Assignment 1: Basic unit testing

In this assignment you will practice with unit testing a small application that handles packages to be delivered at certain addresses. A start-up project is supplied with a partially implemented application and a partially implemented test class. Your job is to correct and extend both the application and the test class in order to pass all the given test methods successfully.

The application

A transport company delivers packages to its clients. The packages will be called deliverables. Each deliverable has an identity number, a weight and a person to whom it is sent. Each person has a name and an address, where packages for that person should be delivered.

The application has been named *DeliverablesApp* and its two main classes are Deliverable and Transport. Each object of the first holds all the information described above for a deliverable and each object of the second holds a collection of deliverables.

We provide a partially built Visual Studio solution that consists of two projects: *DeliverablesApp*, which contains the application itself, and *MyFirstUnitTestProject*, which contains the corresponding testing.

Finishing the set-up

The application itself, *DeliverablesApp*, can be compiled correctly but is not ready to be executed, since its user interface is still rather empty. However, there is no need to complete the user interface since our goal is just to test the Deliverable and Transport classes via the unit testing facilities of Visual Studio.

The testing project, *MyFirstUnitTestProject*, is not to be executed conventionally but through the framework that runs and analyses testing methods: the tests.

In the partially implemented solution we have provided the tests that cannot be executed; try it and you will see. The reason is that the testing project does not “see” the application project, even if they are part of the same solution.
Remedy this by adding a reference in the testing project to the application project as demoed by your teacher during the lecture.

Still, your testing project might not compile correctly. Besides adding a reference in the testing project to the application project, the test class must specify that it wants to use the namespace where the classes of the application project reside.
Add such a “using” directive and the testing project must then compile correctly.

Now you should finally be able to run the tests. However, you will find that only some of them succeed (becomes green coloured) while most of them fail (becomes red coloured).

The task

Your task is to modify the solution to pass all the unit tests. You will see that there are some missing parts in the application’s classes and in the test class, but there might also be mistakes in what is already built. The missing and the faulty parts must be corrected to make all the tests succeed (become green coloured).

A first simple fix

The test method *TestAddDeliverablesFromFile()* fails and it uses a method of the *Transport* class that reads deliverables from a given file. An example file with deliverables is given in the folder "data" contained in the root folder of the provided solution.
This is the file used in the test and more information about its contents is provided in Appendix A.

The testing framework lets you see an error message that indicates why the test fails. The test result can also direct you to the problematic line of code that causes the fail. In this case it is the line: *Assert.AreEqual(d.Name, "Anouk");*

A mistake; the buyer's name is not "Anouk" but "Adele" (check it in the provided deliverables file). Change the buyer's name in the test method, run the test again and check if it now succeeds.
Note that this mistake is for practicing purpose and to get you go through the UnitTest-class.
Normally this kind of mistake should not occur.

Other simple tests

For now, let us focus only in the first six tests. These are about simple creation of transports and addition of deliverables to them.

Some of these first six test methods, apart from the one fixed in the exercise above, also fail. You will find the following statement in the body of these test methods: *Assert.AreEqual(3 * 4, 25)*;

Of course, that is why these test methods fail. Remove this statement and add sensible testing code which, consequently, makes the first six test methods succeed.

Flexible testing of non-unique results

Let us now focus in the last five tests. These deal with finding the heaviest deliverables and with sorting transports.

Some of these last five test methods might fail for a different reason. In many of the tests about finding a heaviest deliverable, you will see that for some methods there might not be a unique correct answer (i.e. multiple deliverables with same 'heaviest' weight). It is important to learn to deal wisely with this fact.

Make sure you build your tests in such a way that the test succeeds with any of the correct answers and not only with one of them. That is, no matter what heaviest deliverable is returned by the application method, the test method must succeed.

Think also about this approach when deciding how to build the last test about sorting. Appendix B shows a possible result of sorting all the deliverables of Appendix A, but this is only one possible result. The test should be ready to succeed with any correct sorting.

Note that the importance of such flexible testing of non-unique results resides on the fact that, most of the times, the person or team building the application classes is not the same person or team building the test classes.

- End of assignment -

Appendix A: Contents of the file in the “data” folder

The folder “data” in the root folder of the provided solution contains a file with the description of several deliverables. For every deliverable, the file contains eight lines:

identity number, weight, receiver’s name, street, house number, postal code, city and a delimiter (which is a line with asterisks).

The file describes the following deliverables:

Id	**Weight**	**Receiver**
1	350	Sven Kramer
2	700	Adele
3	360	RedNex
4	360	Bruno Mars
5	500	Sven Kramer
6	950	Irene Wust
7	750	Naomi van As
8	950	Mr. Blue
9	100	Bruno Mars
10	40	Anouk
11	60	Anouk
12	30	Anouk
13	200	Anouk
14	250	Naomi van As
15	600	Irene Wust

Full information about the receivers:

Name	**Street, house number, postal code and city**	
Mr. Blue	Zaragossastraat 41, 5688DH, Veldhoven	
Sven Kramer	Mainstreet 33, 5688GE, Eindhoven	friend of Naomi
Irene Wust	Kerkstraat 20, 5693DE Eindhoven	
Naomi van As	Mainstreet 33, 5688GE, Eindhoven	friend of Sven
Anouk	Kerkstraat 10, 5693DE Eindhoven	
Adele	Kerkstraat 13, 5693DE Eindhoven	
Bruno Mars	Kerkstraat 39, 5693DE Eindhoven	
RedNex	AAstraat 166, 5688BX, Amsterdam	

Appendix B: A possible result of sorting

After sorting by address the deliverables of Appendix A, one possible result is shown in the rectangle below. Note that this is *one* possible correct result but *not the only one*. Can you think of other possible correct results?

```
Id: 3, weight: 360, should be delivered at: RedNex - AAstraat  166 - 5688BX  Amsterdam

Id: 12, weight: 30, should be delivered at: Anouk - Kerkstraat  10 - 5693DE  Eindhoven

Id: 10, weight: 40, should be delivered at: Anouk - Kerkstraat  10 - 5693DE  Eindhoven

Id: 13, weight: 200, should be delivered at: Anouk - Kerkstraat  10 - 5693DE  Eindhoven

Id: 11, weight: 60, should be delivered at: Anouk - Kerkstraat  10 - 5693DE  Eindhoven

Id: 2, weight: 700, should be delivered at: Adele - Kerkstraat  13 - 5688DE  Eindhoven

Id: 15, weight: 600, should be delivered at: Irene Wust - Kerkstraat  20 - 5693DE  Eindhoven

Id: 6, weight: 950, should be delivered at: Irene Wust - Kerkstraat  20 - 5693DE  Eindhoven

Id: 4, weight: 360, should be delivered at: Bruno Mars - Kerkstraat  39 - 5688DE  Eindhoven

Id: 9, weight: 100, should be delivered at: Bruno Mars - Kerkstraat  39 - 5688DE  Eindhoven

Id: 14, weight: 250, should be delivered at: Naomi van As - Mainstreet  33 - 5688GE  Eindhoven

Id: 1, weight: 350, should be delivered at: Sven Kramer - Mainstreet  33 - 5688GE  Eindhoven

Id: 5, weight: 500, should be delivered at: Sven Kramer - Mainstreet  33 - 5688GE  Eindhoven

Id: 7, weight: 750, should be delivered at: Naomi van As - Mainstreet  33 - 5688GE  Eindhoven

Id: 8, weight: 950, should be delivered at: Mr. Blue - Zaragossastraat  41 - 5688DH  Veldhoven
```