74. Search a 2D Matrix

```java
class Solution {
    public boolean searchMatrix(int[][] arr, int target) {
        int n = arr.length;
        int m = arr[0].length;
        int si = 0;
        int ei = m * n - 1;
        while ( si <= ei ) {
            int mid = (si + ei) / 2;

            int curr = arr[mid / m][mid % m];
            if ( curr == target ) {
                return true;
            } else if ( curr < target ) {
                si = mid + 1;
            } else if ( curr > target ) {
                ei = mid - 1;
            }
        }
        return false;
    }
}
```

Indentation

⇒ Constructor cont.....

→ Copy Constructor

↳ A copy constructor in Java is used to create an object with the help of another object of the same class.

```java
public class Main {
    private int myInt;
    private String myString;

    // copy constructor
    public Main( Main obj ) {
        this.myInt = obj.myInt;
        this.myString = obj.myString;
    }

    // paramaterized contructor
    public Main(int myInt, String myString) {
        this.myInt = myInt;
        this.myString = myString;
    }

    public void displayValues() {
        System.out.println("myInt : " + myInt);
        System.out.println("myString : " + myString);
    }

    public static void main(String[] args) {
        Main originalObj = new Main(10, "Hello");
        Main copiedObj = new Main(originalObj);

        originalObj.displayValues();
        copiedObj.displayValues();
    }
}
```

output

```
myInt : 10
myString : Hello
```
para.

```
myInt : 10
myString : Hello
```
copy.

Imp :-

Copy constructors uses object of same class

⇒ Constructor chaining

Keyword this , this()

↳ Constructor chaining in java refers to the process of one constructor called another constructor from same class.

```java
public class Main {
    private int myInt;
    private String myString;

    public Main() {
        this(0, "Default");
    }

    public Main(int myInt) {
        this(myInt, "Default");
    }

    public Main(String myString) {
        this(0, myString);
    }

    public Main(int myInt, String myString) {
        this.myInt = myInt;
        this.myString = myString;
    }

    public void displayValues() {
        System.out.println("int value : " + myInt);
        System.out.println("String value : " + myString);
    }

    public static void main(String[] args) {
        Main obj1 = new Main();
        obj1.displayValues();

        Main obj2 = new Main(10);
        obj2.displayValues();

        Main obj3 = new Main("World");
        obj3.displayValues();

        Main obj = new Main(20, "Hello");
        obj.displayValues();
    }
}
```

Constructor chaining

what is the diff. b/w this and this()

⇒ Super Constructor

↳ In Java, the super() constructor call is used to invoke the constructor of a super class within a subclass constructor.

Imp

what is the diff. b/w super and super()

diff

super & this

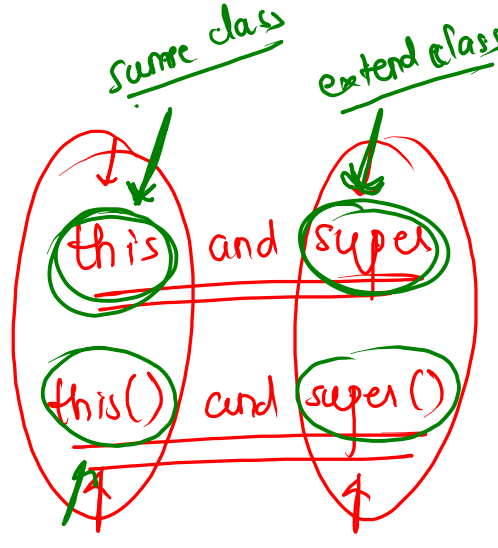
super() & this()

subclass means child class & super class means parent class.

```java
class SuperClass {
    private int superClassValue;

    // super contructor
    public SuperClass(int value) {
        superClassValue = value;
        System.out.println("Super class called");
    }

    public void displaySuperClass() {
        System.out.println("Super class value : " + superClassValue);
    }
}
```

```java
class SubClass extends SuperClass {
    private int subClassValue;

    public SubClass(int superClassValue, int subClassValue) {
        super(superClassValue); // calling superclass contructor
        this.subClassValue = subClassValue;
        System.out.println("Sub class called");
    }

    public void displaySubClassValue() {
        System.out.println("Sub class value : " + subClassValue);
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        SubClass obj = new SubClass(10, 20);
        obj.displaySuperClass();
        obj.displaySubClassValue();
    }
}
```



⇒ Access modifiers

↳ access modifiers are keywords that are used to control the accessibility or visibility of class, methods, variables

→ Types of access modifiers

- Default → No keyword required
- Private
- Protected
- Public

1) Default access modifier

↳ It provide package level access, means classes, methods and variables are accessible only within the same package.

All are in same package

```java
package library;

class Book {
    String title;
    String author;

    void displayInfo() {
        System.out.println(title);
        System.out.println(author);
    }
}
```

```java
package library;

class Library {
    // some methods and variables
}
```

```java
package library;

class Librarian {
    // some methods and variables
}
```

2) Private access modifier

↳ it restricts the access of a class member only to the class in which it is declared.

```java
public class Main {
    public static void main(String[] args) {
        BankAccount obj = new BankAccount("1234567");
        // obj.balance = "0"; // compilation error
        // obj.accountNumber = "132456"; // compilation error
        obj.deposit(500);
        obj.getBalance();
    }
}
```

```java
public class BankAccount {
    private String accountNumber;
    private double balance;

    public BankAccount(String accountNumber) {
        this.accountNumber = accountNumber;
        this.balance = 0.0;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) {
        balance -= amount;
    }

    public double getBalance() {
        return balance;
    }
}
```

⇒ Protected access modifier

↳ Here, in protected access modifier we can access in same packege and also in the extend child class.

```
public class Main {
    public static void main(String[] args) {
        Student obj = new Student();
        obj.name;
    }
}

public class Person {
    protected String name;

    protected void introduce() {
        // introducing
    }
}

// in different package
public class Student extends Person {

}
```

⇒ Public access modifier

↳ here we can access class, methods and variables anywhere regardless of package or class.

```
public class Main {
    public static void main(String[] args) {
        Person obj = new Person();
        obj.name;
    }
}
```

```
public class Person {
    public String name;

    public void introduce() {
        // introducing
    }
}
```

Most Restrictive ← → Least Restrictive

Access Modifiers ->	private	Default/no-access	protected	public
Inside class	Y	Y	Y	Y
Same Package Class	N	Y	Y	Y
Same Package Sub-Class	N	Y	Y	Y
Other Package Class	N	N	N	Y
Other Package Sub-Class	N	N	Y	Y

1011. Capacity To Ship Packages Within D Days

T.C =

O(n log(n))

```java
public int shipWithinDays(int[] weights, int days) {
    int totalWeight = 0;
    int maxWeight = 0;
    for (int i: weights) {
        totalWeight += i;
        maxWeight = Math.max(i, maxWeight);
    }
    int si = maxWeight, ei = totalWeight;
    int ans = ei;
    while (si <= ei) {
        int capacity = si + (ei - si) / 2;
        if (check(weights, capacity, days)) {
            ans = capacity;
            ei = capacity - 1;
        } else {
            si = capacity + 1;
        }
    }
    return ans;
}

public boolean check(int[] weights, int capacity, int days) {
    int day = 1, weight = 0;
    for (int i: weights) {
        weight += i;
        if (weight > capacity) {
            weight = i;
            day++;
        }
    }
    if (day > days) return false;
    return true;
}
```