

Sopln(s4); → bbc bbc bbc

Sopln(s3); → cbc cbc cbc



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);
```

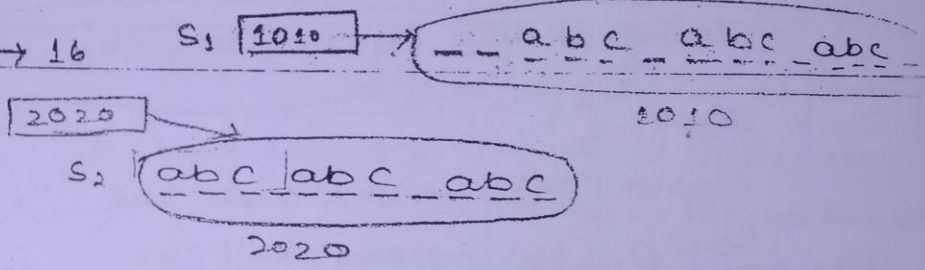
```
        String s3 = s2.replace('v', 'k');
        System.out.println(s3);
        String s4 = s3.replaceFirst("a", "e");
        System.out.println(s4);
    }
}
```

trim() :—

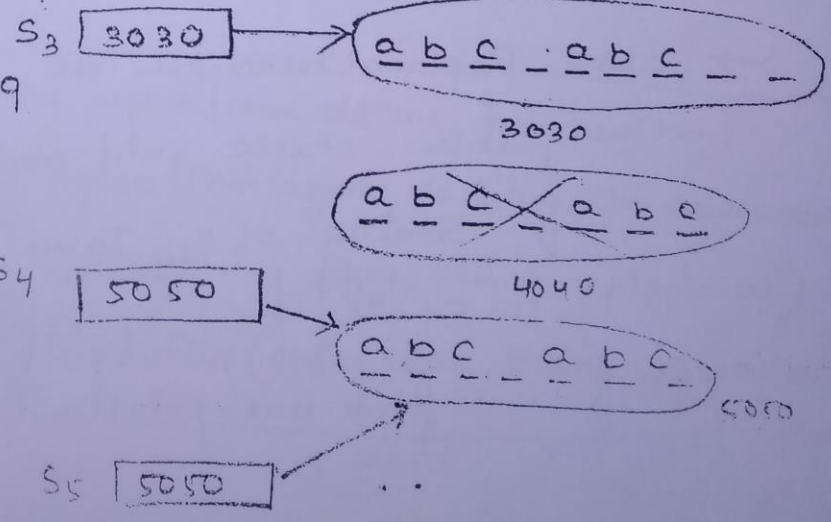
→ It is used for removing leading and trailing spaces. (beg and end spaces).

prototype :— public string trim().

```
String s1 = "  a b c  a b c  a b c  ";
Sopln(s1.length()); → 16
String s2 = s1.trim();
Sopln();
Sopln(s1.length()); → 16
Sopln(s2.length(); → 11
```




```
String s3 = "abc  a b c  ";
s3.trim();
Sopln(s3.length()); → 9
Sopln();
String s4 = s3.trim();
Sopln(s3.length()); → 9
Sopln(s4.length()); → 7
Sopln(s3==s4); // false
Sopln();
```



String s5 = s4.trim();

Sopln (s4.length); → 7

Sopln (s5.length); → 7

Sopln (s4 == s5); // true

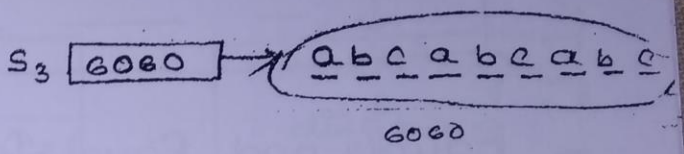
⇒ Trim() removes only begin and end space.

∴ 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

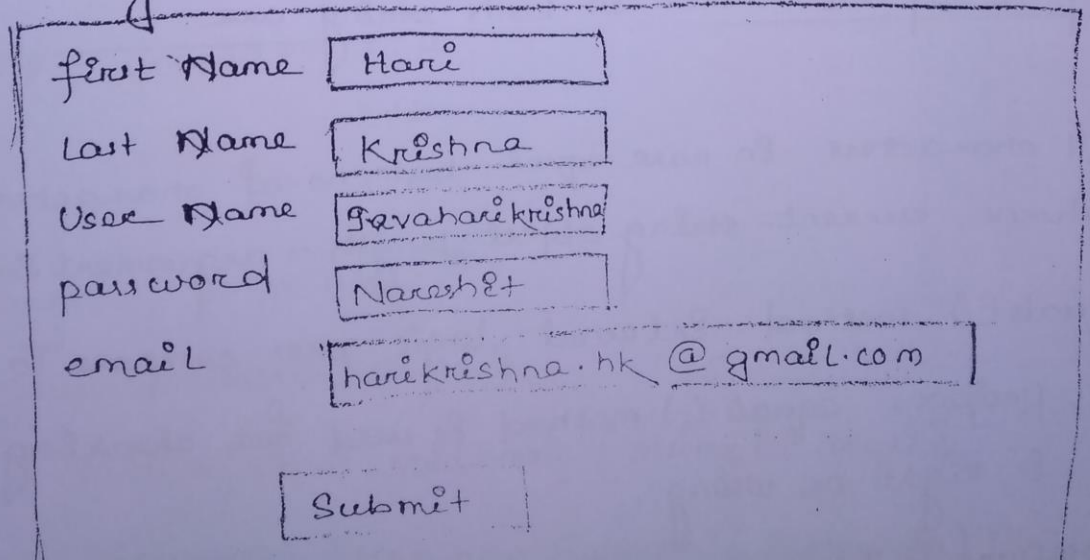
Sopln (s3.length()); → 9

Sopln (s4.length()); → 12

Importance of trim() methods:

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

Ex:= Reg.html

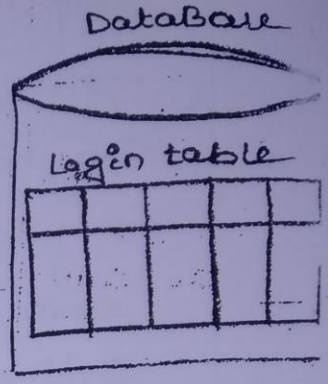


Servlet

```
String fname = req.getParameter("fn").trim();
String lname = req.getParameter("ln").trim();
String usn   = req.getParameter("usn").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 returns the ASCII number difference.

→ It continues this ~~substraction~~ comparion till non-zero ~~not~~ number occured.

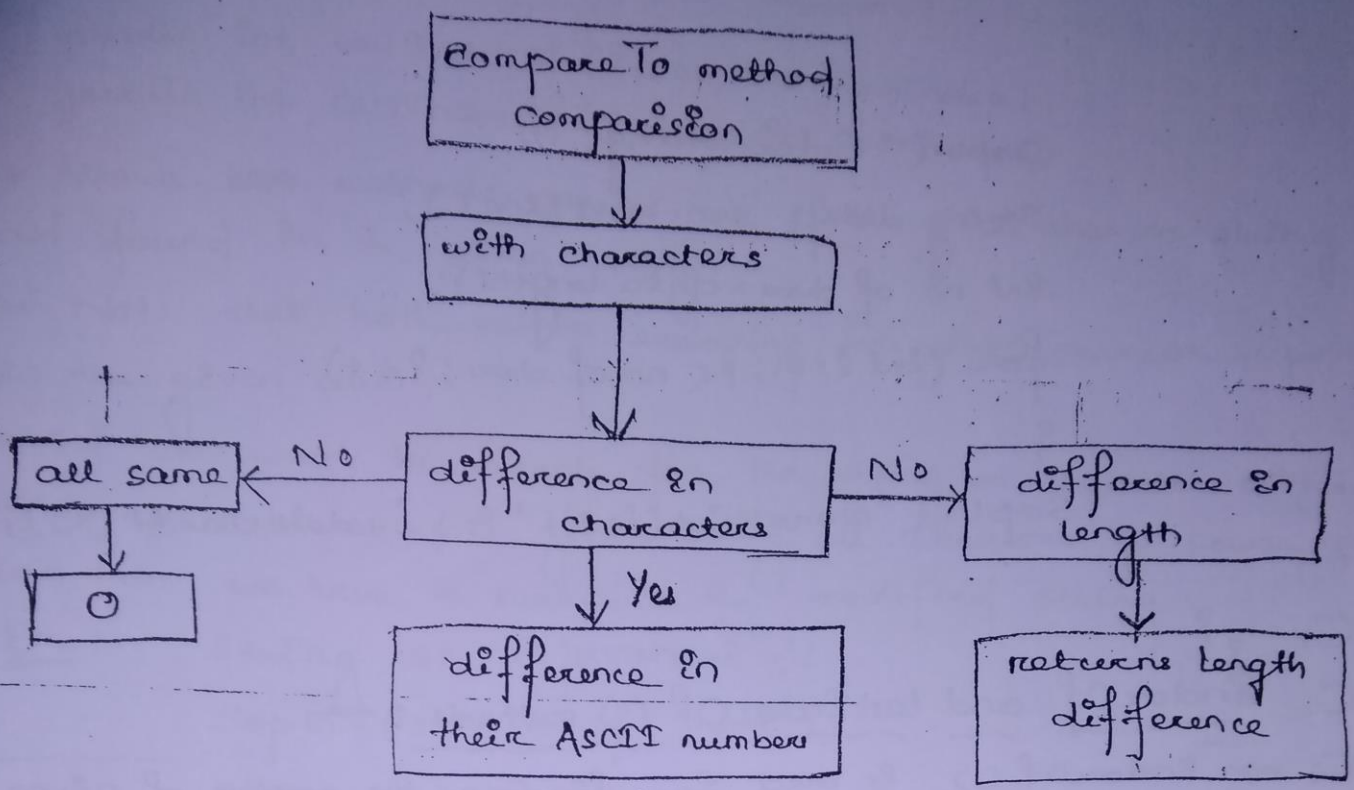
→ If no. of characters in both string are

→ 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. given input is right or wrong.

→ compareTo() method is used for sorting Strings in Array or collection.



Char charAt (Int index) :=

prototype : public string char 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 i/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);
```

```
        Sopln("Enter string: ");
        String data = scn.nextLine();
        Int no of chars = data.length();
        for (Int i=0 ; i< no of chars ; i++)
        {
        Sopln ("character" + (i+1) + " is : " + data.charAt [i]);
    }
}
```

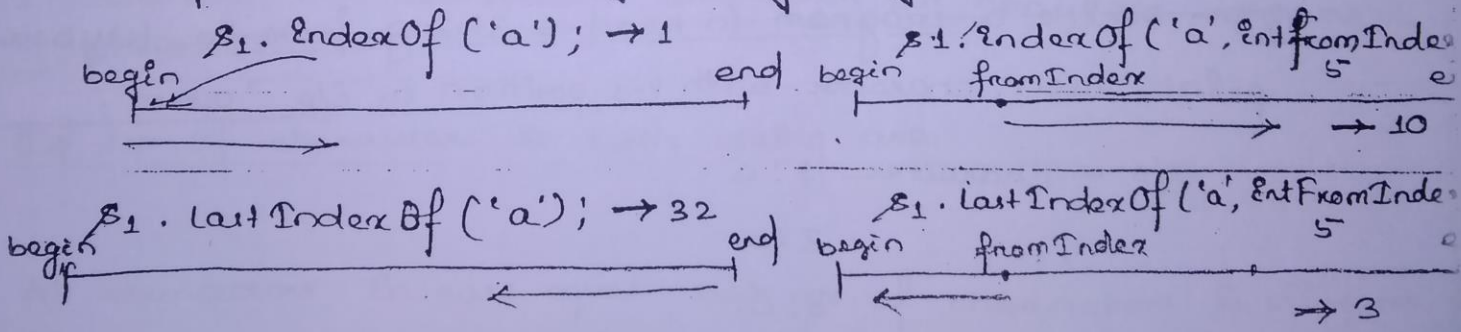
IndexOf and LastIndexOf() methods :—

→ IndexOf() is used for finding the postion of given characters of substring from begin or 0 index() and returns 1st occurance.

→ LastIndexOf() returns the last occurance 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 index.

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



prototypes :

```
public Int IndexOf (int ch)
public Int IndexOf (String s)
public Int LastIndexOf (int ch)
public Int LastIndexOf (String s)
public Int IndexOf (Int ch, Int fromIndex)
Public Int IndexOf (String s, Int fromIndex)
```

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

→ Above 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 s3 = "javaHari";
Sop(s2.indexOf("hari"));
String s3 = s2.toLowerCase
Sop(s3.indexOf("hari"));

→ Above methods 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 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);
    Sopln("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 5 SunMicroSystem given a special method called contains() in String class for finding substring or character in the current string with logic as we develop in searchIn() method.

prototype:

public boolean contains (CharSequence s)

→ In our searchIn method arguement & the actual String and substring we hard coded in this method.

→ In our contains method arguement & 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";

Sopln (s1.contains('a')); // false

Sopln (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[] filesArray = {"1.txt","2.doc","3.txt","4.pdf",
    "5.xls","6.txt"};
    for(int i=0; i<filesArray.length; i++){
        if(filesArray[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 arguement String character completely. It returns true if all are equal.

→ contains() method search for given String as substri

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"
Sopln(s1.substring(5)); // Program
Sopln(s1.substring(5,12)); // Program
Sopln(s1.substring(5,11)); // Progra
Sopln(s1.substring(5,5)); // ""
Sopln(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 a message "available".

→ ~~class~~

```
import java.util.*;
class TestSubString {
    public static void main (String[] args) {
        Scanner scn = new Scanner(System.in);
        while (true) {
            System.out.print("\n 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");
            }
            else {
                System.out.println("Hari is not available");
            }
        }
    }
```

Enter String : Javahari krishna
Hari is available.

Enter String : JavaHariKrishna
Hari is available

Enter String : Hari
Hari is available

Enter String : HaRiKrishna Java
HaRi is available

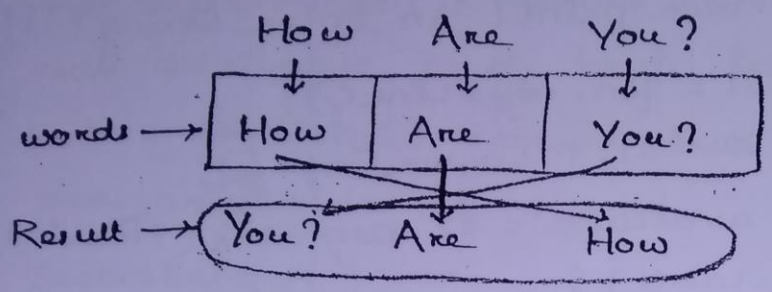
split() :-

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

i/p - How Are You?

o/p - You? Are How

A Procedure:




```
class StringReverser {
    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 ("/n 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 parameter int, long, float, double, char, char[], boolean, (char[], int, int), Object.

→ Byte and short parameter methods are not there because there is no literals subtype byte and short.

⇒ Ex:-

```
Sopln("data:"+10);
```

It 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

14/7/15

Prog 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' 'Welcome to NareshIT'
If spaces only entered also throw Error.

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

```
// isEmptyTest.java
import java.util.*;

public class IsEmptyTest{
    P S V m(S[] args){

        Scanner scn = new Scanner(System.in);
        while(true){
            Sopln("\n Enter name:");
            String name = scn.nextLine();
            if(name.isEmpty()){
                Sopln("Name is required");
                continue;
            }
            else{
                String resName = name.trim();
                if(resName.isEmpty(){
                    Sopln("Name is required");
                    continue;
                }
                Sopln("Hi" + resName);
                Sopln("Welcome to Naresh IT");
                break;
            }//while else
        }//while
    }
```

→ 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'

```
// LengthPasswordTest.java
import java.util.*;

public class LengthPasswordTest {
    P S V m(S[] args){
        Scanner scn = new Scanner(System.in);
        while(true){
        Sopln("Enter Password:");
        String pwd = scn.nextLine();
        if(pwd.isEmpty()){
            Sopln("Password should not be empty");
            continue;
        }
        else{
            int len = pwd.length();
            if(len<8 || len>16){
                Sopln("Password length should be 8 to 16");
                continue;
            }
            else{
                Sopln("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 are digits are not
3. ten digits exists or not 4. Is it really mobile number or not (send OTP sms)

→

```
// MobileValidator.java
import java.util.*;
public class MobileValidator{
    P S V m(S[] args){
        Scanner scn = new Scanner(System.in);
        while(true){
        Sopln("Enter Number:");
        String mobile = scn.nextLine();
         if(mobile.isEmpty()){
             Sopln("Mobile no. is Mandatory");
             continue;
         }
         else{
            ~~long mn = Long.parseLong(mob~~
            try{
                long mn = Long.parseLong(mobile);
                if(mobile.length()!=10){
                    Sopln("Mobile no. should be 10 digits");
                    continue;
                }
                else{
                    Sopln("Activation Key has send to your no.");
                    Sopln("Enter Activation Key:");
                    break;
                }
            Catch(NumberFormatException nfe){
                Sopln("Mobile no. should contain only digits");
                continue;
            }
        }
    }
}
```

Develop a prg to find no. of vowels & consonents available in the given string Read string from Keyboard.

→

// CheckVC.java

```
import java.util.*;
Public class checkVC {
    P S V m(S[] args){
      String s1

    P V checkVC {String s1,
        {
            this.String s1 = s1;


        }
        public tostring(){
            return
```

Mine Code

```
import java.util.*;
public class VowCons {
    P S V m (SC] args){
  Scanner scn = new Scanner (System.in);
    Sopln("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);
```

```
if(ch=='a'||ch=="e"||ch=="i"||ch=="o"||ch=="u"){
    vCount++;
}
else{
    cCount++;
}
}
SopIn("Available vowels are:"vCount);
SopIn("Available Consonents are:"cCount);
}
```

Develop a prg 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) MineCode

```
// LoginPage.java
import java.util.*;
public class LoginPage{
  P S V m(S[] args){
    Scanner scn = new Scanner(System.in);
    //SopIn("Enter Username:");
    //while(true){
    String usn = scn.nextLine();
    SopIn("Enter Password:");
    String pwd = scn.nextLine();
    //if(usn.isEmpty()){
        SopIn(
    //  Continue;
    //}
```

```
if (usn.isEmpty()) {
    sopln(
    continue;
}
else {
if (char ch >65 && ch<90
```

```
//Password PatternValidator.java
import java.util.scanner;
public class PasswordPatternValidator {
    P S V m(S[] args) {
        Scanner scn = new Scanner(System.in);
        boolean upperfound = false;
        boolean digitfound = false;
        boolean spCharfound = false;
        while(true){
            Sopln("Password:");
            String pwd = scn.nextLine();
            int len = pwd.length();
        if(len<8|| len>16){
            Sopln(" 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.isDigit(ch)){
```

// (or)
// if(Character.isLetter(ch)&&
// Character.isUpperCase(

```
            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~~

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'.

~~WAP~~

Rewrite above program to display ~~the~~ substring haris is available even though characters ~~WA~~ 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 @ & '.' characters, '.' 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{
    P S V m (S[] args){
        Scanner scn = new Scanner(System.in);
        Sopln("Enter String");
        String s1 = scn.nextLine();
        String s2 = "Hari";
        for(int i=0; i<s1.length(); i++){
            char ch = s1.~~length();~~
                      charAt(i);
        public int indexOf(String s2){
            sopln("substring Hari available");
        }
    } else {
        Sopln("substring Hari not available");
    }
}
```

Rewrite above program to display substring hari is available even though characters h,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);
        Sopln("Enter a String:");
        String s1 = scn.nextLine();
        for(int i=0; i<s1.length(); i++) {
            char ch = s1.charAt(i);
            if (ch=='h' && ch=='a' && ch=='r' && ch=='i') {
                Sopln("Available");
            }
            else {
                Sopln("Not Available");
            }
        }
    }
```

if (s1.indexOf("hari")!=-1 && s1.indexOf('h')!=-1

Email Validator programe :- (Q in back page)

```
// EmailValidator.java
import java.util.*;

public class EmailValidator {

    P  S  V  m (S[] args) {

        Scanner scn = new Scanner(System.in);

        Sopln("Email:");
        String email = scn.nextline();
        int atIndex;
        if((( atIndex = email.indexOf('@')) != -1) &&
            (email.indexOf('.', atIndex) != -1)) {
            Registered
            Sopln();
        }
        else {
            Sopln("Invalid email");
        }
```

26/7/15

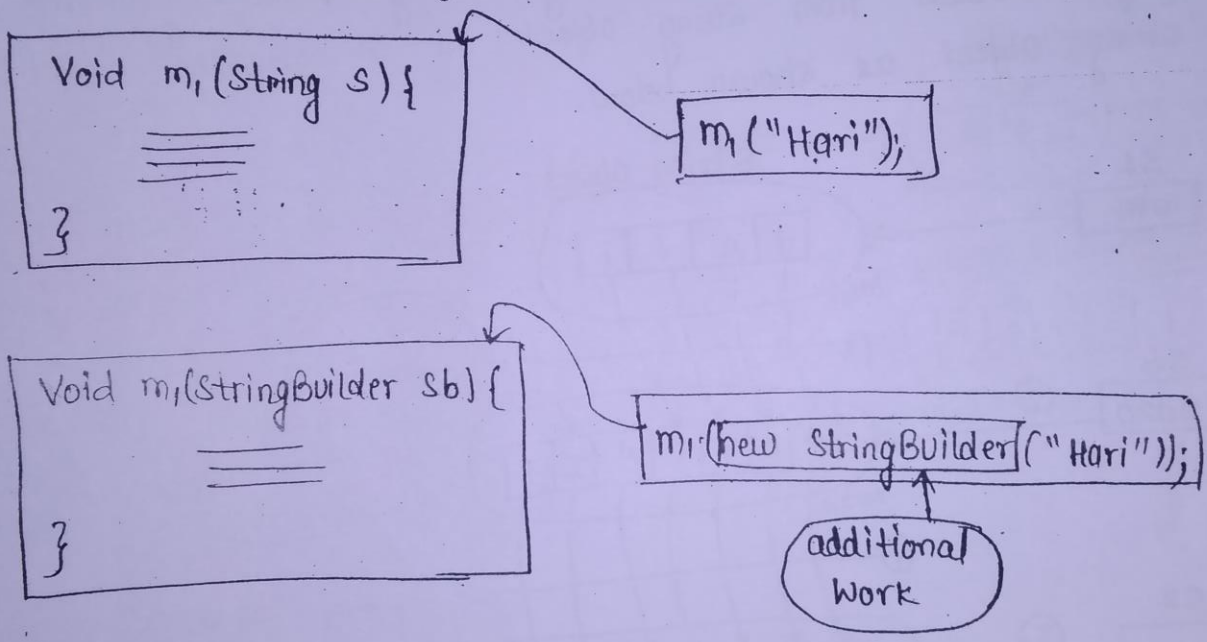
What is the right parameter for taking string data as argument, returing string data from a method?

→ 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 ~~keyboard~~ Keyword constructor explicitly. It increases more lines of code burden to method call.

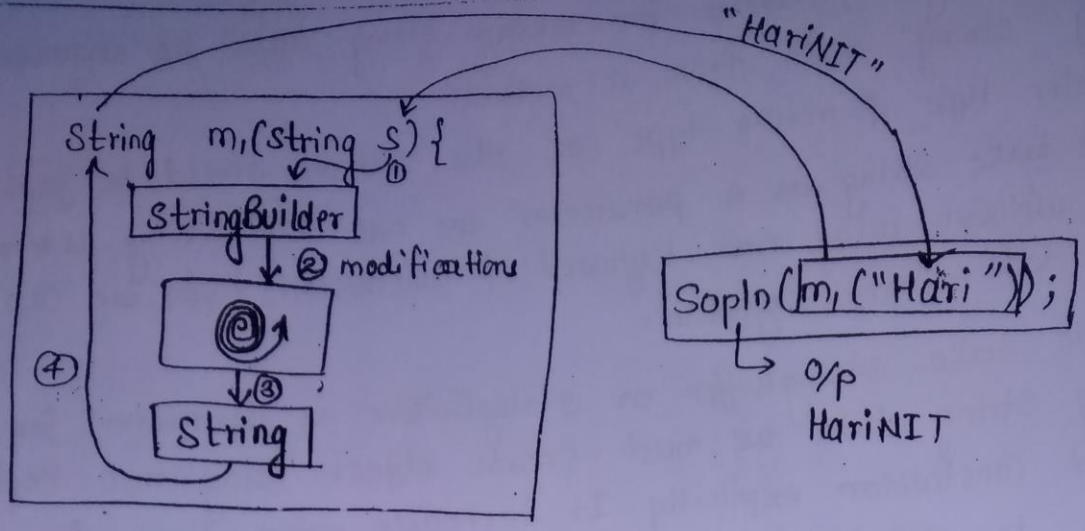
→ Below code shows creating a method with string & StringBuilder as the parameter & calling them.



→ 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.

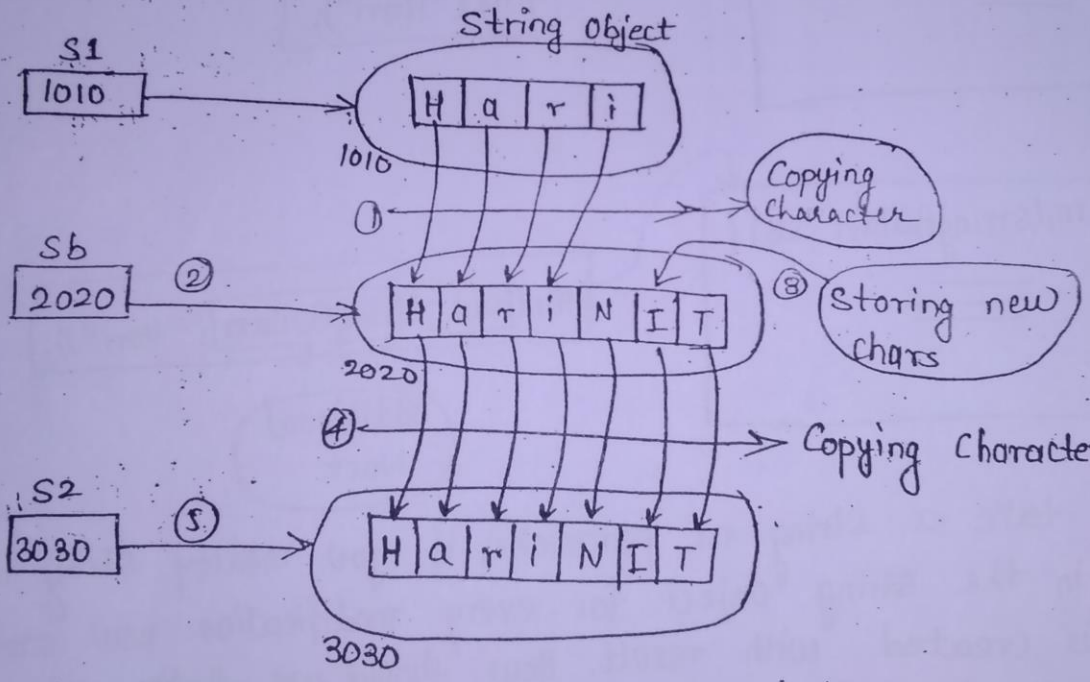
→ Do reqd. modification on StringBuilder object.

→ 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 must use to convert String to StringBuilder & viceversa?

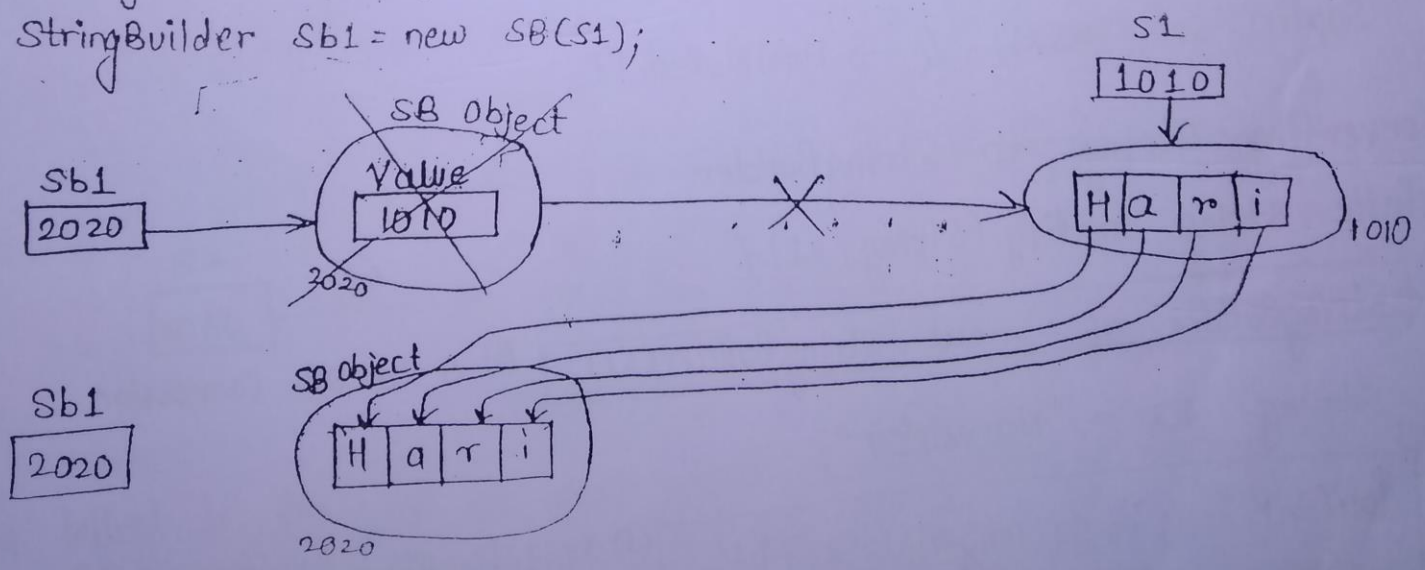
→ In String class & StringBuilder class we have constructor for copying characters from string object to stringBuilder object & StringBuilder object to String object.

```
class String {
    String (StringBuilder sb) { }
    String (StringBuffer sb) { }
    ≡≡≡
}
```

```
class StringBuilder {
    StringBuilder (String s) { }
}
```

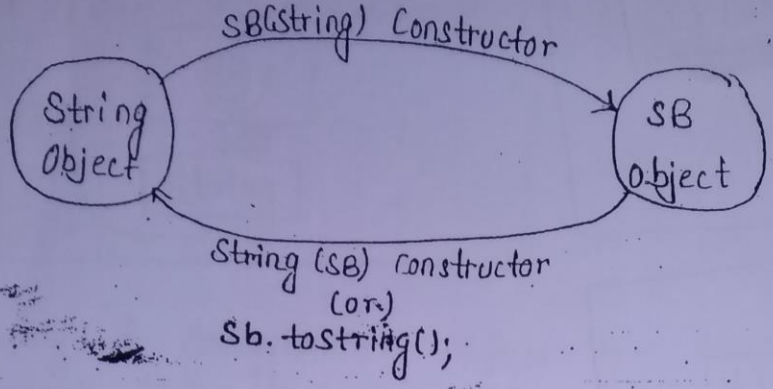
→ The diff bet^n normal class constructors to String & StringBuilder constructors in normal class constructors stores given argument data in current object whereas String, StringBuilder class constructors will copy given argument object characters into current object. They will not store arguments as 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 {

```
P S V m(S[] args) {
    String S1 = "Hari";
    String S2 = modify(S1);
    Sop ln("S1: " + S1);  // → Hari
    Sop ln("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
} ⑤ Returning String
```