**My current Problems at problem solving,**

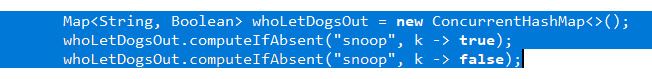
1. **I am scared of starting or making theoretical solution. That’s why I always lose time at the beginning. I have to make this part somehow interesting or do it to a certain level that it gets interesting.**
2. **I have problem on demonstrating problem on pen and paper. I cannot imagine that clear.**

It is always based on input, write down all the input. Don’t thing about optimization at the beginning. First try to make it work anyway.

1. **I have even big problem on improve solution for time and space complexity on pen and paper**
2. **Without debugging, I cannot manage complex problem yet. I have to find a way solve most of the problem without debugging.**
3. **I cannot consider corner cases very well.**
4. **I take extremely lot of time to solve a problem. I had to emotionally connect with a problem. Probably because of lack of understanding.**
5. **I take long time to understand a problem, even if it is small problem. Sometimes one day I just understand the problem and only next day I start to solve the problem. I do not know why. I have to find out the reason and overcome it.**

**Map**

**Map.computeIfAbsent** will insert value inside the map, if this key is not mapped with a value or not mapped with null. Here “snoop” is the key and true or false will be value.

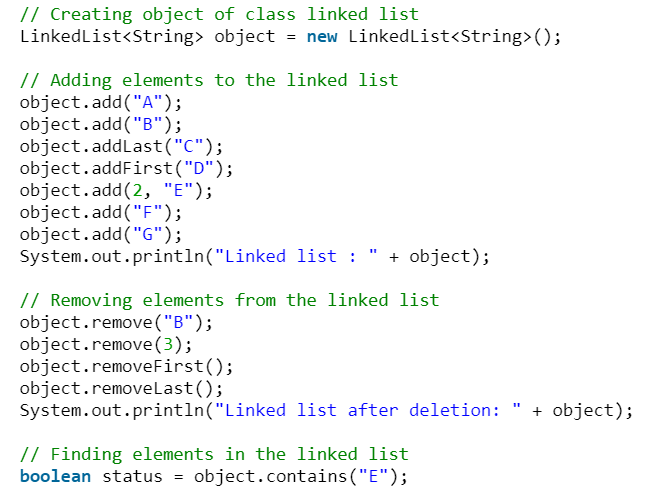


**Linked List**

Very interestingly, it is possible to add any elements in linked list to put element at any location of the list. And the connected elements are scaled with this insert.

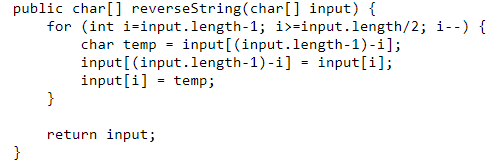
<https://www.geeksforgeeks.org/linked-list-in-java/>

**Graph**

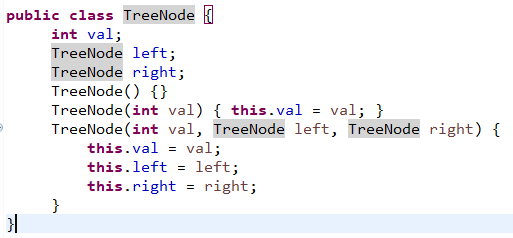


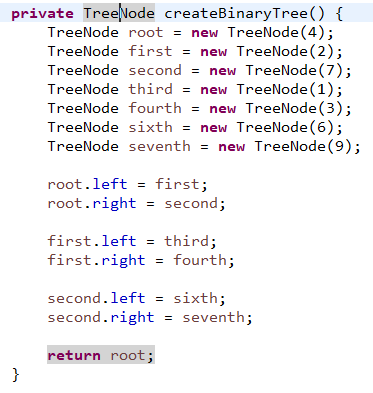
**Reverse an array**

It can reduce the cost from N to logn



**Binary Tree**





**Asserting class if they have equals value**

**Knapsack Problem:**

Sample Input: [[1, 2], [4, 3], [5, 6], [6, 7]] and capacity 10 kg

Sample Output: [10, [1, 3]]

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| [ ] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| [1,2] | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| [4,3] | 0 | 0 | 1 | 4 | 4 | 5 | 5 | 5 | 5 | 5 | 5 |
| [5,6] | 0 | 0 | 1 | 4 | 4 | 5 | 5 | 5 | 6 | 9 | 9 |
| [6,7] | 0 | 0 | 1 | 4 | 4 | 5 | 5 | 6 | 6 | 9 | 10 |

Max(valuesTable[i-1] [j], items[v]+valuesTable[i-1][w-item[w])

Max(9, 6+4) [6,7][4,3]