Monokai AlgoExpert **Quad Layout** Sublime 00:00:00 Java **12px** 

Prompt Run Code

```
Scratchpad
              Our Solution(s)
```

**Video Explanation** 

```
Solution 1
```

```
1\, // Copyright @ 2020 AlgoExpert, LLC. All rights reserved.
   import java.util.*;
 5 class Program {
     // O(nc) time | O(nc) space
      public static List<List<Integer>> knapsackProblem(int[][] items, int capacity) {
       int[][] knapsackValues = new int[items.length + 1][capacity + 1];
        for (int i = 1; i < items.length + 1; i++) {</pre>
          int currentWeight = items[i - 1][1];
10
11
          int currentValue = items[i - 1][0];
12
          for (int c = 0; c < capacity + 1; c++) {
           if (currentWeight > c) {
13
             knapsackValues[i][c] = knapsackValues[i - 1][c];
14
15
              knapsackValues[i][c] =
16
17
                 Math.max(
18
                     knapsackValues[i - 1][c],
                      knapsackValues[i - 1][c - currentWeight] + currentValue);
19
20
21
22
23
       return getKnapsackItems(knapsackValues, items, knapsackValues[items.length][capacity]);
24
25
26
      public static List<List<Integer>>> getKnapsackItems(
         int[][] knapsackValues, int[][] items, int weight) {
27
28
       List<List<Integer>> sequence = new ArrayList<List<Integer>>();
29
       List<Integer> totalWeight = new ArrayList<Integer>();
30
       totalWeight.add(weight);
31
       sequence.add(totalWeight);
       sequence.add(new ArrayList<Integer>());
32
       int i = knapsackValues.length - 1;
33
34
        int c = knapsackValues[0].length - 1;
35
       while (i > 0) {
          if (knapsackValues[i][c] == knapsackValues[i - 1][c]) {
36
37
           i--;
38
          } else {
39
           sequence.get(1).add(0, i - 1);
           c -= items[i - 1][1];
40
41
           i--;
42
43
          if (c == 0) {
44
           break;
45
46
       return sequence;
47
48
49 }
```