**Homework: Defining Classes - Part 2**

**Problem 1. Structure**

* Create a structure Point3D to hold a 3D-coordinate {X, Y, Z} in the Euclidian 3D space.
* Implement the ToString() to enable printing a 3D point.

**Problem 2. Static read-only field**

* Add a private static read-only field to hold the start of the coordinate system – the point O{0, 0, 0}.
* Add a static property to return the point O.

**Problem 3. Static class**

* Write a static class with a static method to calculate the distance between two points in the 3D space.

**Problem 4. Path**

* Create a class Path to hold a sequence of points in the 3D space.
* Create a static class PathStorage with static methods to save and load paths from a text file.
* Use a file format of your choice.

**Problem 5. Generic class**

* Write a generic class GenericList<T> that keeps a list of elements of some parametric type T.
* Keep the elements of the list in an array with fixed capacity which is given as parameter in the class constructor.
* Implement methods for adding element, accessing element by index, removing element by index, inserting element at given position, clearing the list, finding element by its value and ToString().
* Check all input parameters to avoid accessing elements at invalid positions.

**Problem 6. Auto-grow**

* Implement auto-grow functionality: when the internal array is full, create a new array of double size and move all elements to it.

**Problem 7. Min and Max**

* Create generic methods Min<T>() and Max<T>() for finding the minimal and maximal element in the GenericList<T>.
* You may need to add a generic constraints for the type T.

**Problem 8. Matrix**

* Define a class Matrix<T> to hold a matrix of numbers (e.g. integers, floats, decimals).

**Problem 9. Matrix indexer**

* Implement an indexer this[row, col] to access the inner matrix cells.

**Problem 10. Matrix operations**

* Implement the operators + and - (addition and subtraction of matrices of the same size) and \* for matrix multiplication.
* Throw an exception when the operation cannot be performed.
* Implement the true operator (check for non-zero elements).

**Problem 11. Version attribute**

* Create a [Version] attribute that can be applied to structures, classes, interfaces, enumerations and methods and holds a version in the format major.minor (e.g. 2.11).
* Apply the version attribute to a sample class and display its version at runtime.