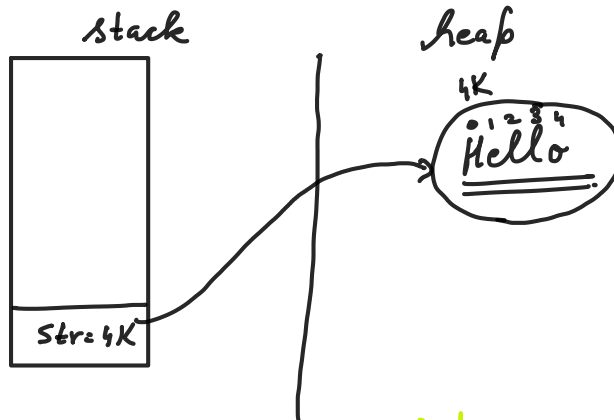


Lecture 6

Saturday, 18 January 2020 2:07 PM

Strings → Non-primitive.

```
String str;
Sysout(str); // x.
str = "Hello";
```



```
Sysout(str); // Hello
Sysout(str.length()); // 5
```

"Hello" x → Java
No null character.

```
Sysout(str.charAt(0)); // 'H'
Sysout(str.charAt(3)); // 'l'
Sysout(str.charAt(str.length())); // Exception
// 5 → Exception

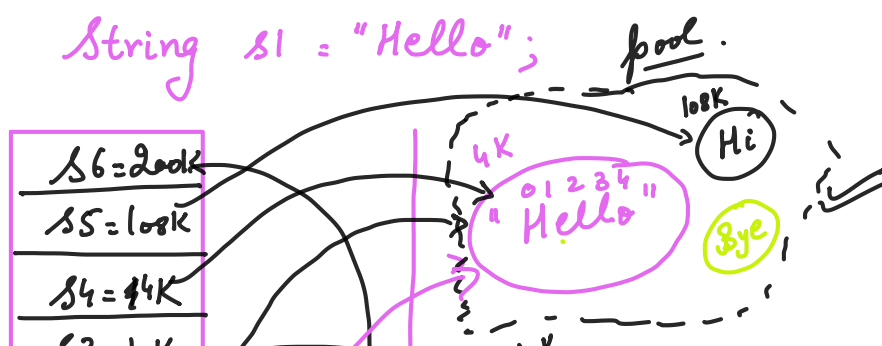
Sysout(arr.toString()); // [ ]
Sysout(str.toString()); // Hello

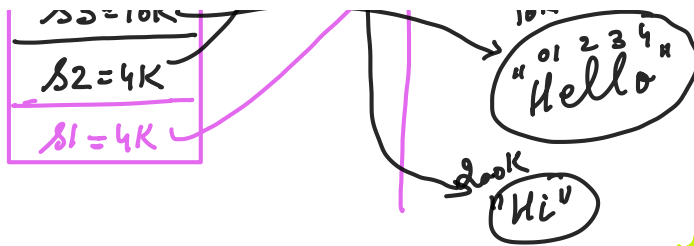
for( ) {
  - x
  3
  → "Hello"
}

for( ) {
  =
  3
  → "Hello"
}

str.substring(3, 5); // "lo"
str.substring(1, 4); // "ello"
str.substring(2); // "ello"
str.substring(4, 2); // ""
str.substring(2, 2); // ""
str.substring(2, 6); // "ello"
str.indexOf("Hell"); // 0
str.indexOf("hell"); // -1 } Case-sensitive.
```

String s1 = "Hello";





String s2 = s1;
 ✓ String s3 = new String("Hello");
 String s4 = "Hello";
 Intern pool.
 X String s5 = "Hi";
 String s6 = new String("Hi");
 .setCharAt(0, 'a'); X
 X s1 = "Bye";

Strings are immutable (can't be changed).
 ↳ No intern pool.

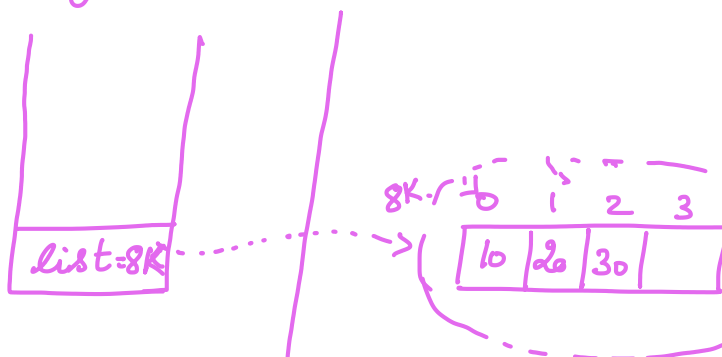
syso(s1)
 syso(s2)

syso(s1 == s2) ✓
 syso(s1.equals(s2)) ✓
 X syso(s1 == s3) . s1.equals(s3) ✓
 ✓ syso(s1 == s4) s1.equals(s4) ✓
 X syso(s2 == s3) s2 — (s3) ✓
 X syso(s4 == s3) s4. — (s3) ✓

Arrays.
 ↳ ArrayList.

Capacity → 4, size = 4


ArrayList<Integer> list = new ArrayList<>();



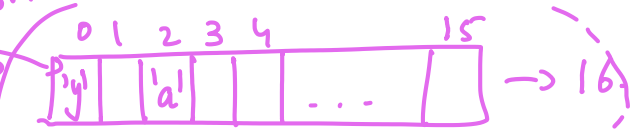
syso(list.size());
 ↳ 0
 syso(list) = []
 list.add(10);
 list. — (20);
 list. — (30);
 list. — (40) X
 list.set(2, 10);

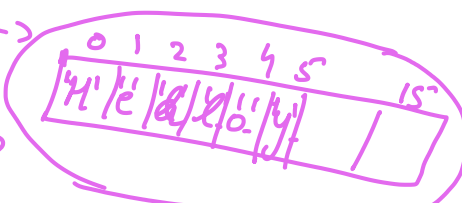
for (int i=0; list.size(); i++) { syso(list);
 syso(list.get(i)); [10, 20, 30]

3

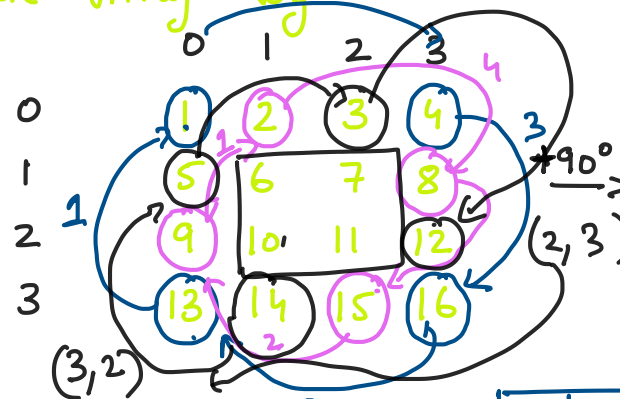
✓ `ArrayList<ArrayList<Integer>> list = new ArrayList<>();`
`→ list.add(new ArrayList<Integer>());`
`list.get(0).add(10);` 4k
 100k.  Size = 2
 Cap = 2

String Builder

10k
`sb = 10k`  `→ 16`
`StringBuilder sb = new StringBuilder("Hello");`
`sb.append('y');`
`sb.setCharAt(2, 'a');`

3
`sb = 100k`  `→ 16`
`sb.append('y');`
`sb.setCharAt(2, 'a');`
`sb.toString() → "Hello"`

Rotate Array By 90°

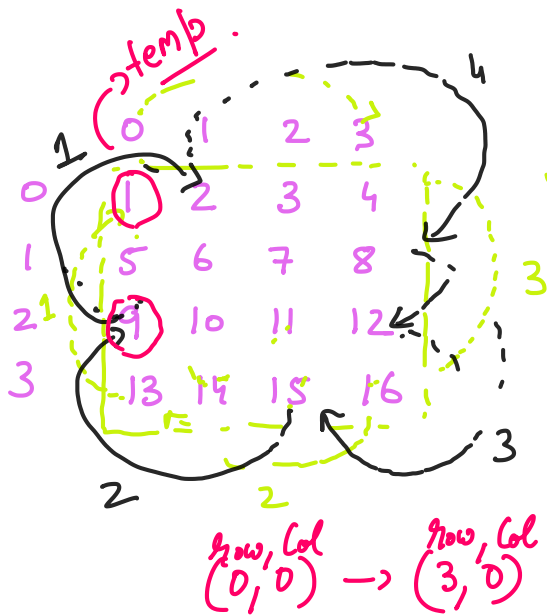
inplace.

 extra space $\rightarrow x^2$

floor $\frac{N}{2} \cdot N \cdot N$

0 1 2 3
 0 13 9 5 1
 1 14 10 6 2
 2 15 11 7 3
 3 16 12 8 4

13	9	5	1
14	10	6	2
15	11	7	3
16	12	8	4

10	11	1	5
16	12	8	4



\hookrightarrow $N/2$ passes
1st pass
 first row 13 \hookrightarrow 1 first

first row $13 \rightsquigarrow 1$ first
 $16 \rightsquigarrow 13$
 $4 \rightsquigarrow 16$
 $1 \rightsquigarrow 4$.

5 → 2

14-25

12 ~ 14

4 ≠ 4

for (int i = 0; i < n/2; i++) { \Rightarrow 2

for (int j = 0; j < n-i-1; j++) {

int temp = arr[i][j] (0,0)

\rightarrow arr[i][j] = arr[n-1-j][i]; (3,0)

(3,0) arr[n-1-j][i] = arr[n-1-i][n-1-j] (3,3)

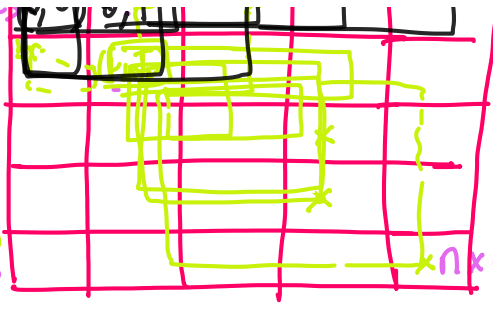
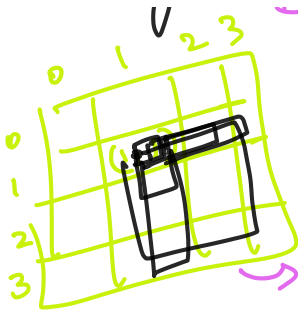
(3,3) arr[n-1-i][n-1-j] = arr[j][n-1-i] (0,3)

arr[j][n-1-i] = temp;

3

$$\begin{aligned} & \{ \\ & (0, 1) \leftarrow (2, 0) \\ & (2, 0) \leftarrow (3, 2) \\ & (3, 2) \leftarrow (1, 3) \\ & (1, 3) \leftarrow (0, 1) \end{aligned}$$

Sum of all submatrices.



$$4 \times 1 \\ 2 \times 4 \\ 1 \times 4 = 16$$

tl_i → 1
tl_j → 1
br_i → 1 → n-1
br_j → 1 → n-1

Rectangles.

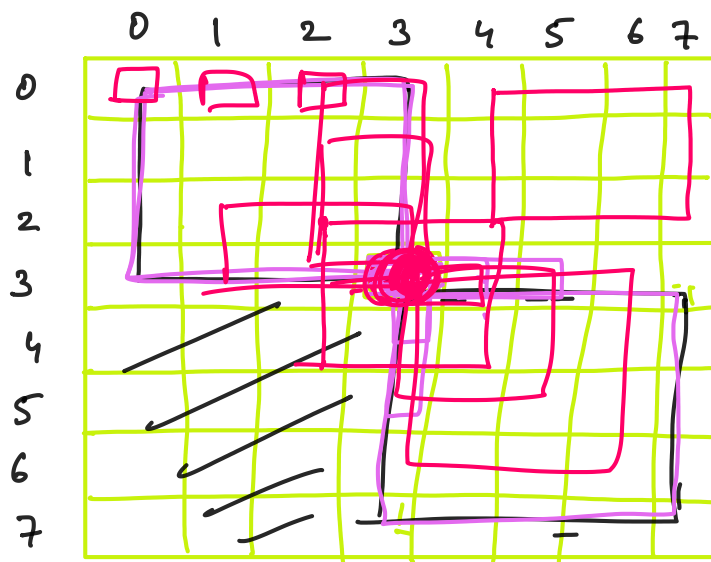
tl_i tl_j
br_i br_j

for (tl_i → 0 ; tl_i ≤ n-1 ; tl_i++) {
 for (tl_j → 0 ; tl_j ≤ n-1 ; tl_j++) {
 for (br_i → tl_i ; br_i ≤ n-1 ; br_i++) {
 for (br_j → tl_j ; br_j ≤ n-1 ; br_j++) {
 for (i → tl_i ; i ≤ br_i ; i++) {
 for (j → tl_j ; j ≤ br_j ; j++) {
 Sum = Sum + arr[i][j];
 }
 }
 }
 }
 }
}

(0,0)

$$O(n^6)$$

tl_i, tl_j



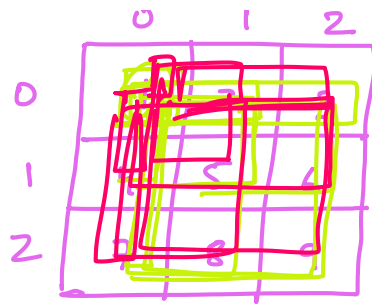
(br, br)

(8x8)

(br_i, br_j)
tl ⇒
br ⇒

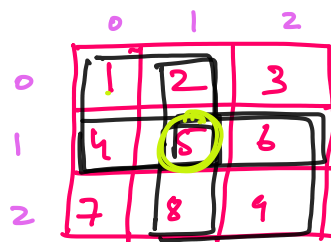
$$(i+1) * (j+1) \quad (4*4) * \\ (n-i) * (n-j); \quad (5*5) * \\ (4*4) * (5+5) + \\ arr[i][j]$$

$$Sum = Sum + (4*4 * 5 + 5) arr[i][j]$$



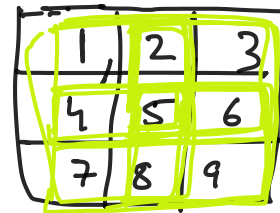
$\hookrightarrow \underline{\underline{500.}}$

$[br, by] \Rightarrow (0,0) \rightarrow (br, by) \rightarrow 1*1 = 1.$
 $(tlr, tly) \rightarrow 9 \leftarrow$
 $9*1+1.$



$4+4+5$

$\underline{\underline{n^2}}$



$\underline{\underline{16}}$