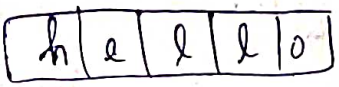
Strings in Java

Unit-1

Q) How r strings represented in C/C++?

ans) In c, c++ strings r arr of char.

eg char c[] = {'H', 'e', 'l', 'l', 'o'};



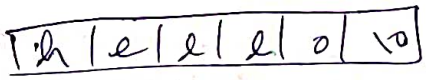
→ characters

c → // arr of char

Char c[] = {"Hello"}; → string

// arr of char terminated by null char.

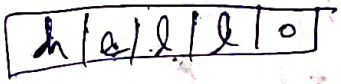
// string is an arr of char, so it can be stored in an arr of char.



c →

String s = "Hello";

// string class can also be used to represent string as an arr of char.



s →

How r string ref in java :

- In java, strings r implemented as obj of String class.
- String is an inbuilt class in java.lang.String
  It is auto^{ly} imported.

String S1 = "Hello world";
//when string is created, they r created in the form of objects

//an obj S1 contains the string "Hello"

What are the features of string in java

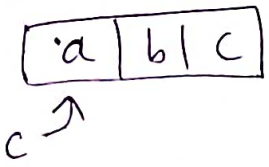
1) strings r unindexable. They can't be indexed like an arr.

eg String s = "Hello";
   Sopln(s[0]); // ✱ Error.

2) Strings r immutable ✱✱✱
- immutable means can't be modified
- strings r made immutable, to inhance performance as optimization is done on the immutable strings.

eg In java :-

1) char c[] = { 'a', 'b', 'c' };
// This is allowed. It is an arr of char.



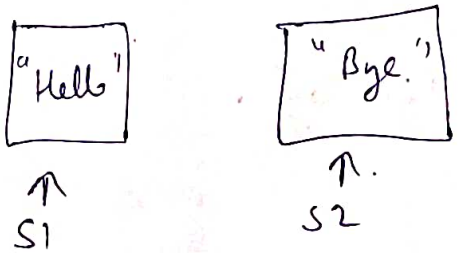
2) char c[] = { "Hello" };
// Not allowed as string obj can't be converted to arr of char.

3) String s = "Hello";
sopln(s[0]); // Indexing is not allowed.

String r ref variable

- When we create a string, then string variable r ref variable.

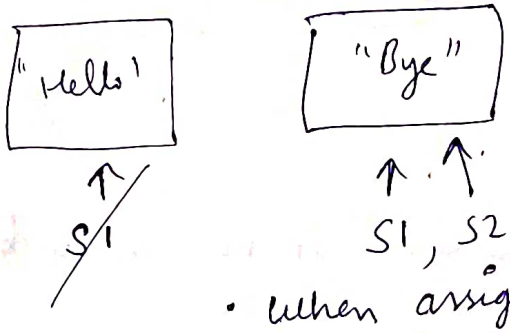
eg String s1 = "Hello";
String s2 = "Bye";



- s1, s2 r ref var. They point to "hello" & "bye" obj.

*** What happens when you use assignment operator (=) with string

```
String s1 = "Hello";
String s2 = "Bye";
   s1 = s2;
// Shallow copy occurs. Both ref point to same obj.
```

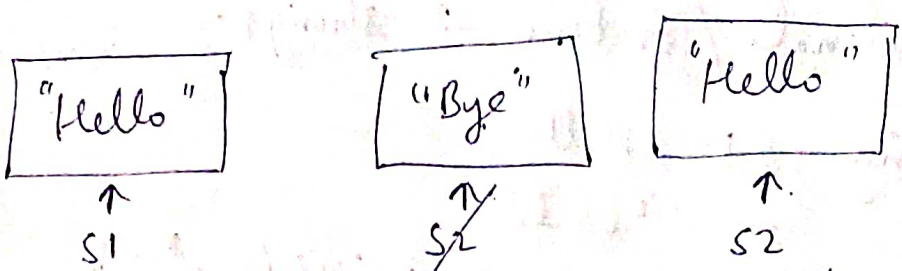


- when assignment opr is used both ref point to same obj.
- obj r not copied as expected.
- Now s1 & s2 point to same obj.

Q) How to copy one string to another.

- use new operator
- Deep copy occurs.
- New obj is created
- Both s1 & s2 point to diff obj.

```
String s1 = "Hell";
"      s2 = "Bye";
       s2 = new String(s1); // s1 copied to s2
                            // deep copy.
```



//A new ~~obj~~ copy of $1 is created & $2 points to new obj.

//Now $1 & $2 point to diff strings.

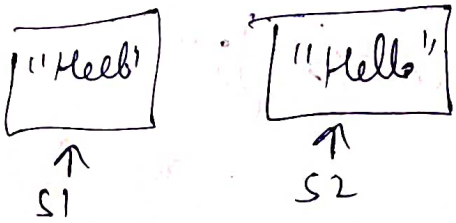
//Deep copy occurs.

Functions in String

- String class provides some fun to operate on strings

① To copy string use new operator
Donot use assignment opr (=)

```
String $1 = "Hello";
Strng $2 = new Strng ($1);
```



② find length of a string.
- Use length()
- Do not use length or size().

```
String $ = "Hello";
Sopln($.length());
```

③ Concatenate / combine 2 strings.

String $s1 = "Hello";
" $s2 = "World";
" $s3 = $s1 + $s2; // "HelloWorld"

$s3 = "4+4=" + 4 + 4
(4 converted to string)

O/P : "4+4=" 44 ✓

Not "4+4=" 8 X.

④ Indexing an el of string - use charAt();

String $s1 = "Hello World";
Sopln($s1[0]); // Not an array, can't be indexed

//use charAt(0); → index 0
Sopln($s1.charAt(0)); // print 0th element

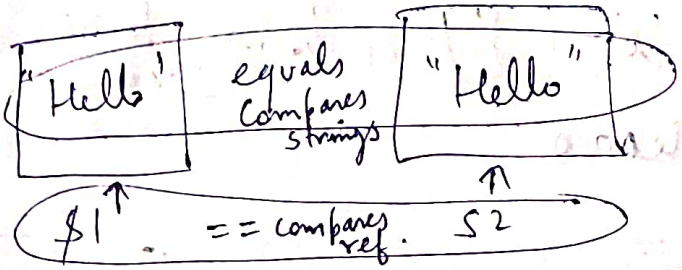
for(i=0; i<n; i++)
{
Sopln($s1.charAt(i));
}

(5) Convert String to char array.

```
String s1 = "Hello";
char ar[];
ar = s1.getChars();
```

(6) Compare if strings r equal or not.
use equals(), do not (s1 == s2)

```
S1 = "Hello";
s2 = "Hello";
```




```
s1.equals(s2); // true
s1 == s2 ; // false both ref r diff
```

(7) indexOf() **

// search a given char or string
// returns position found or -1

eg `s1.indexOf('l');` //returns 2 (0 1 2 3 4 — "Hello")
`s1.indexOf("lo");` // returns 3 (0 1 2 3 4 — "Hello")

⑧ replace ( )

// returns a copy of original string
// String r immutable, so doesn't modifies the original string

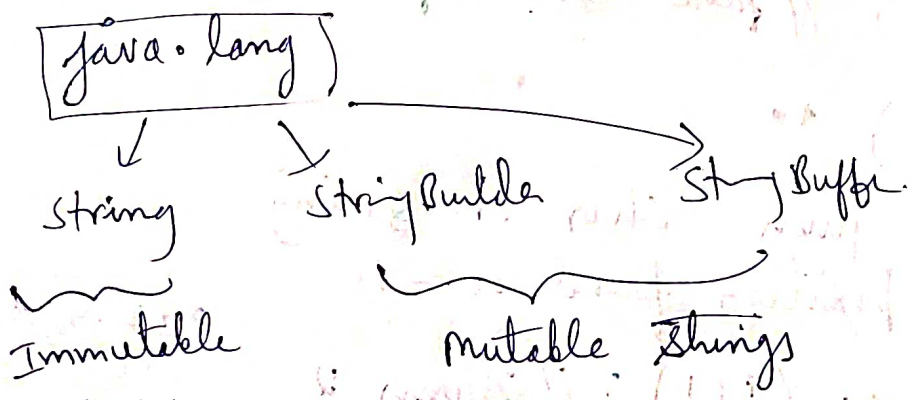
```
$1 = "Hello";
$1.replace('l', 'o');
```

old, new

```
Sopln($1); // $1 is not changed.
```

```
$1 = $1.replace('l', 'o'); // $1 points to new string.
Sopln($1); // Heooo
```

StringBuilder & StringBuffer classes



- String class p/v (provide) immutable strings
- It doesn't p/v any fun to modify strings

- There r 2 classes wch allows to create mutable string objs. They r alternative to string class.
- These classes also p/v fun to modify a string
- There classes p/v growable (size can be changed) & modifiable strings

StringBuffer	StringBuilder
• Less efficient • It is thread safe or synchronized	• faster • Not thread safe.

Fun in StringBuffer Class

- It p/v fun wch can modify a string

1) Creating a string

```
StringBuffer s = "Hello World";
```

2) Change a char at given pos.

```
s.setCharAt(0, 'K'); // Kello world
```

(0 → index, 'K' → new char)

(3) insert ( )

$ = "Hello world";

$.insert (5, "-my-");

index → 5 ; insert string → "-my-"

0 1 2 3 4 5 6 7 8 9

Hello world

Hello-my-world

(4) reverse ( )

// to reverse a string

$.reverse ( );

// "dlrow olleH".

(5) delete ( )

deleteCharAt (index);

$ = "Hello World";

$.delete (4, 8);

start = 4 ; end = 8

0 1 2 3 4 5 6 7 8 9

Hello World

↳ Hell d;

$.deleteCharAt (0);

↳ index

Hello World

↳ ello world

6) replace

$.replace(4, 6, "is");

start, end, new string

0 1 2 3 4 5 6 7 8 9 10

Hello World

Hell is World.