Day 7 - 2D Arrays + functions

Saturday, 23 October 2021 4:29 PM

✓ class Arrayclass → template ✓

①

✓ Public static void main(string [] args) {

↓Access specifiers | printMethod();

}

Public, static printMethod ( ) {

// - - - -

} ✓

Array obj1 = new Array();

obj1. PrintMethode();

}✓ }

Modifiers/Access specifiers

→ Public → access anywhere in my Project

→ Private → access only in the class which it is defined

→ Protected → only within its own class and its subClass

→ default → within its class which it is defined and ~~the~~ its Package.

class ~~ArrayClass~~ shapes {

Public Static void main(string [] args) {

call ✓ PrintMethod ("xyz"); → No arguments

→ Arguments → can have any no. of args

} ~~Protected~~ ✓ ~~Private~~ ✓

~~Public~~ static void PrintMethod (String name) {

↳ parameters

system.out.println("Hello");

}

}

```
class Add {  X not subclass

      PrintMethod ( ) X

}

      ✓                      ✓
class Rectangle extends Shapes {  → subclass
Base class  ✓PrintMethod ( ) ; X Not possible when declared as Private

}
                 ✓
```

}

✓ ✓ ✓

double double double

Returntypes → Private static int add(int num1, int num2){

=5 =>4

return num1+num2 ; //9

}

Public static void main(String[] args){

int value=add(5, 4);

✓ Parameters ✓

}

Public Static void PrintNameAndAge(String name, int age){
}

"xyz" 5

→ //print → xxx

}main

→ PrintNameAndAge("xyz", 5); ✓ → arguments

(5, "xyz"); X

calling name → X

Public static void methodName( ){

for(int i =1; i<=10; i++)

{ if(i==5) return; X return —; X

} }

}

Non-Static Methods

```
class Student {                    →not accessible
    Private String name; // xyz
non-static
    Public (X) String getname() {
        return name;
    }
    Public void setName(String ~~name~~ studentName) {
                                     ← xyz
        this.name = studentname
    }
}
```

X

Student.getName();

```
Student st1 = new Student();
    st1.setName("xyz");
System.out.Println(S1.getName()); //xyz
(st1.name)
```

Set Matrix Zeroes

i/p → matrix

	0	1	2	3
0	0	1	2	0
1	3	4	5	2
2	1	3	1	5

O/P :

0	0	0	0
0	4	5	0
0	3	1	0

entire matrix will become Zero X ✓ (X)

O/P: Approach 1 :

0	0	0	0	0
1	0	4	5	0
2	0	3	1	0

outputMatrix → (Extra space) m rows, n columns

↳ space O(mn) → reduce

boolean (T or F) O(m)

Approach 2:

int[] rowsFlag = new int [3]; rowFlag [1] = True } acts Flags

int [] colFlag = new int [4]; rowFlag [2] = true

rowFlag [0] = true

colFlag [2] = true.

i/p:

	0	1	2	3
0	0	0	0	0
1	0	4	5	0
2	0	3	1	0

rowFlag → boolean

0	1	2
T	F T	F

col Flag

0	1	2	3
T	F	F	T

X space

same space

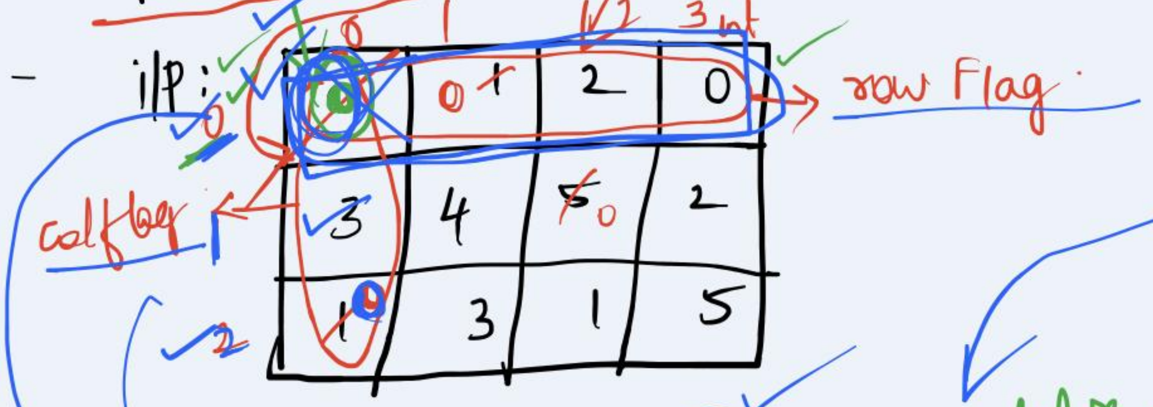
Approach 3:

space

same space

Approach 3:

i/p:

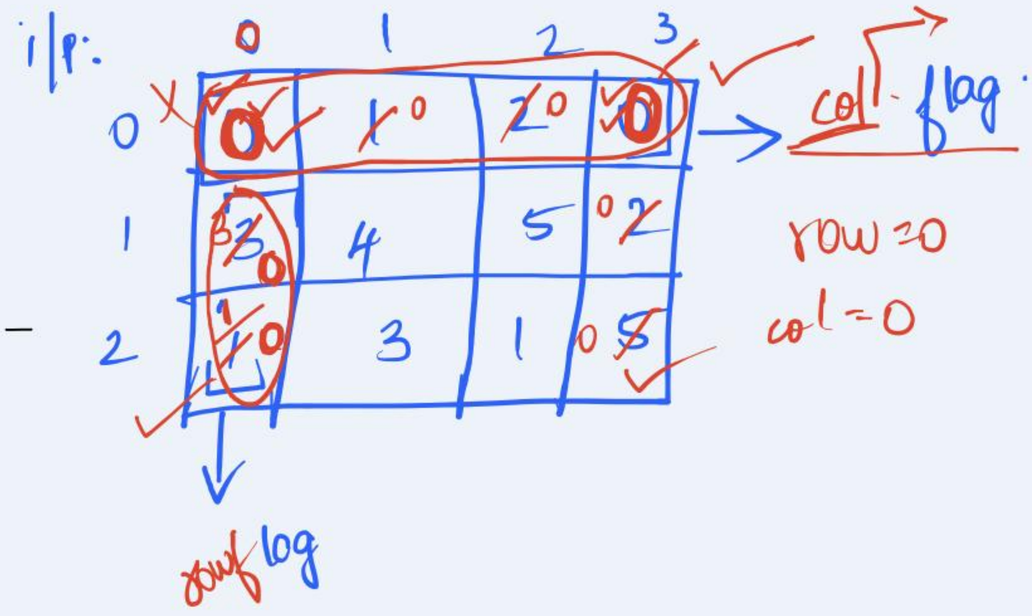


0	~~1~~ 0	2	0
3	4	~~5~~ 0	2
1 0	3	1	5

boolean isFirstColZero = false; → true

boolean isFirstRowZero = false;

i/p:



0	~~1~~ 0	~~2~~ 0	0
~~3~~ 0	4	5	0 ~~2~~
~~1~~ 0	3	1	0 ~~5~~

boolean isFirst ~~col~~ row zero = ~~false~~ True

row = 0
col = 0

row = 0
col = 3

Linear Search

0 1 2 3 4 5 6

nums [1, 8, 3, 4, 7, 80, 6]

target = ~~5~~ 8 10 -> $\boxed{O(n)}$

```
for(int i=0; i<nums.length; i++){
    if(nums[i] == target)
        return i;
}
return -1;
```

-1 -> 1

target is not present

target = 9

(1, 3, 4, 5, 9, 10)

index

O(n)

sorted → Binary search