```java
package knapsac;

/* A Naive recursive implementation
of 0-1 Knapsack problem */
public class Knapsac {

    // A utility function that returns
    // maximum of two integers
    static int max(int a, int b)
    {
        return (a > b) ? a : b;
    }

    // Returns the maximum value that
    // can be put in a knapsack of
    // capacity W
    static int knapSack(int W, int wt[], int val[],
int n)
    {
        // Base Case
        if (n == 0 || W == 0)
            return 0;

        // If weight of the nth item is
        // more than Knapsack capacity W,
        // then this item cannot be included
        // in the optimal solution
        if (wt[n - 1] > W)
            return knapSack(W, wt, val, n - 1);

            // Return the maximum of two cases:
            // (1) nth item included
            // (2) not included
        else
            return max(val[n - 1]
                               + knapSack(W - wt[n - 1
], wt,
                       val, n - 1),
                       knapSack(W, wt, val, n - 1));
    }

    // Driver code
    public static void main(String args[])
    {
        int val[] = new int[] { 60, 100, 120 };
```

```
44          int wt[] = new int[] { 10, 20, 30 };
45          int W = 50;
46          int n = val.length;
47          System.out.println(knapSack(W, wt, val, n));
48      }
49  }
50
51
```