

```
import java.util.Arrays;

public class QuickSort {

    private int temp_array[];

    private int len;

    public void sort(int[] nums) {

        if (nums == null || nums.length == 0) {
            return;
        }

        this.temp_array = nums;
        len = nums.length;
        quickSort(0, len - 1);
    }

    private void quickSort(int low_index, int high_index) {

        int i = low_index;
        int j = high_index;

        // calculate pivot number
        int pivot = temp_array[low_index+(high_index-low_index)/2];

        // Divide into two arrays
        while (i <= j) {
            while (temp_array[i] < pivot) {
                i++;
            }

            while (temp_array[j] > pivot) {
                j--;
            }

            if (i <= j) {
                exchangeNumbers(i, j);
            }
        }
    }

    private void exchangeNumbers(int i, int j) {
        int temp = temp_array[i];
        temp_array[i] = temp_array[j];
        temp_array[j] = temp;
    }
}
```

```

        //move index to next position on both sides

        i++;

        j--;

    }

}

// call quickSort() method recursively
if (low_index < j)
    quickSort(low_index, j);
if (i < high_index)
    quickSort(i, high_index);
}

private void exchangeNumbers(int i, int j) {
    int temp = temp_array[i];
    temp_array[i] = temp_array[j];
    temp_array[j] = temp;
}

// Method to test above
public static void main(String args[])
{
    QuickSort ob = new QuickSort();
    int nums[] = {7, 5, 3, 2, 1, 0, 45};
    System.out.println("Original Array:");
    System.out.println(Arrays.toString(nums));
    ob.sort(nums);
    System.out.println("Sorted Array");
    System.out.println(Arrays.toString(nums));
}
}

```

```

import java.util.Scanner;

public class Knapsack
{
    static int max(int a, int b)
    {
        return (a > b)? a : b;
    }

    static int knapSack(int W, int wt[], int val[], int n)
    {
        int i, w;

        int [][]K = new int[n+1][W+1];

        // Build table K[][] in bottom up manner
        for (i = 0; i <= n; i++)
        {
            for (w = 0; w <= W; w++)
            {
                if (i==0 || w==0)
                    K[i][w] = 0;
                else if (wt[i-1] <= w)
                    K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);
                else
                    K[i][w] = K[i-1][w];
            }
        }

        return K[n][W];
    }

    public static void main(String args[])

```

```

{
    Scanner sc = new Scanner(System.in);

    System.out.println("Enter the number of items: ");

    int n = sc.nextInt();

    System.out.println("Enter the items weights: ");

    int []wt = new int[n];

    for(int i=0; i<n; i++)

        wt[i] = sc.nextInt();


    System.out.println("Enter the items values: ");

    int []val = new int[n];

    for(int i=0; i<n; i++)

        val[i] = sc.nextInt();


    System.out.println("Enter the maximum capacity: ");

    int W = sc.nextInt();


    System.out.println("The maximum value that can be put in a knapsack of capacity W is: " +
knapSack(W, wt, val, n));

    sc.close();
}
}

```