Strings

Non Primitive

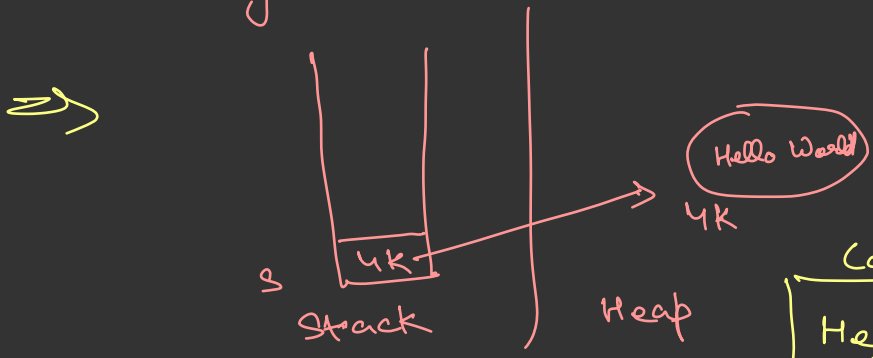
↳ sequence of characters

↳ denoted by "_"

"Newton School" → Only characters

// String s = "Hello World";

Memory



Console

Hey How are you

// "Hey"

Input

scn = new Scanner();

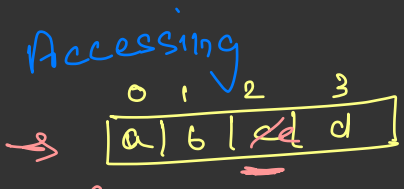
String s1 = scn.next()

String s2 = scn.nextLine() // "_ How are you"

Output

```
Syso (s1);   // Hey
Syso (s2);   // _How are You
```

Accessing




```
a[2] = 'e'
char d = a[0]; // a
```

charAt (index)

S = "Hey" (0, 1, 2)

```
char val = s.charAt(0)   // H
Syso(val)                // H
char val2 = s.charAt(2);
Syso(val2);              // y
```

$ *

→ a, b, c

→ A B C D

Char 0, 1, 2

ASCII

b = 0 ; 0-256

A - 65

B = 66

a = 97

Char c = 97;

Syso(c); // a

Substring

↳ Part of given string

s = "Hey"

s2 "Hey Hello" // 9 length

0 1 2 3 4 5 6 7 8

s.substring(i, j)

i → Start index

j → End index

s.substring(0, 2) // He

(0,1) // H

(0,4) // Hey-

(2,6) // y He

(3,8) // -Hell

(0,9) // Hey Hello

(0,0) // "" Empty String

(2,1) // Error

s.substring(1) // ey Hello

→ Start index

Append Strings (Concatination)

```
String a = "Hey";
String b = "Hi";
String c = a + " " + b;
sysO(c) // Hey Hi
```

```
c // Hey Hi
c = c + 9;   // c += 9
Syso(c);     // Hey Hi 9
```

length is a function

`s.length()` → returns length of s

```
s = "Hey How are you?";
```

(indices: 0 1 2 3 4 5 ...)

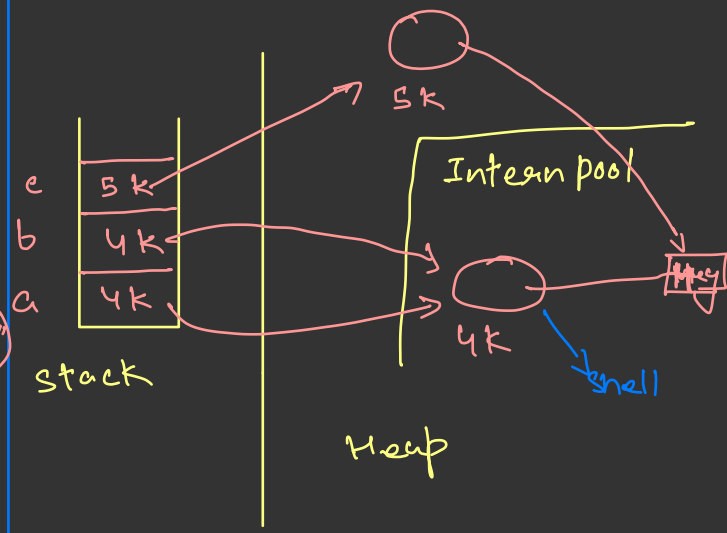
```
String arr[] = s.split(" ");
```

// arr = | "Hey" | "How" | "are" | "you" |
(indices: 0 1 2 3)

`s.toCharArray()` ⇒ | H | e | y | | H | o | w | ...
(indices: 0 1 2 3 4 5 6 ...)

Memory

```
String a = "Hey";
String b = "Hey";
String c = new String("Hey");
```



What ?

Why ? → To optimise space

Implication

(1) Comparision

a) a == b // true

↓

check the address

a == c // false

b) .equals → first compare on address

↳ compare character by character

a.equals(b) // true

a.equals(c) // true

→ Always use .equals

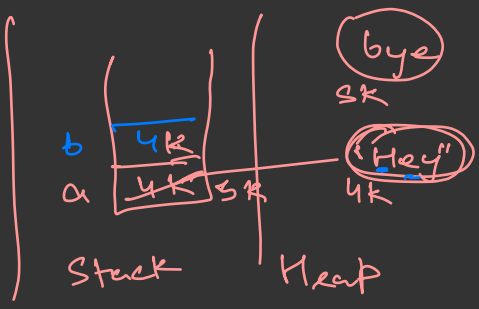
② Immutability

↳ strings are immutable

↳ reference is mutable
instance is not

String a = "Hey"
String b = Hey
a.setCharAt(i) ✗

a = "bye";



Implications of immutability

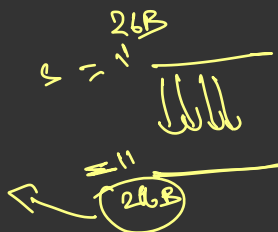
→ Performance

s = "Hey";

s = s + 'a';

s is a string of 2GB

s + = 'a'



Ques Print all substrings

H (0,1) e (1,2) y (2,3)
He (0,2) ey (1,3)
Hey (0 3)

Start = 0 → length - 1

end → start + 1 → length

Ques check palindrome

naman

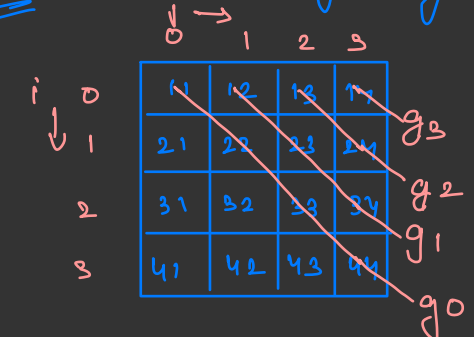
e a

121

212

123 X

Ques Print diagonaly



gap = j - 1 ⇒ 0 – arr.length-1

i → 0

j ⇒ g – arr.length-1;

→ 11, 22, 33, 44, 12, 23, 34, 13, 24, 14

```
for(int gap = 0; gap < arr.length; gap++)
{
    for(int i=0, j=gap; j < arr.length; i++, j++)
    { Syso(arr[i][j]);
    }
}
```