```
Enter File Name:
C:\Users\Jamie\Desktop\AI Files\file7
Furniture: 1 3 7

Weight: 48

Value: 1800

-----------------------------------------------------------------
Enter File Name:
C:\Users\Jamie\Desktop\AI Files\file8
Furniture: 6 9 10 12 13 16 17

Weight: 97

Value: 10800
```

```java
public class Furniture {

    private int weight;             //Individual Array
    private int value;              //Individual Value
    private boolean active;         //If in truck

    //Constructor
    public Furniture(int weight, int value){
        this.weight = weight;
        this.value = value;
        active = false;
    }

    //Copy Constructor
    public Furniture(Furniture other){
        this.weight = other.getWeight();
        this.value = other.getValue();
        this.active = other.isActive();
    }

    public int getValue() {
        return value;
    }

    public int getWeight() {
        return weight;
    }

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }
}

```

```java
import java.util.LinkedList;

//Solves Furniture Problem
public class FurnitureSolver {

    //Truck class
    private class Truck{
        private Furniture[] shipment;           //shipment list
        private int weight;                     //total weight
        private int value;                      //total value
        private int lastChecked;                //last checked item

        //Constructor
        private Truck(int size, int[][] listOfFurniture){
            shipment = new Furniture[size];
            for(int i = 0; i < size; i++)
                shipment[i] = new Furniture(listOfFurniture[i][1], listOfFurniture[i][2]);

            weight = 0;
            value = 0;
            lastChecked = -1;                   //-1 means not checked
        }

        //Copy constructor
        private Truck(Truck other){
            this.shipment = new Furniture[other.shipment.length];
            for(int i = 0; i < this.shipment.length; i++)
                this.shipment[i] = new Furniture(other.shipment[i]);

            this.weight = other.weight;
            this.value = other.value;
            this.lastChecked = other.lastChecked;
        }

        public int getValue() {
            return value;
        }

        public int getLastChecked() {
            return lastChecked;
        }

        public void setLastChecked(int lastChecked) {
            this.lastChecked = lastChecked;
        }

        public void setActive(int toBeSet){
            shipment[toBeSet].setActive(true);

            weight += shipment[toBeSet].getWeight();
            value += shipment[toBeSet].getValue();
        }

        public int getWeight() {
            return weight;
        }

        public void display(){
            System.out.print("Furniture: ");
            for(int i = 0; i < shipment.length; i++)
                if(shipment[i].isActive())
                    System.out.print(i+1 + " ");
            System.out.println("\n");
            System.out.println("Weight: " + weight + "\n");
            System.out.println("Value: " + value + "\n");

        }
    }

    private int size;
    private int weightLimit;
    private int[][] listOfFurniture;

    //Constructor
    public FurnitureSolver(int size,int weightLimit, int[][] listOfFurniture){
        this.size = size;
        this.listOfFurniture = listOfFurniture;             //Only copies reference
        this.weightLimit = weightLimit;
    }

    //solves furniture problem
    public void solve(){
        LinkedList<Truck> list = new LinkedList<>();        //List of trucks
        int maxValue = Integer.MIN_VALUE;                   //max value
        Truck highestValueTruck = null;

        Truck truck = new Truck(size, listOfFurniture);
        list.addFirst(truck);                               //Create and add first

        while (!list.isEmpty()){                            //While has trucks
            truck = list.removeFirst();                     //Remove first

            if(complete(truck)){                            //If complete truck
                if(truck.getValue() > maxValue){            //If highest value
                    maxValue = truck.getValue();            //Update max
                    highestValueTruck = truck;
                }
            }
            else{                                           //If incomplete
                //generate children
                LinkedList<Truck> children = generate(truck);

                for (int i = 0; i < children.size(); i++)
                    list.addFirst(children.get(i));         //Add children to list
            }
        }
        if (highestValueTruck == null)
            System.out.println("No Solution");              //If no solution
        else
            highestValueTruck.display();        //Display highest value truck
    }

    //Generates Children
    private LinkedList<Truck> generate(Truck truck){
        LinkedList<Truck> children = new LinkedList<>();    // children list

        int lastChecked = truck.getLastChecked();

        //Add this furniture
        Truck childPositive = new Truck(truck);
        //Update last checked
        childPositive.setLastChecked(lastChecked+1);
        //set furniture active
        childPositive.setActive(lastChecked+1);

        //add if not over weight limit
        if(childPositive.getWeight() < weightLimit)
            children.addLast(childPositive);

        //Don't add this furniture
        Truck childNegative = new Truck(truck);
        //Update last checked
        childNegative.setLastChecked(lastChecked+1);
        //Don't need to check weight limit
        children.addLast(childNegative);

        return children;
    }

    //checks if truck is complete
    boolean complete(Truck truck){
        if (truck.getLastChecked() == size-1)
            return true;
        return false;
    }
}
```

```java
import java.util.Scanner;
import java.io.*;

public class FurnitureTester {
    public static void main(String[] args)throws IOException{
        Scanner keyIn = new Scanner(System.in);

        System.out.println("Enter File Name:");
        String fileName = keyIn.nextLine();

        File file = new File(fileName);
        Scanner sc = new Scanner(file);


        int numOfFuniture = Integer.parseInt(sc.nextLine());

        sc.nextLine();


        int[][] listOfFurniture = new int[numOfFuniture][3];

        for(int i = 0; i < numOfFuniture; i++){
            String line[] = sc.nextLine().split("\\s+");
            listOfFurniture[i][0] = Integer.parseInt(line[0]);
            listOfFurniture[i][1] = Integer.parseInt(line[1]);
            listOfFurniture[i][2] = Integer.parseInt(line[2]);
        }
        sc.nextLine();
        int weightLimit = Integer.parseInt(sc.nextLine());
        FurnitureSolver s = new FurnitureSolver(numOfFuniture,
weightLimit,listOfFurniture);
        s.solve();

    }
}
```