teach me how to see contraints (understand) in leetcode in java

Great question—**reading constraints is one of the most underrated LeetCode skills**, and it directly affects **which algorithm you choose** and **how you code it in Java**.
I'll teach you a **clear, repeatable way** to *understand constraints* and convert them into decisions.

1️⃣ What are "constraints" really?

Constraints answer **three critical questions**:

1. **How big is the input?** → algorithm choice
2. **What values can appear?** → data types & edge cases
3. **How many test cases / operations?** → time & memory safety

Example constraint section:

```css
1 ≤ n ≤ 10^5
-10^9 ≤ nums[i] ≤ 10^9
```

Copy code

2️⃣ The MOST IMPORTANT rule (memorize this)

👉 Input size decides time complexity

`n` (max size)	Allowed Time Complexity
`n ≤ 10^2`	$O(n^3)$ or worse
`n ≤ 10^3`	$O(n^2)$
`n ≤ 10^5`	$O(n \log n)$ or $O(n)$

`n` (max size)	Allowed Time Complexity
`n ≤ 10^6`	O(n)
`n ≤ 10^7+`	O(n) or O(1) only

💡 **Java rule of thumb:**

~10^7 **operations per second** is safe

3️⃣ Step-by-step: How to READ constraints (LeetCode style)

Example Problem

```css
1 ≤ nums.length ≤ 10^5
-10^4 ≤ nums[i] ≤ 10^4
```

Copy code

Step 1: Look at the size

```matlab
nums.length ≤ 10^5
```

Copy code

❌ O(n^2) → 10^{10} operations → TLE

✅ O(n) or O(n log n)

Instant decision:

> "I need a linear or log-linear solution"

Step 2: Look at value range

```css
-10^4 ≤ nums[i] ≤ 10^4
```

Copy code

This tells you:

- `int` is safe ✓
- Possible **negative values** → beware of indexing
- Small range → maybe **counting array / frequency array**

Step 3: Combine both

Observation	Meaning
Large `n`	Avoid nested loops
Small value range	Use arrays instead of HashMap
Negative numbers	Handle indexing carefully

4️⃣ Constraints → Data Structure choice

A. When to use Array

```css
-1000 ≤ nums[i] ≤ 1000
```

Copy code

✓ Use `int[] freq = new int[2001];`

B. When to use HashMap

```css
-10^9 ≤ nums[i] ≤ 10^9
```

Copy code

✓ Values too large → HashMap

C. When to use Set

```sql
Check duplicates
```

Copy code

✔️ Use `HashSet`

D. When to use Sorting

```css
Order matters + n ≤ 10^5
```

Copy code

✔️ `Arrays.sort()` → O(n log n)

5️⃣ Constraints → Data type choice (Java)

🚨 VERY common mistake

```css
1 ≤ n ≤ 10^5
1 ≤ nums[i] ≤ 10^9
```

Copy code

If you do:

```java
int sum += nums[i];
```

Copy code

❌ Overflow!

Correct:

```java
long sum = 0;
```

Copy code

Rule:

Constraint	Use
$\le 10^9$	int
Multiplication / sum	long
$\le 10^{18}$	long

6️⃣ Constraints → Edge cases

Constraints hint **what to test**:

```
1 ≤ n ≤ 10^5
```

Copy code

Edge cases:

- `n = 1`
- `n = max`
- All same values
- All negative / positive

7️⃣ Example Walkthrough (Classic)

🔹 Two Sum

```css
2 ≤ nums.length ≤ 10^4
-10^9 ≤ nums[i] ≤ 10^9
```

Copy code

Thinking:

- n = 10^4 → $O(n^2)$ barely ok, but risky
- Values large → no array indexing
- Need fast lookup → HashMap

Final choice:

```java
Map<Integer, Integer> map = new HashMap<>();
```

Copy code

8️⃣ Golden 5-step checklist (use EVERY problem)

When you open a LeetCode problem:

1. 🔍 **Max input size?**
2. ⏱️ **Allowed time complexity?**
3. 🧠 **Best data structure?**
4. 🔢 **int or long?**
5. ⚠️ **Edge cases from constraints?**

If you answer these, **80% of the problem is already solved**.

9️⃣ Want to practice this skill?

If you want, I can:

- Walk through **1 real LeetCode problem** step by step
- Teach **constraint → algorithm patterns**
- Give you **constraint decoding exercises**
- Show **Java-specific pitfalls**

Just tell me 👍