String Buffer

As we discuss earlier that String class is Immutable but using the concept of String buffer we make it mutable class.

Interface

1. Implementing polymorphism in different ways.
2. Supporting of multiple Inheritance
3. Interface in Java is similar to the class because it also contain data members and method

By Only diffrence is data member is final and method is only ~~applic~~ available with declaration.

String

① String is non-primitive Data type
(Size is not fixed)

② It is sequence of characters
(Array of characters)

char[] c = {'a', 'd', 'e', 'p', 'a', 'k'}
String s = new String(c);

char sequence ||

③ String is a class
↳ public final class String
extend Object
implement CharSequence
Serializable, Comparable

④ String s = new String();

(String is an Immutable object)

both are creating obj but difference b/w them is SCP (String constant pool)

⑤ String s = "deepak";

To create String there are three classes

① String ② StringBuffer ③ StringBuilder

String Constant pool (String Literal pool)

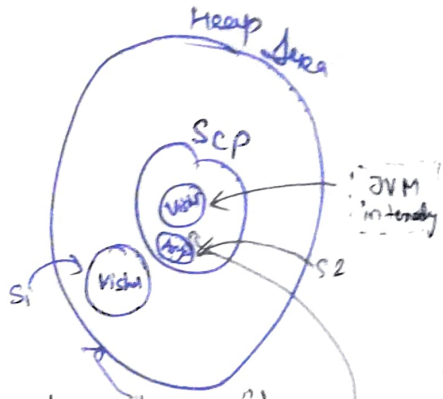
when String obj is created then where & how it is stored

scp → used to store String object.

SCP → is an area in heap memory where Java store string literal

5 Memory areas

- Method Area → 1.8 ver (SCP) (shifted bcz scp size is constant here)
- Heap Area → 1.7 version (SCP)
- Stack Area
- Pc Register
- Native Method Area



Ex:- String S1 = new String ("vishal");
(S1 → obj, new → heap area, "vishal" → string literal also creating)

↳ Here 2 obj is created

Ex:- String S2 = "Aryan";
↳ only string literal is created

↳ here 1 obj is created

The String obj present in SCP are not applicable for Garbage collection bcz a reference variable internally is maintained by JVM.

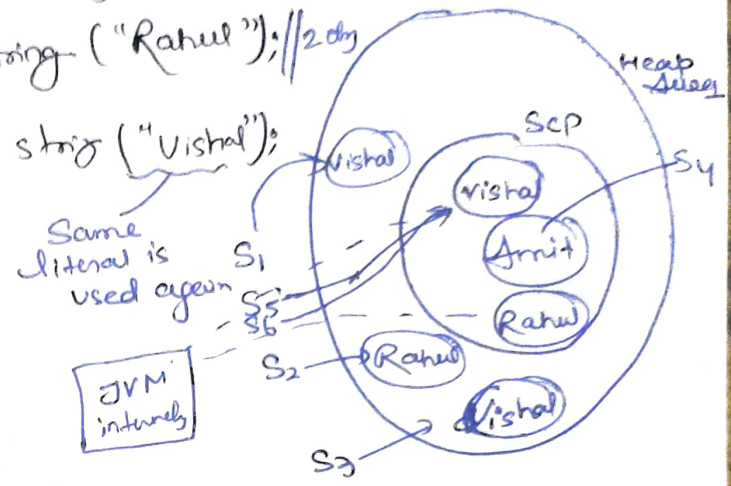
Special case

Ex:- String s1 = new String ("Vishal"); // 2 obj.

Ex:- String s2 = new String ("Rahul"); // 2 obj

1 obj // String s3 = new String ("Vishal");

Same literal is used again



1 obj // Ex:- String s4 = "Amit";

0 object // String s5 = "Vishal";

already this literal present

Although JVM first kr pr same literal ko s5 point kre ge...

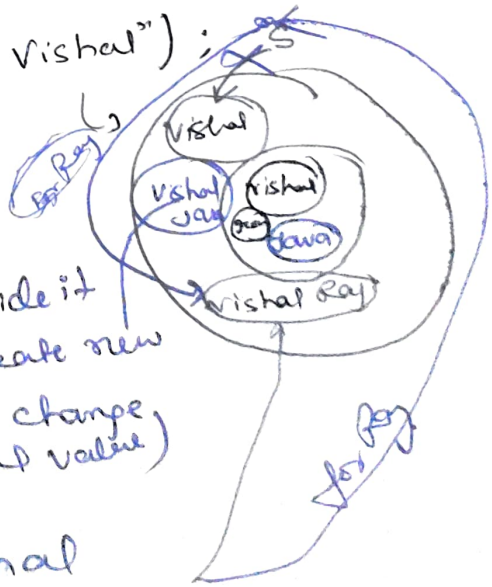
0 object // String s6 = "Vishal";

Immutable (Unchangeable)

i.e. String Obj. are Immutable

Ex:-

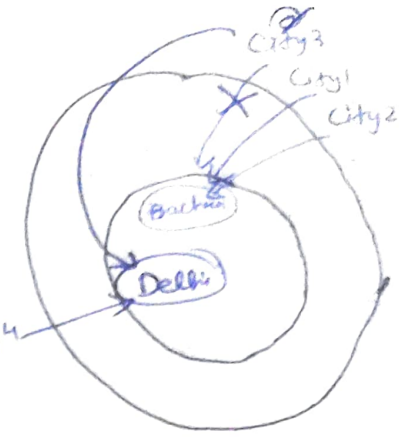
```
String s = new String ("Vishal");
s.concat("Java");
```



It will not override it bcz Immutable it will create new

(It will not change original value)

```
Sop(s); // Vishal
s = s.concat("Ray");
Sop(s); // Vishal Ray.
```



- String city1 = "Bachra";
- String city2 = "Bachra";
- String city3 = "Bachra";

↳ Here no. of obj will be 1 (execution will be fast)

- String city3 = "Delhi";

↳ It will not change previous place i.e (Bachra) it will make new obj bcz if it changes previous value it will effect to all others.

- String city4 = "Delhi";

Why String class is final

final class cannot extend (It can't Inherit)

```
final class ABC
{
}
class Test extends ABC
{
}
```

(we can't)

```
class ABC
{
  final void show()
  {
  }
}
class Test extends ABC
{
  void show()
  {
  }
}
```

(we can't Override show() method)

---

final int a = 10;

a = 20; (This Variable Value is fixed)

final class String
↓
r

→ bcz string class got special features which is not available to other Java classes and making the String class final prevent's subclasses that could break these assumption.

Imp Features of string class
- SCP
- Immutable
- + operator
- security
- synchronization
- security
- class loading

* Final is the keyword used with Class, method, Variable but Immutability is the Concept used for object in which Obj state and Content cannot be changed

final StringBuffer sb = new StringBuffer ("vishal");

~~Sb = sb. append ("Java");~~ — error due to final

Sb.append ("Java");
|
Ham final hai fir v purana value ko change kr de rhe hai but ref change nhi kr skte

Here Immutability breaks
Buffer class Immutable Nhi hai

Equals Method's

Difference b/w == operator & .equals method

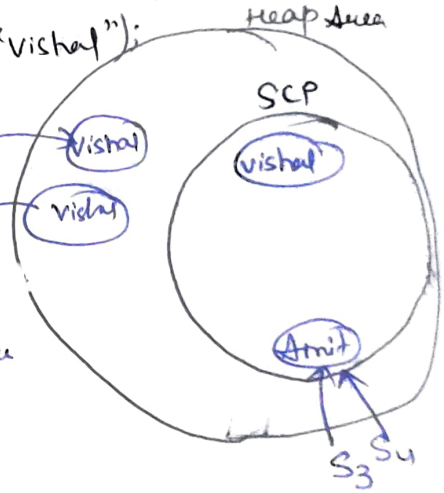
== operator is used for reference comparison and address comparison.

- .equals() method is used for content comparison.

Ex:- String s1 = new String ("vishal");
String s2 = new String ("vishal");

Sop (s1==s2); //false

bcz both have diffrent reference ie s1 & s2



Ex:- String s3 = "Amit";
String s4 = "Amit";

Sop (s3==s4); // True

└ both are refering to same value

---

1) Method of Object class

1. clone()
2. equals (object obj)
3. finalize()
4. getclass()
5. hashcode()
6. notify()
7. notifyAll()
8. tostring()
9. wait()
10. wait (long time out)
11. wait (long time out, int nanos)

class object

```
{
    public boolean equal (object obj)
    {
        return (this == obj);
    }
}
```

current class ke obj

```
class Demo
{
    psvm ()
    {
        String s1 = new String ("Vishal");
        String s2 = new String ("Vishal");
        sop (s1 == s2) || False
    }
}
```

Equals method of object class is used to compare the reference or address of two objects (i.e If two object point to same memory location or not).

(== & Equals method ⇒ Same for Object class)

But

In class String extends object

```
{
    public boolean Equals (object obj)
    {
        // statement
    }
}
```

@Overrided

Here this equal class not for reference comparison

T] is for Content Comparison

sop (s1.equals (s2)); || True

String Constructor

```
class String
{
    public String ()  // constructor's (no-argument)
    {
    }
    public String (String s)
    {
    }
    public String (StringBuffer sb)
    {
    }
    public String (StringBuilder sb)
    {
    }
    public String (char[] ch)
    {
    }
    public String (byte[] b)
    {
    }
}
```

Ex:- PSVM

Immutable — `String s2 = new String ();`
`sop (s2.length ());` ⇒ 0

Immutable — `StringBuffered/Builder sb = new StringBuffered/Builder ("Vishal");`
Immutable — `String s2 = new String (sb);`
`sop (s2);` ⇒ Vishal

```
byte[] b = {101, 102, 103};
String s2 = new String(b);
    sop(s2);   // efg
```

convert byte to string

```
char[] c = {'a', 'b', 'c'};
    String s2 = new String(c);
        sop(s2);  // abc
```

↳ byte ke and char ke value

Why char is better choice then over String for Strong password

password

↳ bcz string obj are Immutable
scp not applicable for Garbage collection

```
char[] s1 = new char{'a','b','c'}
    sop(s1);
↳ // [c@6sqe]
```

↳ obj is printed

∴ char array is better option

→ String Methods

class Test
{
psvm()
{

name = " "; // ↳ I count spaces also

```
String name = "abc";   // name = null
String email = "abc@gmail.com";
String pass = "abc123";
    Sop(name.length());
int i = name.length();
    if(i==0)
    {
        sop("Name is Empty");
    }
```

It will Exception throw

} }

isEmpty() → method of string class is included in java str it return True if the given string is Empty, else false.
(Return True if the length of the stry is 0.)

Ex:- class Test

```
{
    psvm()
    {
        String name = "abc";
        String email = "abc@gmail.com";
        String pass = "abc123";
        sop(name.isEmpty()); //False
                              └→ return boolean value

        if(name.isEmpty() == True)
        {
            sop("Name is Empty");
        }
```

Trim() → Method of string class eliminates only leading and trailing spaces

It check this unicodes value before and after the string, If it exists then removes the space and return the omitted string

```
{
    psvm()
    {
        str --:
        String s = name.trim();   (return String value)
        sop(s);
        if(name.trim().length()==0)
        {
        }
    }
```

Equals() → It is a method. Compares the content of given two strings. If any character is not matched, it return false. If all character are matched then it return True.

→ return boolean

```
class Test {
    p s v m()
    {
        String s1 = "Vishal";
        String s2 = "Amit";
        Sop ( s1.equals(s2) );
                    ↳ false (Boolean)
```

Uppercase & Lowercase
↳ Different.

```
String s1 = "Vishal"
String s2 = "vishal"
sop ( s1.equals (s2) );
        ↳ false.
```

Ignore Uppercase & Lowercase
↳ equalsIgnoreCase()

```
sop ( s1.equalsIgnoreCase (s2) );
        ↳ True
```

Check For Empty using equals → `sop( s1.equals("") );`
↳ False

CompareTo & CompareToIgnoreCase() — Used for comparing two strings lexicographically. Each character of both the string is converted into a Unicode value for comparison. If both the string are equal then this method return '0' else it return positive or negative value.

↳ return value in Int

```
Class Test
{
    psvm ()
    {
        String s1 = "a";    //97    97-65=32
        String s2 = "A";    //65
        Sop (s1.compareTo (s2));
    }
}
```

0 => s1 == s2
+ => s1 > s2
- => s1 < s2

32 //Sop

want to calculate Length -

```
String s1 = "Vishal";
String s2 = "";
Sop (s1.compareTo (s2));
```

↳ 6

+ Operator

```
Class Test
{
    psvm ()
    {
        String s1 = "Vishal";
        String s2 = "Kumar";
        Sop (s1 + s2);   // Vishal Kumar
        Sop (s1 + 10);   // Vishal 10
        Sop (s1 + 10 + 20);   // Vishal 10 20
        Sop (10 + 20 + s1);   // 30 Vishal
    }
}
```

```
sop (s1 + 20 | 10); // Vishal 2
sop (s1 + (10 - 5); // Error
```

Concat (String str) → this method Concatenates One String to the end of another String.

It return a String with the value of the String passed into the method, appended to the end of the String.

```
sop (s1.concat (s2));
```
↳ VishalKumar

Join → static method which Concatenates the given elements with the delimiter and returns the Concatenated String.

* Static join (charSequence delimiter, charSequence... elements)

```
sop (String.join (",", s1, s2);
```
↳ Vishal, Kumar

SubSequence (int beginIndex, int endIndex) – return a CharSequence. The subsequence start with the Char value at the specified Index and end with the Char Value at (end-1)

ex:- class Test

{

psvm (

String s = "This is Index";

Sop (s.substring (3, 4)); → This

return is String type

Sop (s.SubSequence (3,4)); → This

return CharSequence value

Sop (s.substring (3)); → s is Index

return String value

To Update String

There methods are there

1) Replace () 2) ReplaceFirst () 3) ReplaceAll ()

```
ex: class Test
    {
        p s v m ()
        {
            String s1 = "this is demo";
            Sop (s1.replace ("is", "was"));
                                old  new
```

Thwas was demo // both got changed

```
Sop (s1.replaceFirst ("is", "was"));
```

↳ Thwas is demo
(first time only replace)

```
Sop (s1.replaceAll ("is", "was"));
```

↳ Thwas was demo.

Difference b/w ReplaceAll & replace is -

In ReplaceAll we can use Regex (i.e. regular Expression) also.

```
Sop (s1.replaceAll ("is(.)", "was"));
```

↳ Thwaswasdemo

```
Sop (s1.replaceAll ("is(.*)", "was"));
```

↳ thwas

replace (char oldchar, char newchar) → return a String replacing all the old character or charSequence to new characters or charSequence.

ReplaceFirst (String regex, String replacement) → it replaces the first substring that fits the specified regular expression with the replacement statement.

Replaceall (string regex, String replacement) → It replaces all the substrings that fits the specified regular expression with the replacement string.

Indexof() - this method returns the position of the first occurrence of specified character(s) in a string or return -1, If the character not exist.

Ex:- class Test
{
psvm()
{
String s = "Vishal";
0 1 2 3 4 5 6
char/string as argument
Sop(s.Indexof('i')); //1
Sop(s.Index of("al"); //4
→ If return Integer Value
Sop(s.LastIndex of('v')); //6

CharAt() → This method return the character at the specified Index. Index value should lie b/w 0 & length()-1.

Integer value as an argument
h // Sop(s.charAt(3));
↳ return ~~String~~ character

Contains() → This method search the sequence of characters in the given string. If return True if sequence of char. value are found otherwise return false.

Sop(s.Contains("hai"));
↳ True (Boolean return)

StartWith() → this method tells if a String starts with the specified prefix beginning from 1st index. If yes then it will return true else return false.

Sop(s.StartWith("d")); // False

↳ return Boolean

Sop(s.endwith("V")); // True

String Conversion

Case Conversion

- ⊙ ToUpperCase() → convert all character of string to uppercase
- ⊙ ToLowerCase() → convert all character of string to lowercase

Ex:- class Test
{
psvm
{
String s = "Vishal";

VISHAL // Sop(s.toUppercase());

vishal // Sop(s.toLowercase());

Type Conversion

⊙ The valueOf() → JI converts different type of values into String. By the help of String.valueOf() method, we can convert int or long or float or double or object or any other type to String.

NOTE :- ValueOf() method is Static method that why we can call valueOf() method directly by class.

Ex.

```
int a = 10;
String s1 = String.valueOf(a);
```

[due to static method]

toCharArray() → This method converts the given String into a sequence of character. The returned array length is equal to the length of the String.

Vishal // `char[] c = s.toCharArray();`

StringBuffer

① Main Difference b/w String & StringBuffer is

String → Immutable StringBuffer → Mutable

```
String name = "Ray";
name.concat("Kumar");
```

name → ~~Ray~~ Ray

create another memory

name → Ray Kumar

Immutable

```
StringBuffer sb = new StringBuffer("Ray");
sb.append("kr");
```

sb → Raykr

② When we should use String and StringBuffer?

If the data doesn't change or change one or two times only, use String

If the data is frequently changing like in Calculator, Notepad use StringBuffer

syntax of StringBuffer :-

public final class StringBuffer extends AbstractStringBuilder implements java.io.Serializable, CharSequence.

4 constructor :-
StringBuffer ()
StringBuffer (CharSequence seq)
StringBuffer (String str)
StringBuffer (int capacity)

(Method → search ?)

StringBuilder
↳ mutable, non-Synchronized

In StringBuffer → mutable object, Synchronized method

⊙ Synchronization in java guarantees that no two threads can execute a synchronized method which require the same lock simultaneously.
thus, Synchronization increases waiting Time of Thread and effect performance of the system

∴ To Overcome we use StringBuilder

StringBuffer & StringBuilder both have same function, Constructor & method but non-Synchronized

Syntax: public final class StringBuilder extends AbstractStringBuilder implements java.io.Serializable, charsequence

String	StringBuffer	StringBuilder
⊙ Storage: Heap area & SCP	Heap area	Heap area
⊙ Object: Immutable	mutable	mutable
⊙ Memory: if we change the value of string a lot of times it will allocate more memory	Consume less	Consume less memory
⊙ Thread safe - Not safe	All are synchronized ∴ safe ✓	⊙ non-Synchronized ⊙ Not safe
⊙ performance → slow	⊙ as compare to String fast	⊙ fast as StringBuffer
⊙ use → if data is not changing frequently	⊙ changing frequently	⊙ if changing frequently data

Q WAP to reverse a given String ?

we know String s = "Vishal"
0 1 2 3 4 5

1) s.length = 6

2) s.charAT (4) = a

String rev = "";

```
for (int i = s.length - 1 ; i >= 0 ; i--)
{
    rev = rev + s.charAt (i);
}
```

i = 5

rev = "" + l = l

i--;

i = 4

rev = l + a = la

i--

i = 3 . . .

(lahsiv) ✓

Split()

It split a string into an array of substring.

It return a new array

It doesn't change the Original string

If (" ") is used as a Separator, the str is split b/w words.

```
// split string from space.
String text = " Java is a fun";
String[] result = text.split(" ");
```

→ Java is a fun

StringArray

String[] array ⟹ Str array without size

```
String[] arr = new String[2];
```

Convert Array to Str

1) Arrays.toStr(s);

⟹ String.valueOf(str array)

Constructor

- It is block (similar to method) Having same name as that of class Name.
- It does'nt have return type.
- The only modifier applicable for Constructor are public, protected, default & private.
- It Execute automatically when we create an Object.

Need

```
class Employee
{
  String name;
  int Emp-id;

  psvm( String[] args)
  {
    Employee e1 = new Employee();
      e1.name = "deepak";
      e1.emp-id = "101";
    Employee e2 = new Employee();
      e2.name = "abc";
      e2.emp-id = "102";
    :
  }
```

Value changed give same value. (e1 → name-null, emp-id-0; e2 → name-null, Empid-0) Creating same value.

Different Value — but too many obj.

So for unique value use const.

```
class Employee
{
  String name;
  int Emp-id;
  Employee(String name, int emp-id)
  {
    this.name = name;
    this.emp-id = emp-id;
  }
  psvm (--)
  {
    Emp e1 = new Emp("deepak", 101);
  }
}
```

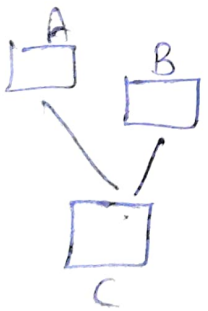
Use: To Initialize an Object

Inheritance

we can not Inherit Constructor from Super to SubClass and private member also.

Constructor can be Involked but can't Inherit as bcz it is not the member of class.

More than One Super class Can't use bcz there is no Concept of multiple Inheritance in Java.



X bcz if A & B have same method and that method is inherited by C and then we call that method so Compiler Confuse whome method is to call

static

```
class Test
{
    static void display()
    {
        sop("1");
    }
}
```

due to this we can not call this method through obj. we had to call this with class name

(static Method belongs to class not to Object)

```
psvm(---)
{
    display();
    Test.display();
}
```

(within a class)

Test → class name.

Need :- Just for Memory Management.

Ex:-

```
class Test
{
    void display()
    {
    }
    psvm(---)
    {
        Test t = new Test();
        t.display();
    }
}
```

It create an obj which occupy space

But in static No need to create an Object it directly call object

Rule for "Static" Method

- "Static" method belong to the class, not to the object.
- A "static" method can be accessed directly by class name and doesn't need any obj
- A "static" Method can acces only static data. Can't Acces non-Static data (Instance Variable)
- "Static" Method Call only other static method and cannot call a non-static method.
- "Static" Method Cannot refer to "this" or "Super" keyword in anyway.

Ex:

```
class Test
{
    int i = 10;
    static void meth()
    {
        show();
        Sop(i);
    }
    void show()
    {
    }
}
```

Error

Can't Acces non-static Variable & non-static can't called in static method