```
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;
import java.util.Stack;

public class Assignment5 {

    public static void main(String[] args) {
        System.out.println("Max Increasing Subsequences:");
            System.out.println("[1,14,5,6,2,3]: " + sumIncreasingSubsequence(new int[]
{1,14,5,6,2,3}));

            System.out.println("input-5.1.txt: " + sumIncreasingSubsequence(inputFile("input-
5.1.txt")));

            System.out.println("input-5.2.txt: " + sumIncreasingSubsequence(inputFile("input-
5.2.txt")));

            System.out.println("input-5.3.txt: " + sumIncreasingSubsequence(inputFile("input-
5.3.txt")));

            System.out.println("input-5.4.txt: " + sumIncreasingSubsequence(inputFile("input-
5.4.txt")));
    }

     /** reads the file and converts it to an int array */
    public static int[] inputFile(String filename) {
        try (Scanner console = new Scanner(new FileReader(filename))) {
            //size is the first number
            final int n = console.nextInt();

            //reads ints in the file
            int[] A = new int[n];
            for (int i = 0; i < n; i++)
                A[i] = console.nextInt();

            return A;
        }catch(FileNotFoundException e) {
            System.err.println("File not found: '" + filename + "'");
        }

        return null;
    }

    /** calculates the sum of the increasing subsequence
     * @param A the array of numbers
     * @return the max sum
     *
     */
    static public int sumIncreasingSubsequence(int[] A){
        if (A == null || A.length == 0) return 0;

        int N = A.length;
        int[] s = new int[N]; // max sum of an increasing subsequence with last element
        int[] p = new int[N]; // array of indexes of the elements preceding a[i];
        int i_maxSum = 0; // index of the max sum

        for (int i = 0; i < N; i++) {
            //You start with s[i] = a[i] and p[i] = ? where {?} is a number that represents the index
of the element before a[i] ; the number chosen doesnt matter .
            s[i] = A[i]; //the starting sum of A[i] which is whatever the first number from i is
            p[i] = -1; // element preceding index ; only useful to show summation {?} can be any
number less than 0 cus its an index
            for (int j = 0; j < i; j++) { //from left to whatever index i is ; j is an index to the
```

```
left of i
                if (A[i] >= A[j] && s[j] + A[i] >= s[i]){  // if A[j] is less than or equal to A[i]
and the sum of the subsequence ending at j plus A[i] is greater than the sum of the subsequence
ending at i
                    s[i] = s[j] + A[i] ; //sum is recalculated and left index of i is added to A[i]
                    p[i] = j;
                }
            }
            if (s[i] > s[i_maxSum]) // if summation at current i is greater than the last max sum
then i is the new max sum
                i_maxSum = i;
        }

        //part that exists only to show summation; doesnt effect maxsum
        {
            Stack<Integer> summation = new Stack<>();
            for (int i = i_maxSum; i != -1; i = p[i])
                summation.push(A[i]);

            System.err.print("\nSummation: \t");

            while (!summation.isEmpty())
                System.err.print(summation.pop() + " ");

            System.err.println();
        }

        return s[i_maxSum];
    }
}
```