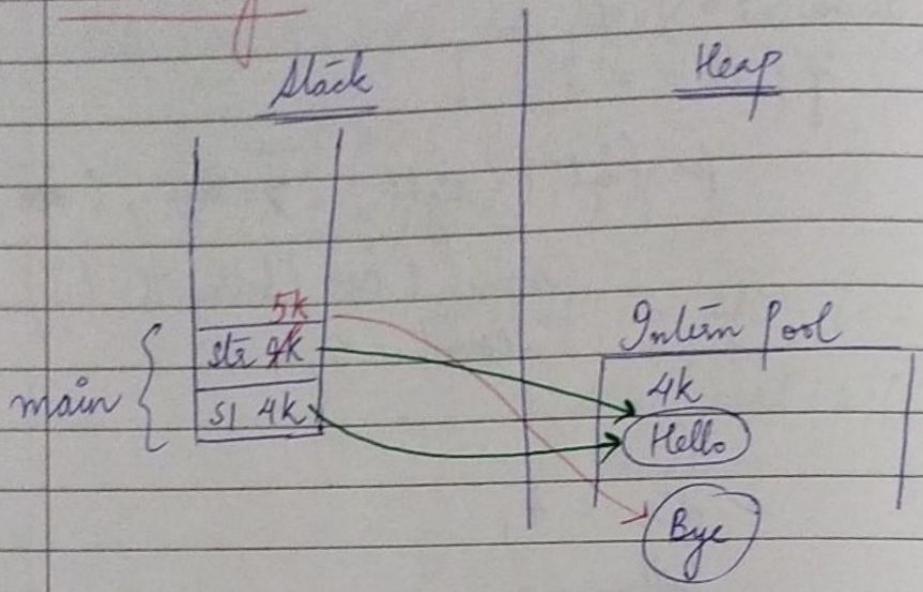
String



String str = "Hello";

String s1 = "Hello";

→ Space saved

→ Now, if str = "Bye";
s1 = ?

Strings are IMMUTABLE

No

⇒ We cannot change the value at 4k address
⇒ We can definitely make str point to another location

(4k Hello) → (5k Bye)

PAGE NO. icon
DATE : / /

substring

① substring (si, ei)
include exclude

String str = "Hello"; (0 1 2 3 4)

str.ss(0,3) → Hel
(0,4) → Hell
(2,4) → ll
(2,5) // No ERROR → llo
(3,3) → "" (empty string)

② substring (si)

ss(1) → ello
ss(3) → lo

Concatenation

S1 = "hi"
S2 = "bye"

S3 = S1 + S2
= hibye

S1.concat(S2) = hibye

PAGE NO. icon
DATE : / /

0 1 2 3

Q. a b c d

si ei	si ei	si ei	si ei
a (0,1)	b (1,2)	c (2,3)	d (3,4)
a b (0,2)	b c (1,3)	c d (2,4)	
a b c (0,3)	b c d (1,4)		
a b c d (0,4)			

si ⇒ 0 → (3) ← str.length()-1

ei ⇒ si+1 → (4) ← str.length()

```
for(int i = 0; i <= str.length() -1; i++){
    for(int j = i+1; j <= str.length(); j++){
        String ss = str.substring(i,j);
        System.out(ss);
    }
}
```

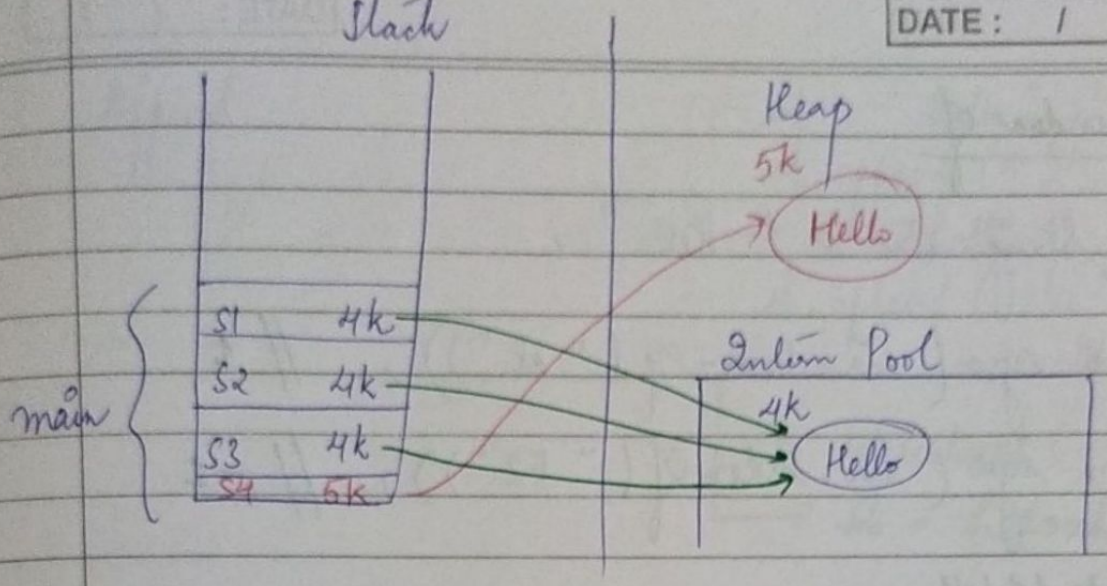
Concept

S1 = "Hello"

S2 = S1 ←

S3 = "Hello"

** JVM will not check in intern pool
** JVM will check in intern pool whether the string exists or not



S4 = new String ("Hello");

↓

Creates new string outside the intern pool

** S1 == S2
4k 4k → address is same
return True //

** S1 == S3 → true //
4k 4k

** S1 == S4
4k 5k → false //

But we want to compare content of strings (not ref. addresses) //

∴ use S1.equals(S4) → true //

↓ Hello ↓ Hello

→ Character by character

PAGE NO. icon
DATE : / /

indexOf

```
String str = "Hello";
** syso(str.indexOf("el")); // 1
** syso(str.indexOf("El")); // -1
```

startsWith

```
str.startsWith("He")   // true
str.startsWith("he")   // false
```

StringBuilder

	String	StringBuilder
1.	`String s = "hello"`	`StringBuilder sb = new StringBuilder();`
2.	`syso(s);`	`syso(sb);`
3.	`s.length();`	`sb.length();`
4.	`s.charAt(2);`	`sb.charAt(2);`
5.	Update X	`sb.setCharAt(2, 'w');` → hewlo
		`sb.insert(3, 'a');` → hewalo
		`sb.insert(4, "bc");` → hewlbco
		`sb.insert(5, "de");` → hewlode
6.	Insert X	

Range: $0 \rightarrow \leq$ str.length

PAGE NO. icon
DATE : / /

7. Append

```
sb = "hello";
sb.append(a);
→ sb = "helloa"
sb.append("abc")
→ sb = "helloabc
```

8.

```
sb.deleteCharAt(2);
→ sb = heloabc
```

String Builder Function Range

1. setCharAt → 0 → ≤ sb.length() - 1

hello
01234

setCA(5, 'e'); // ERROR (Trying to access an index which is not present)

2. deleteCharAt → 0 → ≤ sb.length() - 1
3. insert → 0 → ≤ sb.length()

(5)

Performance

Strings are immutable → so,

String s1 = "abc"; → 4k
String s2 = "def"; → 5k

s3 = s1 + s2 → 6k

"abcdef"

→ original string not changes nhi hote
→ Items gets copied to another location & then operations would be performed

PAGE NO. icon
DATE : / /

Imp

```
String s = "";
for (i=1 → n) {
    s = s + i;
}
```

$O(n)$ ~~$O(n)$~~

$O(n^2)$

3

$i=1$ `""` $+1$ = `"1"` → 10k

1 $i=2$ `"1"` $+2$ = `"12"` → 11k

2 $i=3$ `"12"` $+3$ = `"123"` → 12k

3 $i=4$ `"123"` $+4$ = `"1234"`

4

5

6

:

$$1 + 2 + 3 + 4 + \dots + n$$

$$\frac{n(n+1)}{2} = n^2$$

StringBuilder → $O(n)$

Performance Comparison

```
long start = System.currentTimeMillis();
appendString(100000);
long end = System.currentTimeMillis();
Sysо(end - start)
```

PAGE NO. icon
DATE : / /

```
void appendString (int n) {
    String s = "";
    for (int i = 1; i <= n; i++) {
        s = s + i;
    }
}
```

```
void appendString (int n) {
    StringBuilder sb = new StringBuilder();
    for (int i = 1; i <= n; i++) {
        sb.append(i);
    }
}
```

o/p : 10656
81 <1 sec !!

** SB to string
sb.toString();

** String to SB
String str = "hello";
StringBuilder sb = new SB(str);

** Take i/p of SB
(1) Take i/p of string
(2) string to SB