0 / 1 Knapsack in Recursive method

Java

Code :

```
package DP;
import java.util.*;

public class knapsack01recur{

    static int Knap(int[] wt,int[] val,int w,int n,int[][] dp){

        // When the bag has no capacity (w=0)
        //all the items have been iterated (n-- korte korte zero hoeche)
        //no items are there to collect (n=0)
        if(w==0 || n==0){
            return 0;
        }
        // If the element is present in our dp array
        if(dp[n][w]!= -1){
            return dp[n][w];
        }
        // An item is present that can be carried in the bag
        if(wt[n-1]<=w){
            dp[n][w] = Math.max( val[n-1]+Knap(wt, val, w-wt[n-1], n-1, dp),
Knap(wt, val, w, n-1, dp));
            return dp[n][w];
        }
        //An item is present that cannot be carried in the bag
        else{
            dp[n][w] = Knap(wt,val,w,n-1,dp);
            return dp[n][w];
        }

    }


    public static void main(String[] args) throws Exception{
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the value of n : ");
        int n = sc.nextInt();
        int[] wt = new int[n];
        int[] val = new int[n];
        for(int i= 0;i<n;i++) wt[i] = sc.nextInt();
        for(int i= 0;i<n;i++) val[i] = sc.nextInt();
        System.out.print("Enter the weight of allowed in the bag(w) : ");
```

```
        int w = sc.nextInt();
        int[][] dp = new int[n+1][w+1];

        for(int[] i : dp ){
            Arrays.fill(i,-1);
        }

        System.out.println(Knap(wt,val,w,n,dp));
        for(int[] i: dp)
            System.out.println(Arrays.toString(i));
    }
}
```

Output sample:

```
Enter the value of n : 4
2 3 4 5
10 20 30 40
Enter the weight of allowed in the bag(w) : 7
50
[-1, -1, -1, -1, -1, -1, -1, -1]
[-1, -1, 10, 10, 10, -1, -1, 10]
[-1, -1, 10, 20, -1, -1, -1, 30]
[-1, -1, 10, -1, -1, -1, -1, 50]
[-1, -1, -1, -1, -1, -1, -1, 50]
PS E:\code\practice>
```