class Knapsack {

public int solveKnapsack(int[] profits, int[] weights, int capacity) {

return this.knapsackRecursive(profits, weights, capacity, 0);

}

private int knapsackRecursive(int[] profits, int[] weights, int capacity, int currentIndex) {

// base checks

if (capacity <= 0 || currentIndex >= profits.length)

return 0;

// recursive call after choosing the element at the currentIndex

// if the weight of the element at currentIndex exceeds the capacity, we shouldn't process this

int profit1 = 0;

if( weights[currentIndex] <= capacity )

profit1 = profits[currentIndex] + knapsackRecursive(profits, weights,

capacity - weights[currentIndex], currentIndex + 1);

// recursive call after excluding the element at the currentIndex

int profit2 = knapsackRecursive(profits, weights, capacity, currentIndex + 1);

return Math.max(profit1, profit2);

}

public static void main(String[] args) {

Knapsack ks = new Knapsack();

int[] profits = {1, 6, 10, 16};

int[] weights = {1, 2, 3, 5};

int maxProfit = ks.solveKnapsack(profits, weights, 7);

System.out.println("Total knapsack profit ---> " + maxProfit);

maxProfit = ks.solveKnapsack(profits, weights, 6);

System.out.println("Total knapsack profit ---> " + maxProfit);

}

}

