```
/** Third algorithm - Dynamic programming */
 public void dynamicAlgorithm() {
     int[][] table = new int[totalItem + 1][capacity + 1];

     // construct the table
     for (int i = 1; i < totalItem + 1 ; i ++){
         table[i][0] = 0; // Base case
         for (int j = 1; j < capacity+1 ; j++) {
             table [0][j] = 0;  // Base case
             // able to pick item
             if (itemArray[i-1].weight <= j)
                 //Pick the max of {this Item's value + total value of space left , total value from previous }
                 table[i][j] = Math.max(itemArray[i-1].value +
                                        table[i-1][j - itemArray[i-1].weight],
                                        table[i-1][j]);
             // Not able to pick item
             else
                 table[i][j] = table[i-1][j];
         }
     }

     // Trace back, start from last cell of table (containing the totalValue)
     int[] resultIndex = new int[totalItem];
     int rowIndex = totalItem;
     int columnIndex = capacity;
     int totalValue = table[rowIndex][columnIndex];
     int totalWeight = 0;
     while (rowIndex != 0 && columnIndex != 0) {
         if (table[rowIndex][columnIndex] != table[rowIndex-1][columnIndex]) {
             resultIndex[rowIndex-1] = rowIndex;
             columnIndex -= itemArray[rowIndex-1].weight;
             totalWeight += itemArray[rowIndex-1].weight;
         }
         rowIndex -= 1;
     }
     // Print result
     System.out.println("Dynamic Programming solution: "
                        + totalValue + " " + totalWeight);
     for (int i : resultIndex)
         if (i != 0)
             System.out.print(i + " " );
     System.out.println();
 }
```