* 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[];

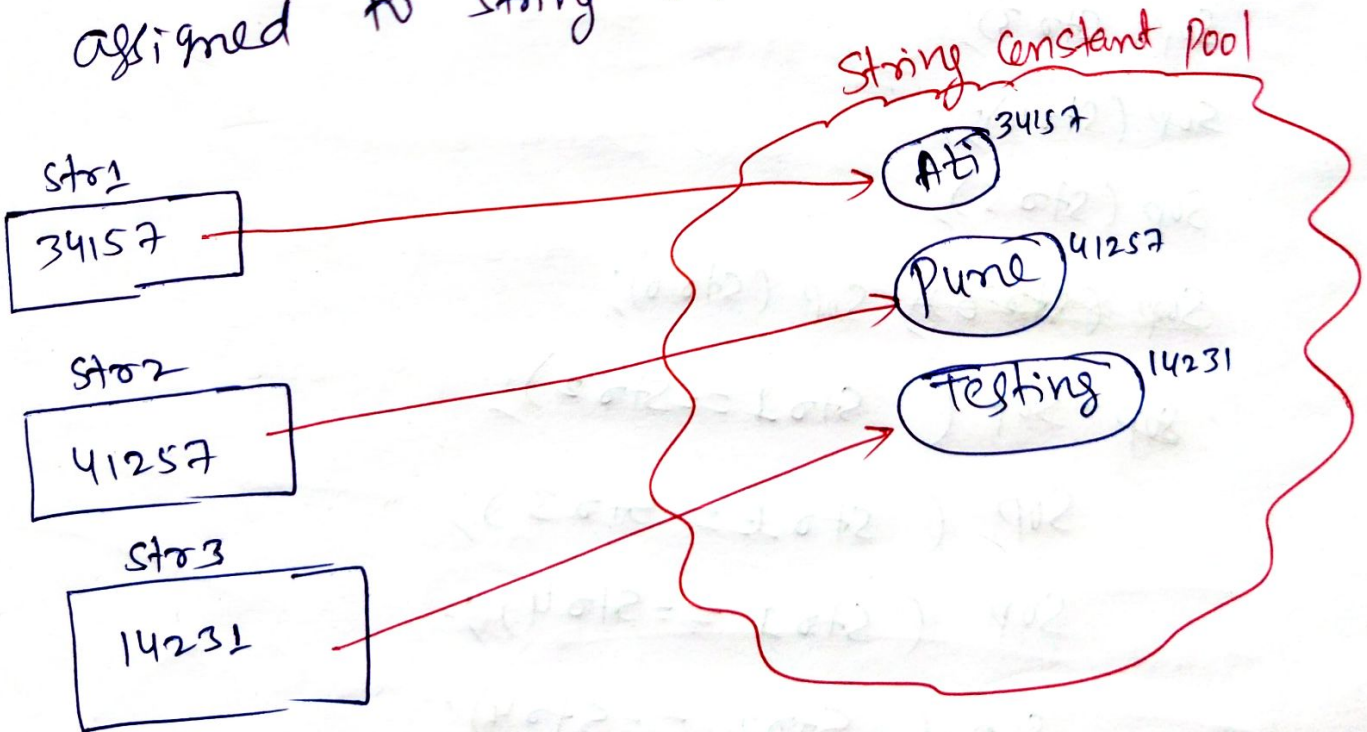
- String in generally is a sequence of characters., But in java, String is an object that represents a sequence of characters.
- String objects are immutable objects. It means once the objects is created then the content or data of the objects can't be modified.
- If you try to modify the content of 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.

1) Without using new operator →

- Jvm allocates the memory for the reference variable
- Syntax:

```
String str1 = "Ati";
String str2 = "Pune";
String str3 = "Testing";
```

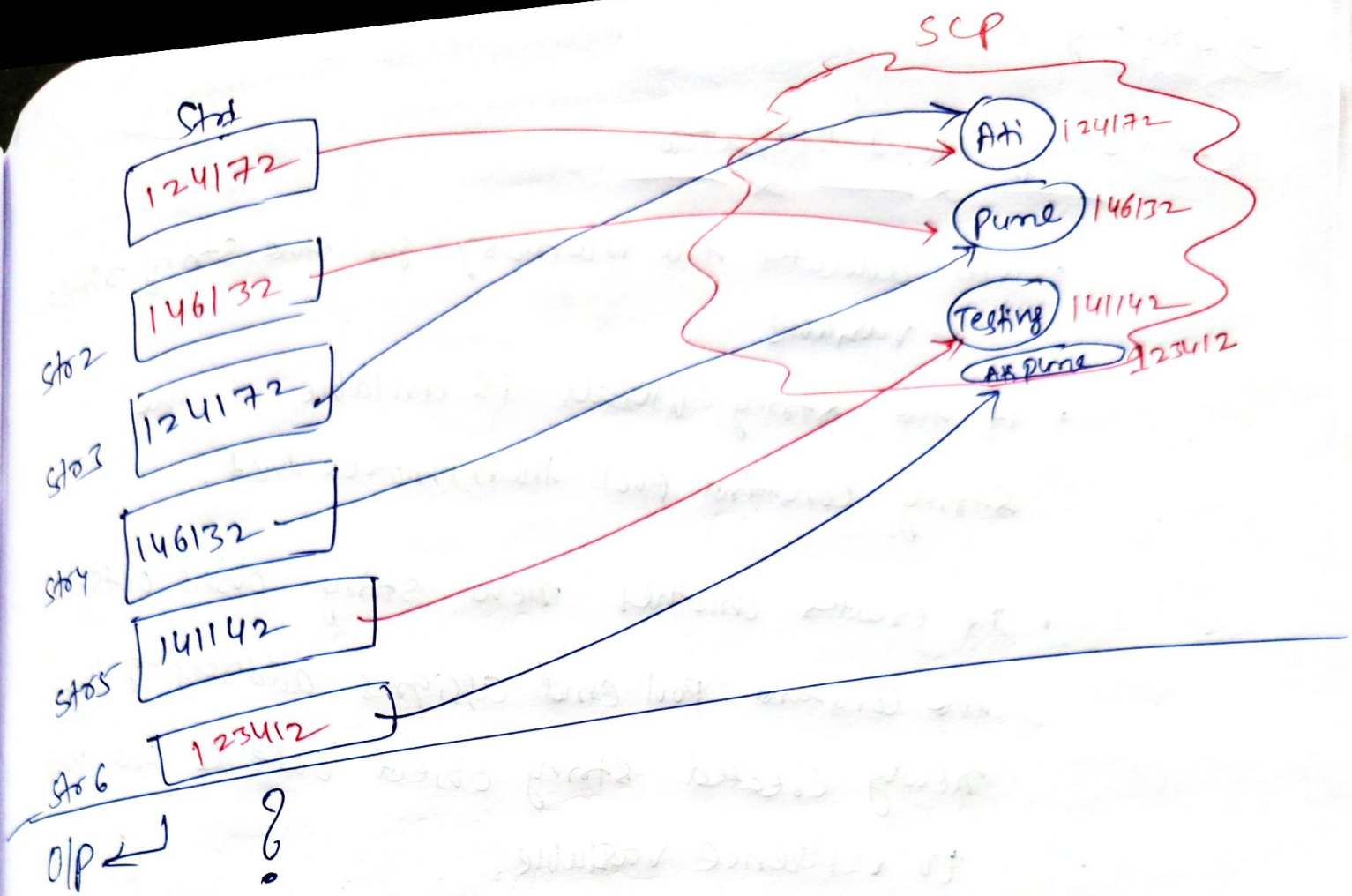
- Jvm verifies the string literal in the string constant Pool (SCP).
- If String literal is not available in the SCP then it creates new string object inside the SCP and newely created string object address will be assigned to string reference variable.



- 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(str1==str2);
    sop(str1==str3);
    sop(str1==str4);
    sop(str2==str4);
    sop(str5==str3);
    sop(str3==str4);
    }
}
```


SCP
Std
124172
Std2
146132
Std3
124172
Std4
146132
Std5
141142
Std6
123412
Ati
124172
Pune
146132
Testing
141142
Aspune
123412
O/P
?

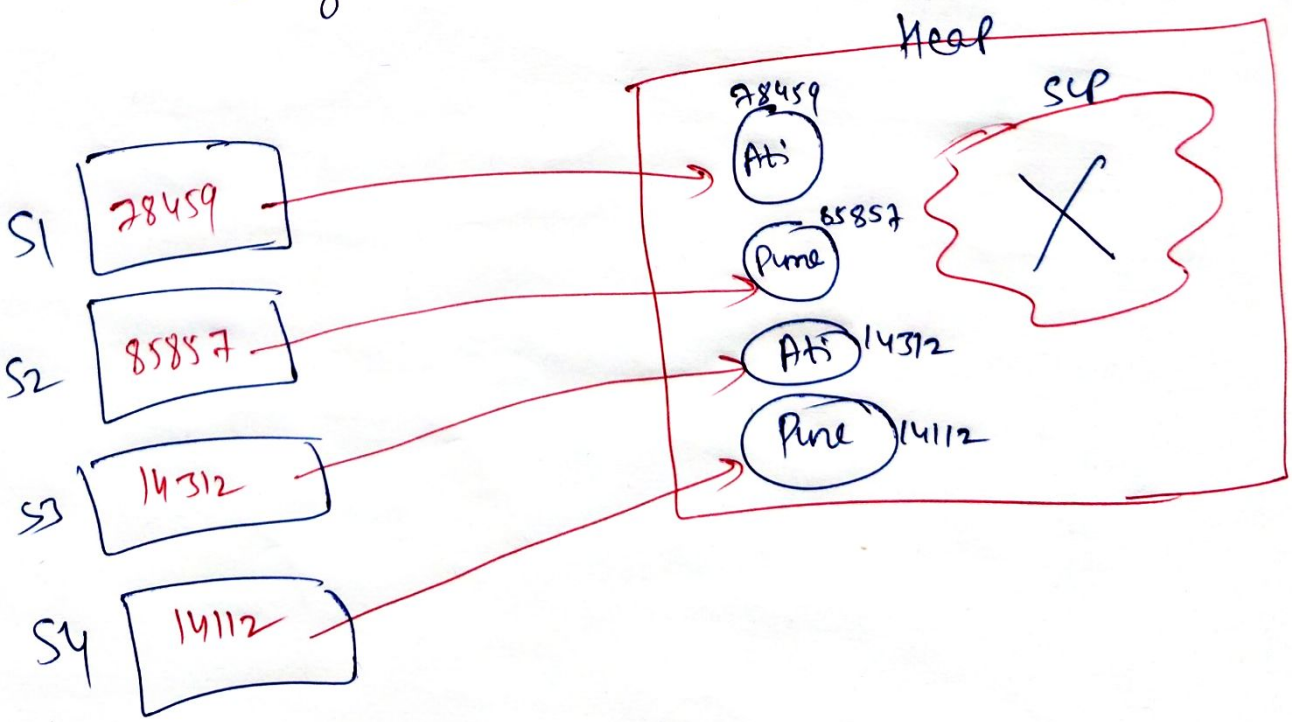
~~2) Creating~~ 0

(2) Using new operator

- jvm allocates the memory for the string reference variable
- if the string literals is avilable 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");
```



Runtime 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.

```
public class StringNewExample {

psvm(-)
{
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
s5.concate("testing"); // Heap/Scp/Runtime ? Heap
s1.concate("Testing"); // Heap - Runtime
String s7 = s4.concate("Testing"); // Heap - Runtime

sop(s1 == s3);
sop(s2 == s4);
sop(s1 == s5);
sop(s2 == s6);
sop(s3 == s5);

sop(s4 == s6);
sop(s1.equals(s3));
sop(s2.equals(s4));
sop(s1.equals(s5));
sop(s3.equals(s6));
}}
```

String Pool / Heap diagram

String class

Constructors :→

Public String(); String with the 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 bytes.

Methods of String class

Name of method	Description
1) Public int length();	returns length of string
2) Public boolean isEmpty();	returns true iff length is 0.
3) Public String concat (String);	Concatenates the string
4) Public String toLowerCase();	convert to lowercase
5) Public String toUppercase();	convert to uppercase.
6) Public String trim();	trim whitespace from beginning and end of a string.
7) Public String toString();	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 rectified.

(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 occurance of specified char.

(14) Public int indexOf (string st): Returns the index of 1st occurance of specified string

(15) Public int lastIndexOf (int ch);

(16) Public int lastIndexOf (string st); } Returns the index of last occurance 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 ochar, char nchar)
replace all occurance of ochar with nchar.

(20) public string replaceAll (string regEx, string newstring)
replaces all occurance of matching string with newstring.

(21) Public boolean contains (charsequence);
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()

(24) Public native String intern(); returns ref of string 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.)

* StringBuffer *

- 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 SCP concept))

Constructors:

(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

capacity = 16 + sb1.length();

- string Buffer with empty string value, the capacity will be

capacity = 16, next capacity = (16 x 2) + 2 = 34.

70 → 142

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 char 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) (13) lastIndexOf(String str)

(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 Program:

```
class StringBufferExample {

    psvm(-)
    {
        StringBuffer sb= new StringBuffer("Test");
        StringBuffer sb1 = new StringBuffer("Automation");
        StringBuffer sb2 = sb.append("framework");
        StringBuffer sb3 = new StringBuffer("Test");
        StringBuffer sb4 = sb1.append();
        sop (sb1 == sb);
        sop (sb == sb3);
        sop (sb2);
        sop (sb3);
        sop (sb4);
    }
}
```

StringBuilder

- StringBuilder is a final class in java.lang package
- StringBuilder class functionality is same as StringBuffer only except the method available in the stringBuilder class are non synchronized.
- StringBuilder is not thread-safe.
  (multiple threads can access it simultaneously).
- NO SCP concept
- all the methods from StringBuffer are applicable in StringBuilder, with return type of StringBuilder.
- hashcode() and equals() methods are not overridden in StringBuffer and StringBuilder.
- StringBuilder sb = new StringBuilder();
  StringBuilder sb1 = new StringBuilder("abc");
  StringBuilder sb2 = new StringBuilder(25);
- if we want to compare content of StringBuffer or StringBuilder Object then you need to convert the object of Builder and Buffer into String.

  Ex: sb.toString();

Difference Between String and StringBuffer

String	StringBuffer
(1) immutable	mutable
(2) having methods concat() and trim()	having methods append() and trimToSize()
(3) used when content not frequently change	frequently content changes.
(4) Scp concept	No Scp concept..
(5) No capacity concept	capacity concept
(6) Objects created using new and literals	objects are created only using new
(7) No thread safe	Thread safe.
(8) equals method is overridden & hashcode	equals and hashcode method is not overridden.
(9) implementing java.lang.Comparable	— No —
(10) Concatination operator (+) can be used	Concatination operator (+) can't be used.

String	StringBuilder
1) Synchronized	StringBuilder not Synchronized
2) Thread safe	not thread safe
3) Java 1.0	Java 1.5
4) low Performance	High Performance.
5) capacity concept not applicable	capacity concept applicable.
6) SCP concept	No SCP concept
7) equals & hashcode are overriden	— No —
8) objects are created using new and literal	new using