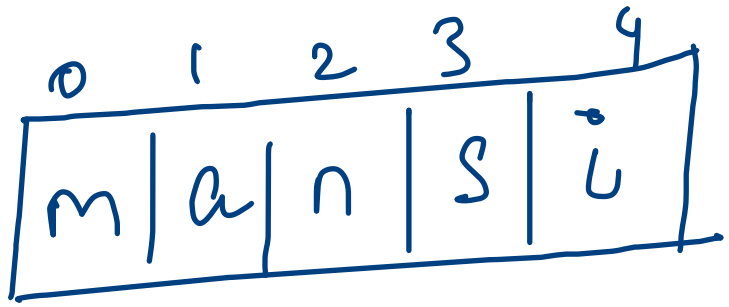
Strings : used to store more than one character.

Eg. name.



Indexing start from Zero.

char c = 'c' ; — character

String str = "abcdef" ;

String — Non-primitive data type / keyword

str — name of string / reference

"abcdef" — data

To print the string

use charAt()

```
Syso(str.charAt(i));
```

```
Syso(str[i]); X wrong way.
```

→ str.length(); - for finding the length of the string.

substrings : part of the string

✓✓ str.substring(start Index, End Index);

str.substring(1, 4); → // ans

last index will not print, will print till last-1.

✓ str.substring(start Index);

str.substring(1); // ansi

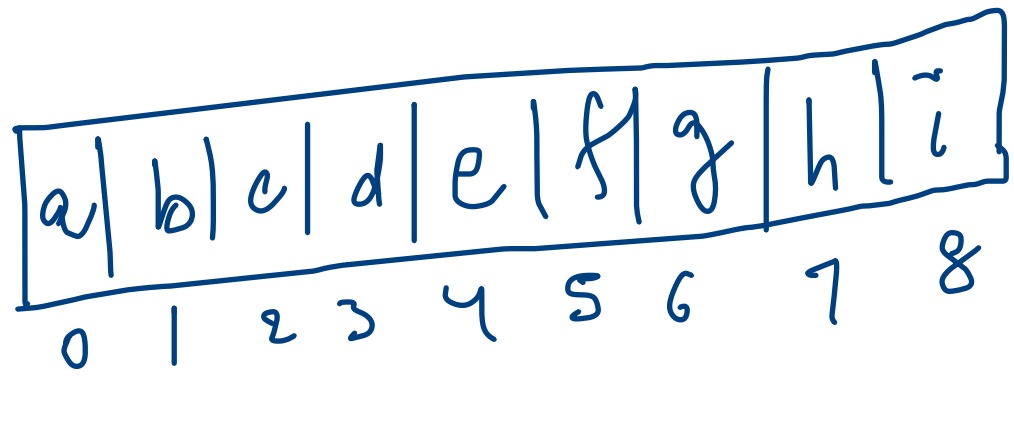
→ str.substring(0); → complete string will get printed

→ sysо(str); → complete string

→ str.substring(str.length()); → empty string

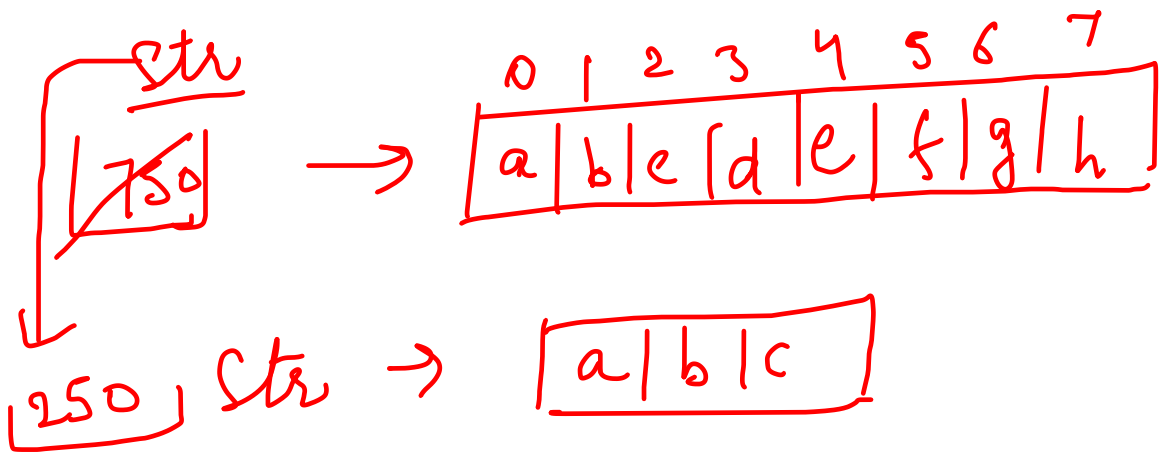
→ str.contains("cde"); T

→ str.contains("bde"); F



→ it will check whether given str is present in your main string.

How strings are stored?



String str = "abcdefgh";

str = "abc";

Strings in java are <u>immutable</u>.

↳ means we can't change the elements of a string.

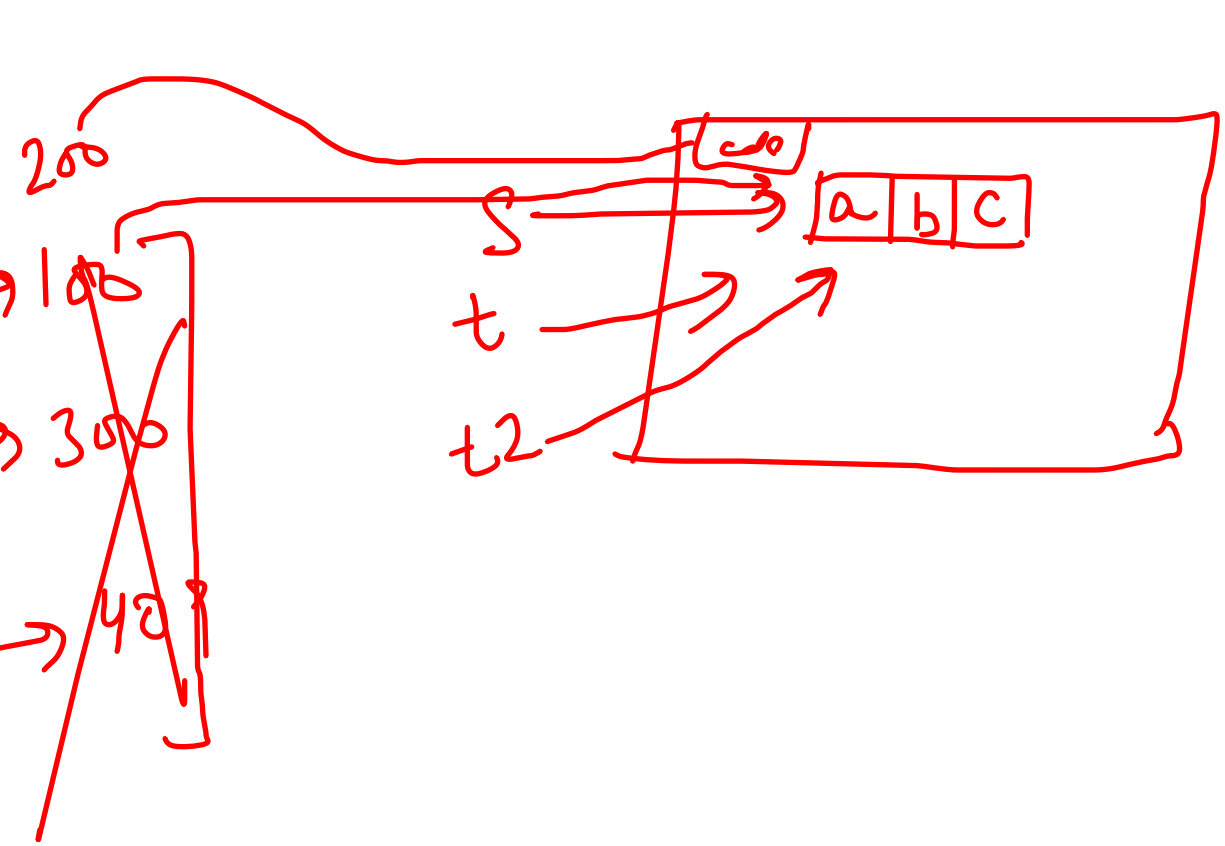
set charAt(i)

String Pool

String s = "abc"; → 100

String t = "abc"; → 300

String t2 = "abc"; → 400



Strings having same element refer to a single string.
No new string will be created.

→ memory efficient

Reference change - allowed
Content change - not allowed.

str = "abc"

str = "abc" + "def"

→ abcdef.

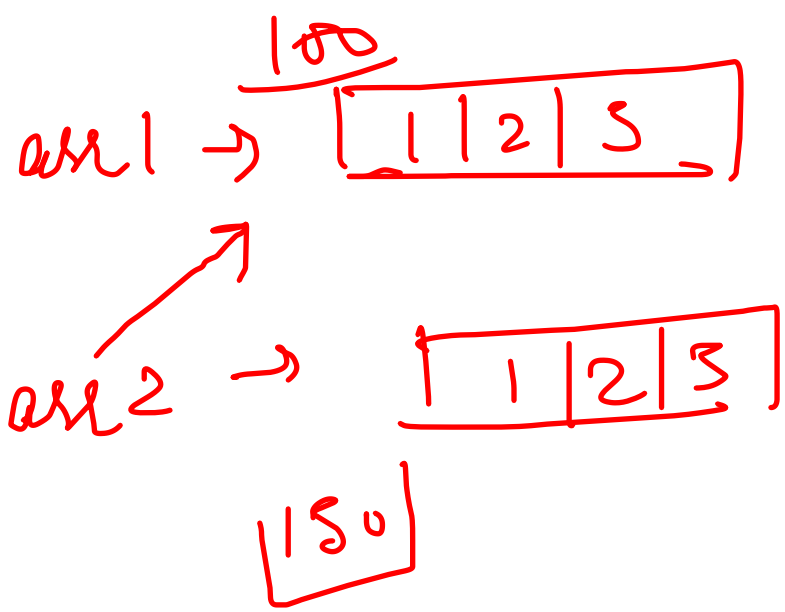
→ str.concat("ghi");

o/p → abcdefghi

Comparision of non-primitives

arr1[] = {1,2,3};

arr2[] = {1,2,3}



arr2[] = arr1

100 ≠ 150

address will get compared not content.

arr1[i] == arr2[i]

In case of strings, the o/p will be equal due to string pool concept.

To compare content, we use .equals.

```
arrays.sort(arr);

int sum = 0;
int num = 1;
int added = 0;

while(added < k){
    if(arrays.binarySearch(arr, num) < 0){
        sum += num;
        added++;
    }
    num++;
}
return sum;
```