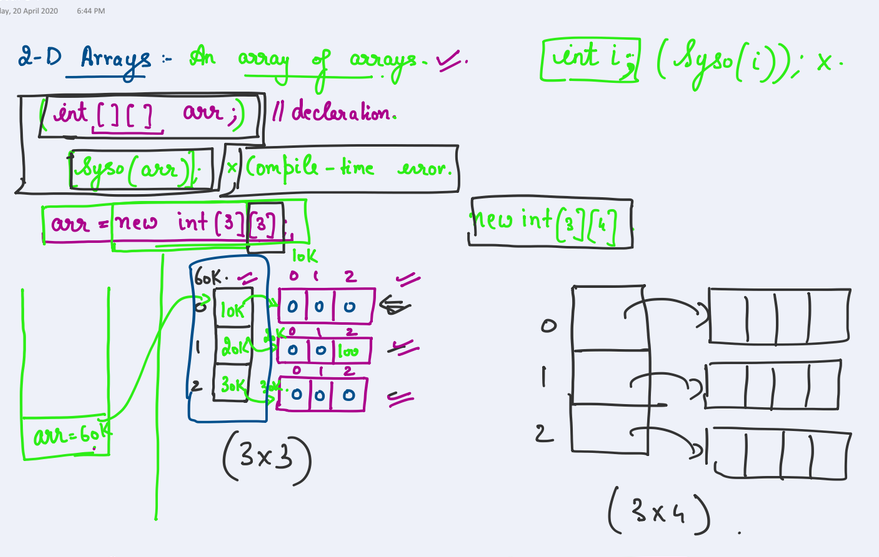
Lecture 7
Monday, 20 April 2020 6:44 PM

2-D Arrays :- An array of arrays. ✓.

int i; (Syso(i)); x.

```
int [][] arr; // declaration.
Syso(arr); x Compile-time error.
arr = new int[3][3];
new int[3][4]
```



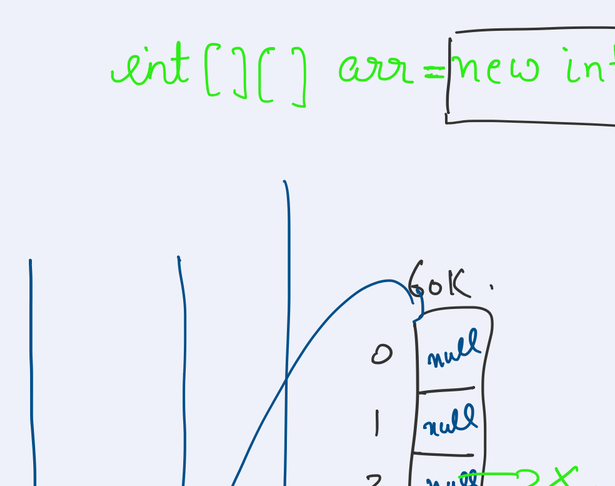
60K
10K
20K
30K
0 1 2
0 0 0
0 0 100
arr=60K
(3x3)
0
1
2
(3x4)

(4x3)
Syso(arr.length) ④

```
Syso(arr); // 60K.
Syso(arr[1].length); //3
Syso(arr[0].length); //3
Syso(arr[0][4]); // Error.
Syso(arr.length); //3
Syso(arr[0][1]);
  10K [1]; //0.
arr[1][2] = 100;
  20K [2]:
```

int [][] arr = new int[4][ ]; → ?

4x3 → 4 rows, 3 cols.

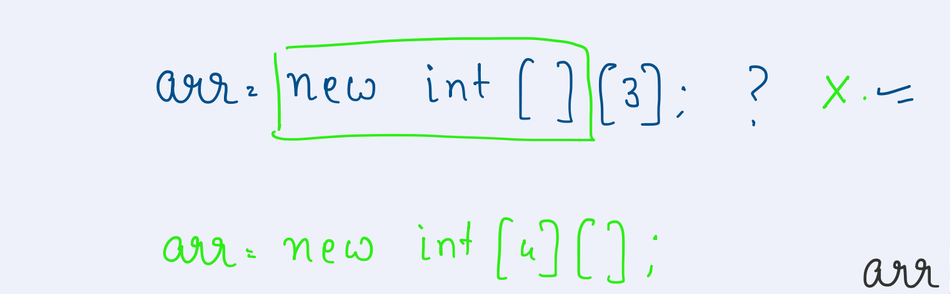


60K.
0 null
1 null
2 null → x.
3 null
arr=60K

```
Syso(arr[0]); //null
Syso(arr); 60K.
Syso(arr[2][1]); Error
```
null pointer exception.

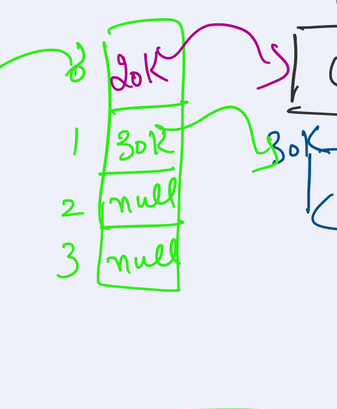
arr = new int[ ][3]; ? x.

arr = new int[4][ ];
arr[0] = new int[4];
arr[1] = new int[2];



20K 0 1 2 3
0 20K
1 30K
2 null
3 null
0 0 0 0
30K 0 1
0 0
arr:60K

```
Syso(arr) // 60K
Syso(arr[1]); // 30K
Syso(arr[2]); null.
Syso(arr[1][1]); //0
x. Syso(arr[2][1]); null pointer
```

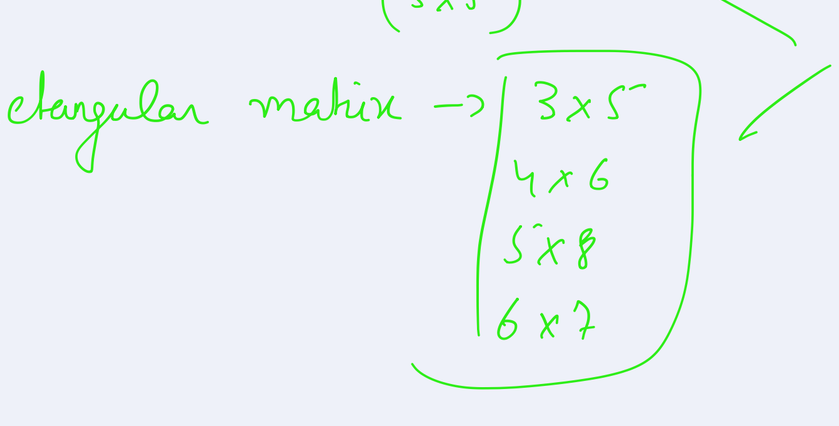


Square matrix → (3x3)
(4x4)
(5x5)

Rectangular matrix → 3x5
4x6
5x8
6x7

Questions

Q→



J
0 1 2 3
i
0 11 12 13 14
1 21 22 23 24
2 31 32 33 34
3 41 42 43 44
l to r → 0 Even
r to l. → odd -1
l to r → 2. Even.
r to l. → odd ∈ 3.

Wave Print → (Row-wise);
11, 12, 13, 14, 24, 23, 22, 21, 31, 32, 33, 34, 44, 43, 42, 41.

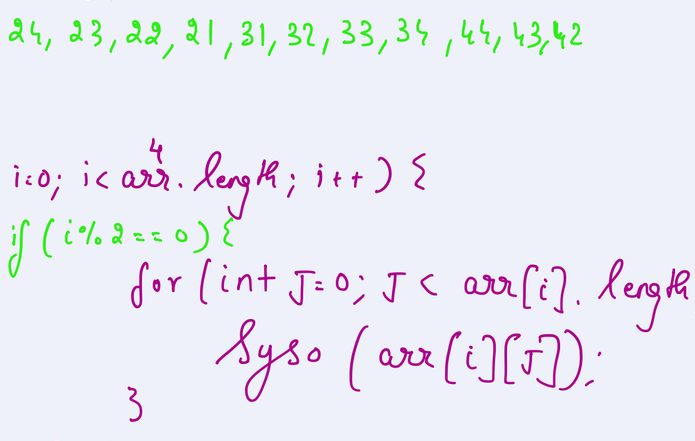
3x5

24, 23, 22, 21,
arr[1][3] arr[1][2] arr[1][1] arr[1][0].

```
for (int i=0; i<arr.length; i++) {
    if (i%2==0) {
        for (int J=0; J < arr[i].length; J++) {
            Syso (arr[i][J]);
        }
    } else {
        for (int J= arr[i].length-1; J>=0; J--) {
            Syso(arr[i][J]);
        }
    }
}
```

odd

(2,2)



left
dir=1
right
0 1 2 3
Q→ 0
11 12 13 14
21 22 23 24
31 32 33 34
41 42 43 44
bottom
top
dir=4
dir=2
dir=3
Spiral Print

11, 12, 13, 14, 24, 34, 44, 43, 42, 41, 31, 21, 22, 23, 33, 32.

(4x4) = ⑯
5x4 = ⑳

[2][0]
[1][0]

[3][2]
[3][1]
[3][0]

11 12 13 14.
24 34 44
43 42 41

(32)

(33)

```
public static void main(String[] args){
  int dir=1;
  int left=0; int top=0
  int right= arr[top].length-1;
  int bottom= arr.length-1;
  int Count=(bottom+1)*(right+1);   //16.
  while (left<=right && top<=bottom){
    if (Count>0){              (4>0)
      if(dir==1){              => 1.
        for (int i=left; i<=right; i++){
          Syso(arr[top][i]);   //[0][0] [0][1]
          Count--;
        }
      }
      dir=2.
      top++;
    }
    if (Count>0){              //(12>0)
      if(dir==2){
        for (int i=top; i<=bottom; i++){
          Syso(arr[i][right]);
          Count--;
        }
      }
      dir=3;
      right--;
    }
    if (Count>0) {
      if (dir==3){
        for(int i=right; i>=left; i--){
          Syso(arr[bottom][i]);
          Count--;
        }
      }
      bottom--;
      dir=4;
    }
    if (Count>0) {             x (6>0)
      if (dir==4){
        for (int i=bottom; i>=top; i--){
          Syso(arr[i][left]);
          Count--;
        }
      }
      dir=1;
      left++;
    }
  }
```

(1<=2 && 1<=2).

0<=3 && 0<=3.

Count: ~~16~~ ~~15~~ ~~12~~ ~~11~~ ~~10~~ ~~9~~ ~~8~~ ~~7~~ ~~6~~ ~~5~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0

9 > 0

[1][2]
[1][2]

(1 && 2)

(1x4) => ④

```
if (Count>0) {
    for (———)
```

left right
0 1 2 3
11, 12, 13, 14.

bottom
~~top→4~~

(11, 12, 13, 14)

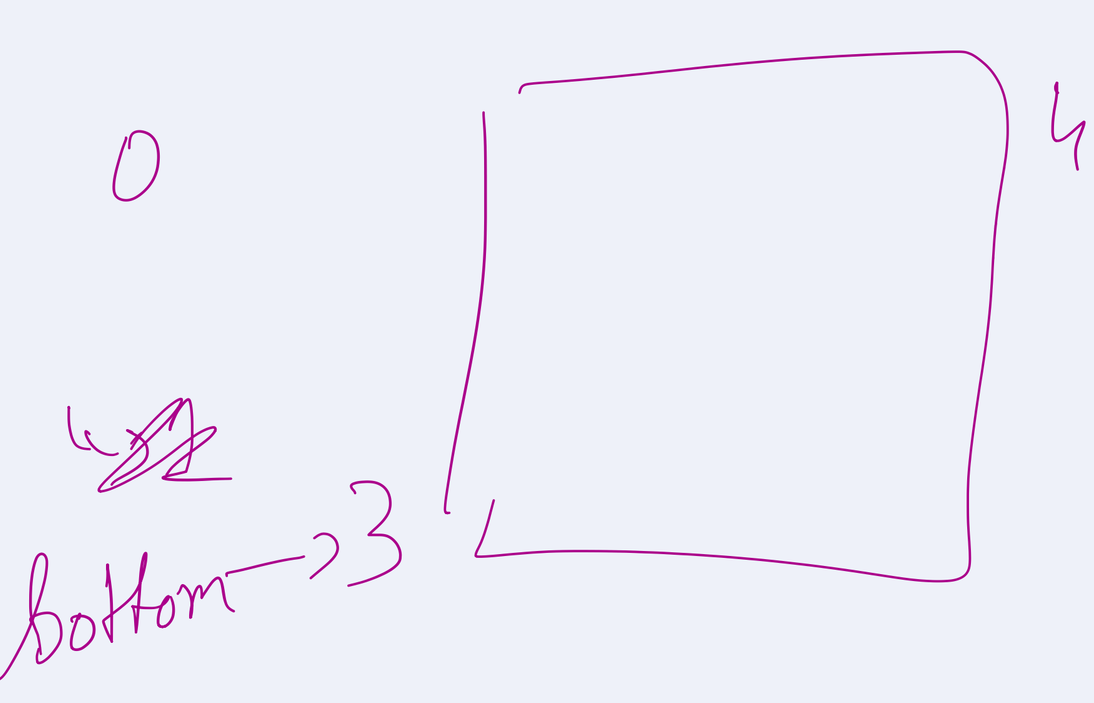
for (___)

14, 13, 12, 11

for ( ) {

for ( ) {

4x5



0
4
right
4
bottom→3

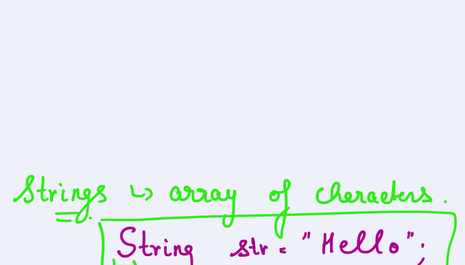
(3+1)*(4+1)

Resume at 8:42 ⇒
→ Strings

Strings → array of characters.

String str = "Hello"; //non-primitive data-types.

String S1="Hi";
S1=Str;
→ 40K
S1= 40K



40K.
"Hello"
S1:40K
Str:40K

"Hello" Syso(s1);
"Hello" Syso(str);

0 1 2 3 4 5
"Hello"
0 1 2 3 4
"Hello"
→ 4th index
→ str → 5.

```
Syso (Str.charAt(0));  //H.
Syso (Str.charAt(3)); //l
Syso (Str.length()); // 5
Syso (Str.charAt(Str.length()));
```

Str.length() → function
arr.length. → property.

Hello

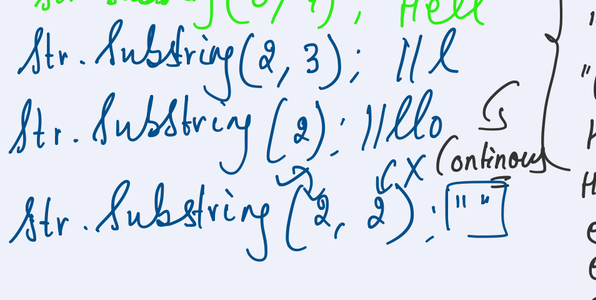
```
               beg   end. 5
Str.SubString (0, 3); // Hel
Str.Substring(0, 4); Hell
Str.Substring(2, 3); //l
Str.Substring(2); //llo  (continous)
Str.Substring(2, 2); ""
```

"Hello"
"H"
"He"
"Hel"
Hell
Hello
e
el
ell
ello
l
ll
llo
l
lo
o

```
Str.StartsWith("He"); //true.
Str.Starts with("he"); false;
```

* Strings are immutable in java. (imp).
→ Can't be changed

String str = "Hello";
Str= "Bye";

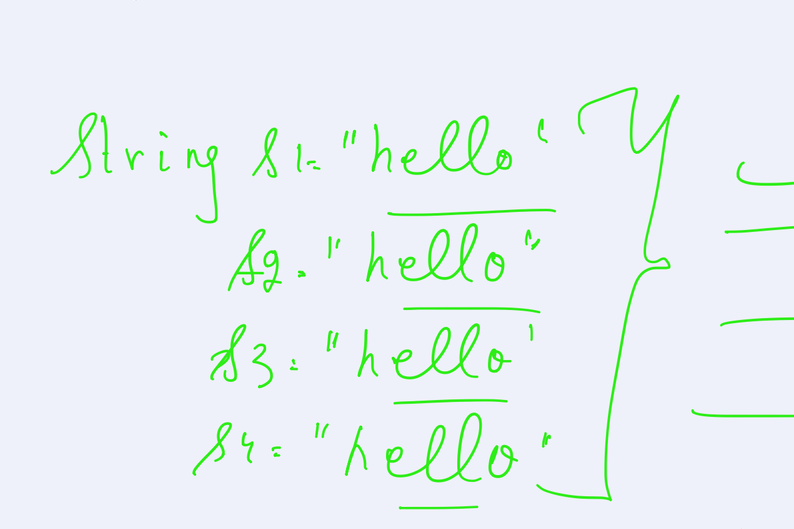


60K.
0 1 2 3 4
"Hello"
Str:100K
100K
"Bye"
100K

```
String S1="Hello"   -①
String S2="Hello";  -②
String S3=S1;       -③
String S4=new String("Hello"); ④
```

String Str="Hello";

String pool.



S5:80K
S4:200K
S3:80K
S2:80K
S1=80K
80K
0 1 2 3 4
"Hello"
200K
"Hello"

String S5="Hello";

== / equals.

```
        80K    80K
Syso(S1==S2); //true
Syso(S2==S3); // true.
Syso(S1==S4); false.
Syso(S1.equals(S2)). //true.
Syso(S1.equals(S4)); //true.
```

== address

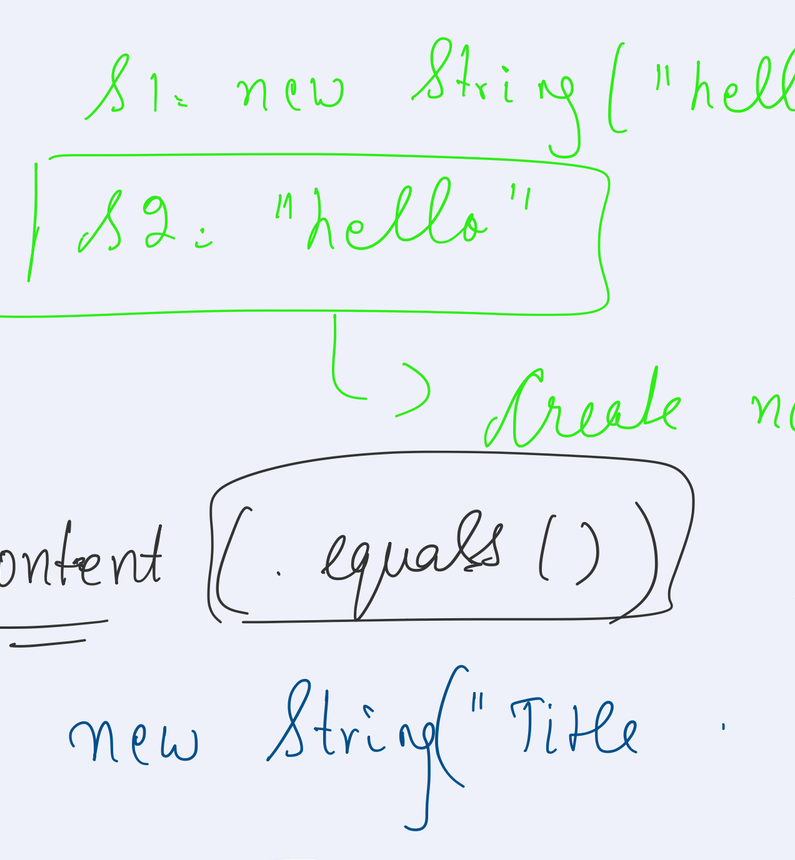
equals Content

Q: why they are immutable?
S2 = "~~H~~ Hello"

String S1="hello"
S2="hello"
S3="hello"
S4="hello"

S2 = "Hello"
S1, S2, S3, S5 => 80K
"Hello"

S1 "Hello" S3:"Hi"
S2:"Hi" 100K
S2:S1;



S3:~~100K~~
S2:100K
S1:100K
100K
"Hello
200K
"Hi"

S1= new String("hello");
S2= "hello"
→ Create new String in pool.

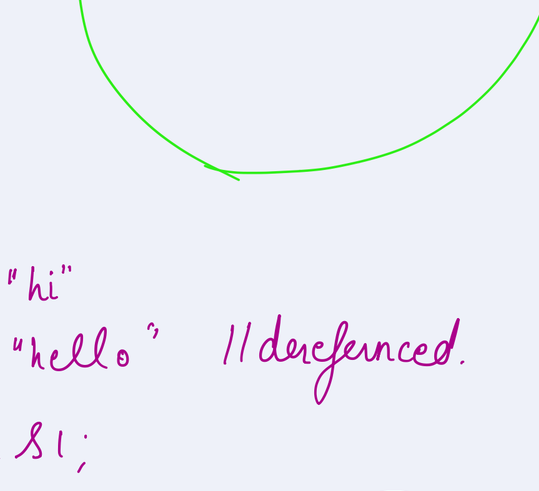
Content (.equals()) ==

new String(" Title

Syso(Str)
→ Referencing → memory

String[] Str= new String
char[] char= ?

S1:"Hello"
S2:"Hello" → 200K



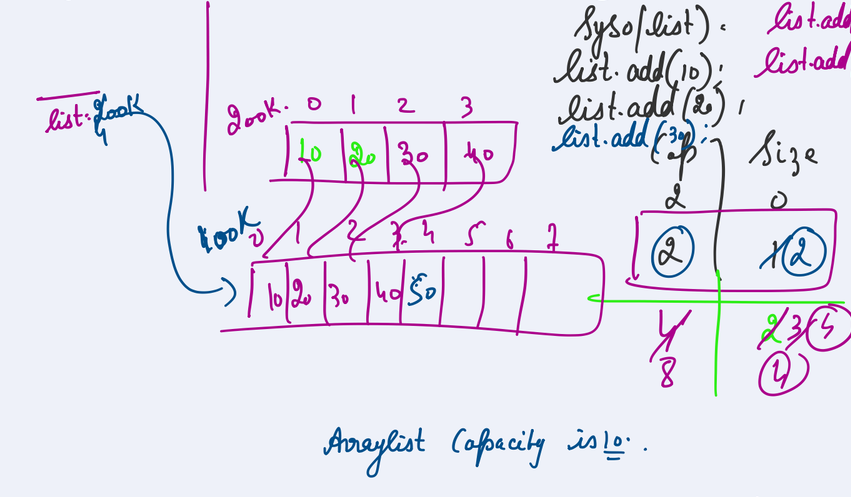
S1:"hi"
S2:"hello" //dereferenced.
S2:S1;
S3:"hello" } → new

Arrays → 1-D
→ 2-D } → Size is fixed.
→ Strings

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

Capacity ② → Max
Size (elements present)

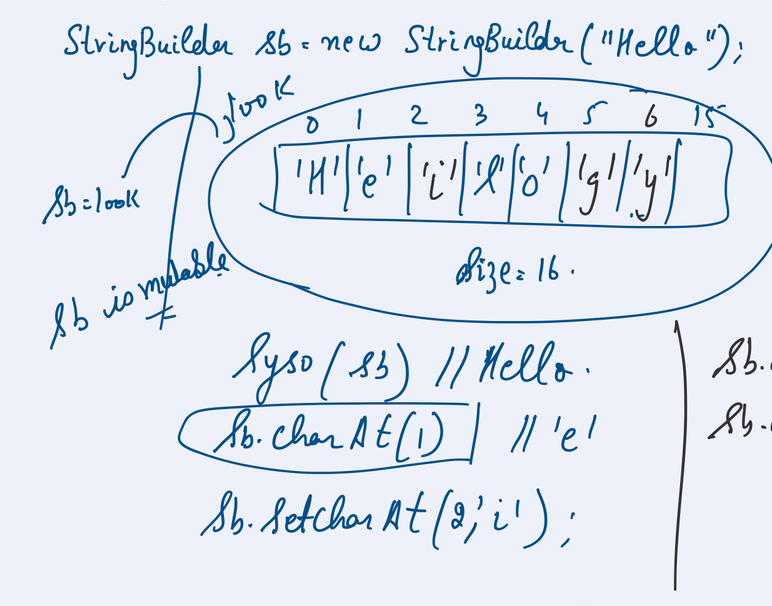
```
Syso(list).
list.add(10);
list.add(20);
list.add(30);
list.add(10)
list.add(50)
```



list:100K
200K. 0 1 2 3
10 20 30 40
100K 0 1 2 3 4 5 6 7
10 20 30 40 50
cap
size
2 0
② 1②
~~4~~ 8
~~2~~ 3 ⑤
④

Arraylist Capacity is 10.

StringBuilder Sb = new StringBuilder("Hello");



100K
0 1 2 3 4 5 6 15
'H' 'e' 'i' 'l' 'o' 'g' 'y'
Size=16.
Sb=100K

Sb is mutable

```
Syso(Sb) // Hello.
Sb.charAt(1) // 'e'
Sb.SetCharAt(2,'i');
Sb.append('g');
Sb.append("y");
```

Arrays, Strings, 2-D Arrays, Arraylist StringBuilder.

Sb.toString → ()
→ returns the String

Arrays → one D
→ two D
→ wednesday
Strings
Bit Masking