

String

String:

- String is a Built in class in java.lang package.
- String is a ~~final~~ final class, so you can't define the subclass for String class.
- String class implements the following interfaces
 - java.io.Serializable
 - java.lang.Comparable
 - java.lang.CharSequence
- String class has following variable to hold data

```
private final char value[];
```
- String in generally is a sequence of characters, But in java, String is an object that represents a sequence of characters.
- String objects are immutable objects. It means once the object is created then the content or data of the object can't be modified.
- If you try to modify the content of object then new object will be created as a result
- we can create the object of String in two ways
 - ① without using new operator
 - ② using new operator.

Without using new operator

JVM allocates the memory for the reference variable

Syntax:

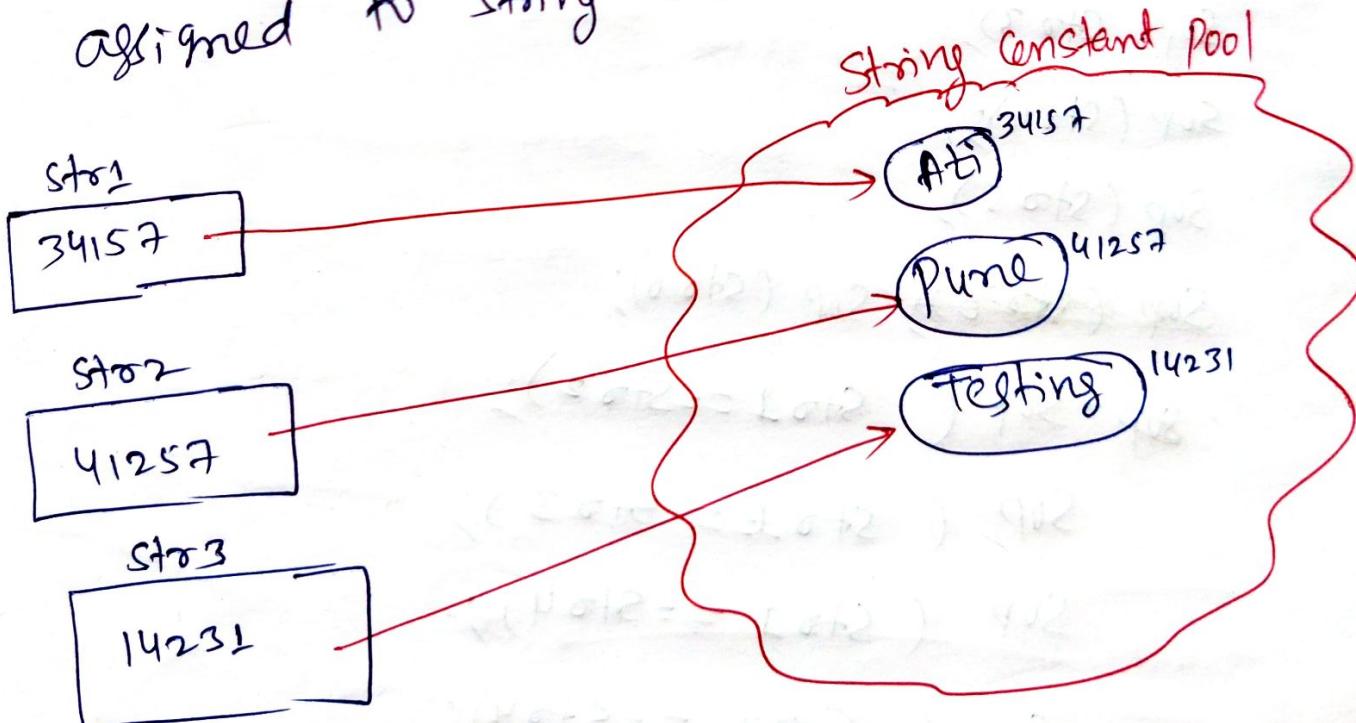
String str1 = "Ati";

String str2 = "Pune";

String str3 = "Testing";

JVM verifies the string literal in the String Constant Pool (SCP).

If string literal is not available in the SCP then it creates new string object inside the SCP and newly created string object address will be assigned to string reference variable.



Object is created in SCP and address will be assigned to string reference variables.

class StringWithoutNewExample {

psvm(-)

{

String str1 = "Ati";

String str2 = " pune";

String str3 = str1;

String str4 = str2;

String str5 = "Testing";

String str6 = str1 + str2;

SOP(str1);

SOP(str2);

SOP(str3);

SOP(str4);

SOP(str5);

~~SOP(str6)~~, SOP(str6);

SOP SOP (str1 == str2);

SOP (str1 == str3);

SOP (str1 == str4);

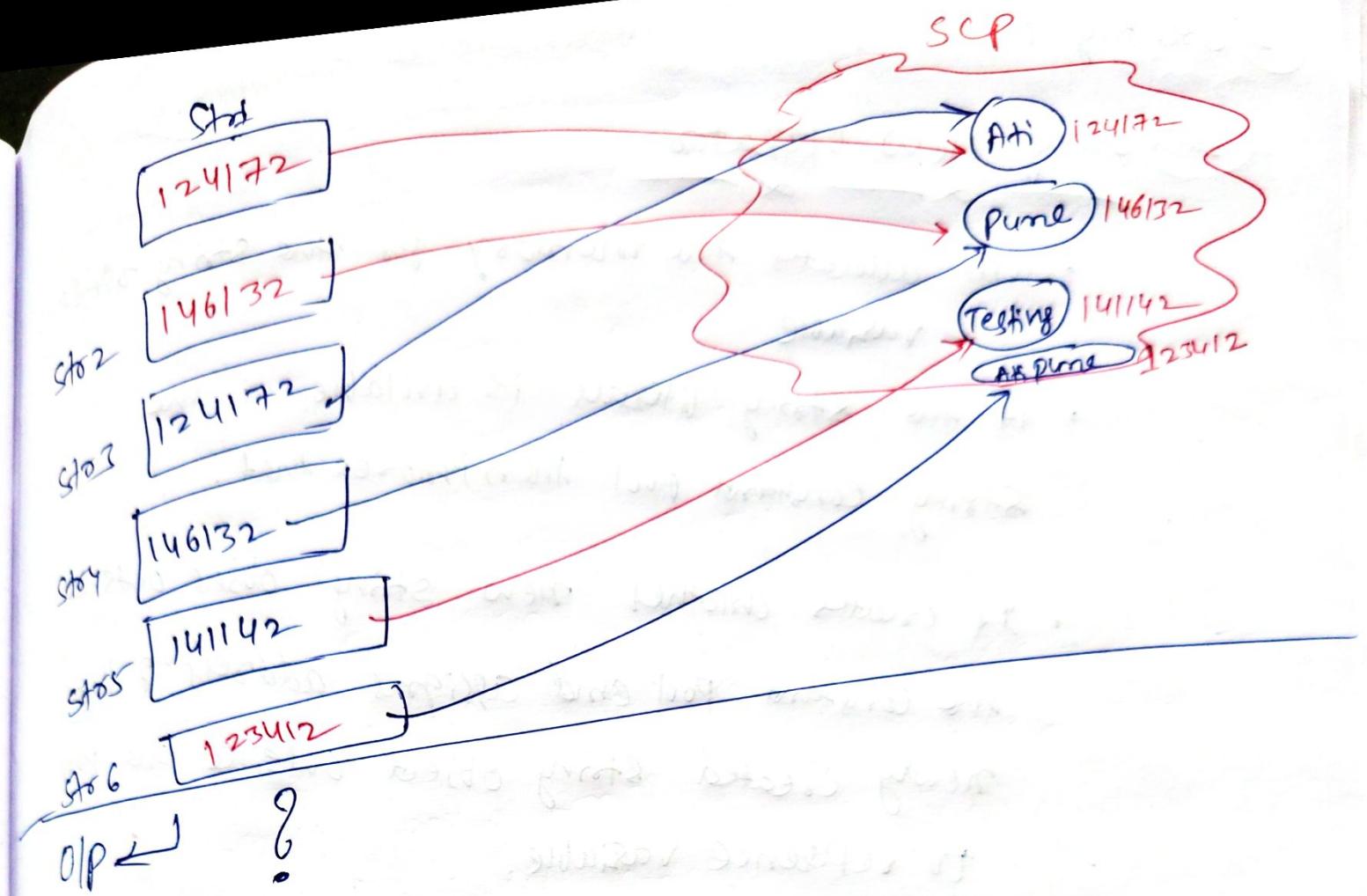
SOP (str2 == str4);

SOP (str5 == str3);

SOP (str3 == str4);

}

3



Creating

② Using new operator

- jvm allocates the memory for the string reference variable
- if the string literals is available in the string constant pool then ignores that.
- It creates another new string object outside the constant pool and assigns address of the newly created string object outside the pool to reference variable.

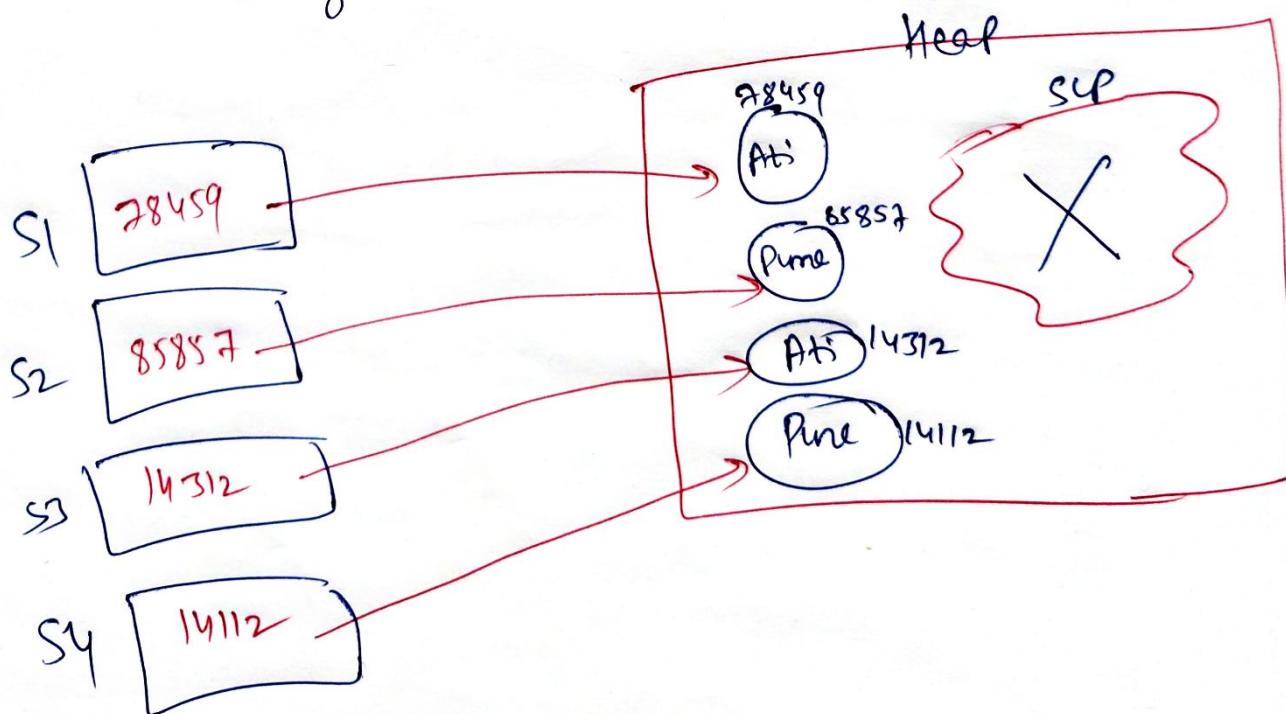
Syntax:

String s1 = new String ("Ati");

String s2 = new String ("Pune");

String s3 = new String ("Ati");

String s4 = new String ("Pune");



Runtine objects that are created are stored in heap area only.
By using Heap object if you want to get corresponding
SCP object reference then we can use intern() method.
If object not presents, it will create and returns.
SCP objects will be destroyed at the time of
JVM shutdown.

String new Example :-

```
public class StringNewExample {
    public void main() {
        String s1 = new String ("Ati"); // Heap
        String s2 = new String ("pune"); // Heap
        String s3 = new String ("Ati"); // Heap
        String s4 = new String ("pune"); // Heap
        String s5 = "Ati"; // SCP
        String s6 = "pune"; // SCP
        String s7 = s4.concat("Testing"); // Heap/SCP/Runtine?
        s5.concat("Testing"); // Heap - Runtine
        s1.concat("Testing"); // Heap - Runtine
        System.out.println(s1 == s3); // SOP (s1 == s3);
        System.out.println(s2 == s4); // SOP (s2 == s4);
        System.out.println(s1 == s5); // SOP (s1.equals(s5));
        System.out.println(s2 == s6); // SOP (s2.equals(s6));
        System.out.println(s3 == s5); // SOP (s3.equals(s5));
        System.out.println(s4 == s6); // SOP (s4.equals(s6));
    }
}
```

String Pool / Heap diagram

P
pu
put
Pu
get
pw

- ①
- ②
- ③ Pu
- ④ Pu
- ⑤ Pu
- ⑥ P
- ⑦ P

String class
Constructors : →
 public String(); string with empty content
 public String (String str); string with specified string content
 public String (StringBuffer); string with specified StringBuffer
 public String (StringBuilder); string with specified StringBuilder
 public String (char[] arr); string with specified array of chars
 public String (byte[] arr); string with specified array of ~~arr~~ bytes.

Methods of String class

Name of method

- ① public int length();
- ② public boolean isEmpty();
- ③ public String concat (String);
- ④ public String toLowerCase();
- ⑤ public String toUpperCase();
- ⑥ public String trim();
- ⑦ public String toString();

Description

- returns length of string
- returns true if length is 0.
- concatenates the string
- convert to lowercase
- convert to uppercase.
- trim whitespace from beginning and end of a string.
- returns the current object content.

- ⑧ Public char charAt (int index); Returns char at specified index.
- ⑨ Public boolean equals (Object); compares content of this string to specified object. case of the char also be refied.
- ⑩ Public boolean equalsIgnoreCase (String); compares content of this string to specified object. case will be ignored.
- ⑪ Public boolean startsWith (String p); checks string starts with p
- ⑫ Public boolean endsWith (String s); checks string ends with s
- ⑬ Public int indexOf (int ch); Returns the index of 1st occurrence of specified char.
- ⑭ Public int indexOf (String st); Returns the index of 1st occurrence of specified string
- ⑮ Public int lastIndexOf (int ch); } Returns the index of last occurrence of char.
- ⑯ Public int lastIndexOf (String st); } Returns the index of last occurrence of char and string.
- ⑰ Public String substring (int beginIndex); returns a substring starting from beginIndex to end of the original string.
- ⑱ Public String substring (int beginIndex, int endIndex); returns a substring from original string from beginIndex to (endIndex-1).

- (19) public string replace (char oldchar, char newchar)
replace all occurrence of oldchar with newchar.
- (20) public string replaceAll (string regex, string newstring)
replaces all occurrence of matching string with
newstring.
- (21) public boolean contains (char sequence);
Compares content of this string to the
specified charsequence.
- (22) public string[] split (string regex)
splits this string when the given
regular expression matches.
- (23) public int hashCode()
returns set of string
- (24) public native string intern(): object from constant pool.
- (25) public int compareTo (String)
public int compareToIgnoreCase (String)
Compares content of this string to the
specified object. (case/without case.)

String Buffer

- `StringBuffer` is a final class in `java.lang` package.
- it is used to store the sequence of characters.
- `StringBuffer` is a mutable sequence of characters.
It means the content of the `StringBuffer` can be modified after creation.
- Every `StringBuffer` object has a capacity associated with it. (number of characters it can hold).
- The methods available in the `StringBuffer` class are synchronized and thread safe.
- default capacity of `StringBuffer` is 16.
- implements `Serializable`, `CharSequence`, `Appendable`.
- we can't allocate memory with literals (no `sc.nextInt()` concept)

Constructor:

- ① `StringBuffer()` `sb = new StringBuffer()`
- ② `StringBuffer` `sb1 = new StringBuffer (String str)`
- ③ `StringBuffer` `sb2 = new StringBuffer (int capacity)`

StringBuffer with string value, the capacity will be

$$\text{capacity} = 16 + sb1.length();$$

StringBuffer with empty string value, the capacity will be

$$(\text{Capacity} = 16), \text{next capacity} = (16 \times 2) + 2 = 34.$$

Methods of StringBuffer

- ① `StringBuffer append (String s)` : used to append the specified string with this string.
overloaded method it can take char, int, boolean, float, double.
- ② `StringBuffer insert (int offset, String s)`
used to insert the specified string with this string at specified position. also overloaded.
- ③ `StringBuffer replace (start, endindex, String str);`
④ `StringBuffer delete (startindex, endindex);`
- ⑤ `StringBuffer reverse ();`
- ⑥ `int capacity ()` : returns the capacity
- ⑦ `void ensureCapacity (int minCapacity)`
it is used to ensure the capacity at least equal to the given minimum capacity.
- ⑧ `public char charAt (int index)` returns character at specified position.
- ⑨ `int length ()` returns the length of string (total number of characters).
- ⑩ `String substring (int beginIndex)`
- ⑪ `String substring (beginIndex, endIndex)`
- ⑫ `int indexOf (String str)` (last index of (String str))
⑬

(14) void SetLength (int len)

This method sets the length of character sequence.

(15) void trimToSize()

This method attempts to reduce storage used for character sequence.

- Release the extra allocated free memory.
- Length and capacity will be same.

Example Programs -

class StringBuffer Example {

PSVM (-)

{

StringBuffer sb = new StringBuffer ("Test");

StringBuffer sb1 = new StringBuffer ("Automatic");

StringBuffer sb2 = sb.append ("Framework");

StringBuffer sb3 = new StringBuffer ("Test");

~~StringBuffer sb3 = sb.append ("Test");~~

SOP (sb1 == sb);

SOP (sb == sb3);

SOP (sb2);

SOP (sb3);

SOP (sb4);

3