# **Homework: Static Members and Namespaces**

This document defines the homework assignments from the "OOP" Course @ Software University. Please submit as homework a single **zip / rar / 7z** archive holding the solutions (source code) of all below described problems. The solutions should be written in C#.

### Problem 1. Point3D

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

Add a private **static read-only field** in the **Point3D** class to hold the start of the coordinate system – the point **StartingPoint** {0, 0, 0}. Add a **static property** to return the starting point.

### **Problem 2. Distance Calculator**

Write a static class **DistanceCalculator** with a static method to **calculate the distance** between two points in the 3D space. Search in Internet how to calculate distance in the 3D Euclidian space.

### Problem 3. \* Paths

Create a class **Path3D** to hold a sequence of points in the 3D space. Create a static class **Storage** with static methods to save and load paths from a text file. Use a file format of your choice. Learn how to read and write text files in Internet. Ensure you close correctly all files with the "**using**" statement.

## Problem 4. \* HTML Dispatcher

Write a class **ElementBuilder** that creates HTML elements:

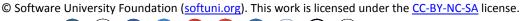
- The class constructor should take the **element's name** as argument.
- Write a method AddAtribute(attribute, value) that adds an attribute and value to the element. For
  example, we create an element a and add the attributes href="www.softuni.bg" and class="links". The
  result is <a href="www.softuni.bg" class="links"><a/>.
- Write a method **AddContent(string)** that inserts content inside the current tag (e.g. **<div>***Text***</div>**).
- Overload the \* operator for **ElementBuilder** objects. The operator should multiply the string value of the element **n** times and return the result as string. (e.g. \* 3 = \* | 1| >

Sample Source Code	Output
ElementBuilder div =	<div <="" id="page" td=""></div>
<pre>new ElementBuilder("div");</pre>	class="big">Hello> <div <="" id="page" td=""></div>
<pre>div.AddAttribute("id", "page");</pre>	class="big">Hello>
<pre>div.AddAttribute("class", "big");</pre>	
<pre>div.AddContent("Hello");</pre>	
Console.WriteLine(div * 2);	

Write a static class **HTMLDispatcher** that holds 3 **static** methods: **CreateImage()**, **CreateURL()**, **CreateInput()**, which takes a set of arguments and return the HTML element as string. Use the **ElementBuilder** class.

- CreateImage() takes image source, alt and title.
- CreateURL() tekes url, title and text.
- CreateInput() takes input type, name and value.























Test the HTML Dispatcher by creating various HTML elements, using the implemented static methods.

## Problem 5. \*\* BitArray

Write a class BitArray that holds a bit sequence of integer numbers. It should support bit arrays of size between 1 and 100 000 bits. The number of bits is assigned when initializing the object. The class should support bit indexation (accessing and changing any bit at any position – e.g. num[2] = 0, num[867] = 1, etc.)

Override ToString() to print the number in decimal format. For example, we can create a BitArray object num with 8 bits (bits are 0 by default). We change the bit at position 7 to have a value of 1 (num[7] = 1) and print it on the console. The result is 128.

Tips: Write your own algorithm for binary-to-decimal conversion. Encapsulate all fields. Throw proper exceptions in case of improper input data or indexes, with descriptive messages.

## **Problem 6. Namespaces**

Design a group of classes to work with geometric figures. Put them into namespaces. You do not to implement the classes, just create them and put them into namespaces.

Namespace **Geometry.Geometry2D** holds classes:

- Point2D
- Figure2D
- Square
- Rectangle
- Polygon
- Circle
- Ellipse
- DistanceCalculator2D

#### Namespace Geometry.Geometry3D holds classes:

- Point3D
- Path3D
- DistanceCalculator3D

#### Namespace Geometry. Storage holds classes:

- GeometryXMLStorage
- GeometryBinaryStorage
- GeometrySVGStorage

#### Namespace Geometry.UI holds classes:

- Screen2D
- Screen3D





















