PCS4 Assignments – Week 3

Version: 2016 April

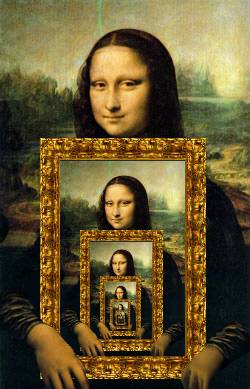


Table of Contents

[\* Assignment 3.1: Deliver the deliverables, again 2](#_Toc450765937)



[\* Assignment 3.2: Write several recursive methods 3](#_Toc450765938)

**NOTE**: The assignments marked with an asterisk are the most important ones.   
 The others serve as extra practice.



\* Assignment 3.1: Deliver the deliverables, again

Last week there was an assignment about sorting the deliverables of a transport company.

This assignment is also about sorting, but now with the help of the sorting-methods of the List-class.

So first, make a copy of the solution of last week.



**PART 1:**

The method SortByWeight() in the TransportCompany-class must sort the elements of the deliverables-list by weight (from a light weight to a heavy weight).

Replace your implementation for this method by

this.myDeliverables.Sort();

Since you use the Sort-method of the list-class without any parameters, it means that the Deliverable-class must implement the IComparable-interface.

Make it work.

**PART 2:**



Consider the method SortByAddress() in the TransportCompany-class.

Replace your implementation for this method by

this.myDeliverables.Sort( ???? );

Since the sorting-criteria is different than in part 1, we cannot do the same trick (the trick to use the Sort-method without parameters). It means that we should use one of the other two ways: using the Comparison or the IComparer.

Choose one of these techniques and implement it.



**PART 3:**

And now the method SortForPostman() in the TransportCompany-class (or another sorting if you feel uncomfortable with this one).

Replace your implementation for this method by

this.myDeliverables.Sort( ???? );

but now do it by using the other technique (so not the technique used in the former parts).

\* Assignment 3.2: Write several recursive methods

All questions below are about writing a recursive method. Implement it and make a simple application to test it.

1. Write a recursive method with the heading: public int sumSquares(int n) that   
    returns the sum 02 + 12 + 22 + 32 + 42 + . . . (n-1)2 + n2.  
   So, after performing  
    int a = sumSquares(5);  
   the value of a is 55 ( = 02 + 12 + 22 + 32 + 42 + 52 ).



1. Write a recursive method with the heading: public int sum(int[] X, int n) that   
    returns the sum of the first n elements of the array X, so it returns  
    X[0] + X[1] + X[2] + . . . + X[n-1].
2. Write a recursive method with the heading: public int max(int[] X, int n) that   
    returns the maximum value of the first n elements of the array X, so it returns the maximum value of X[0], X[1] , X[2] , . . . , X[n-1]. You may assume that n>=1.



1. Write a recursive method with the heading: public int sumOfDigits(int n) that   
    returns the sum of the digits of n. You may assume that n>=0.  
   So, after performing  
    int a = sumOfDigits(43195);  
   the value of a is 22 ( = 4 + 3 + 1 + 9 + 5 ).  
     
   Hint: Summing the digits of n could start with the most significant digit or with the least significant digit. The question is: how can you easily split a number n in its first or last digit and "the rest"? So In this example: you can see sumOfDigits(43195) as  
    4 + sumOfDigits(3195) but also as  
    sumOfDigits(4319) + 5.  
   Which is the easiest way to do?



1. Write a recursive method with the heading: public bool allBig(int[] X, int n) that   
    returns true if all of the first n numbers in the array X are bigger than 1000; otherwise it returns false.  
   In other words: it returns true, if ( Ai : 0<=i<n : X[i]>1000 ).



1. Write a recursive method with the heading: public bool isAscending(int[] X, int n) that   
    returns true if the first n numbers in the array X are in ascending order; otherwise it returns false.  
   In other words: it returns true, if ( Ai : 0<=i<n-1 : X[i] <= X[i+1] ).