You can perform round-trip engineering in Visual Studio, to keep C# source code and class model in sync. In this tutorial, we will see how to create a class model in Visual Studio, and eventually generating source code from model.

**Preparation**

In order to follow and complete this tutorial, you must have Visual Paradigm installed, which can be downloaded from Visual Paradigm [download page](https://www.visual-paradigm.com/download.jsp). Of course, you need Visual Studio ready, and with Visual Studio integration installed in advance. Finally, to make the tutorial easier to follow we are not going to describe every little step required to draw a class diagram in detail. We are assuming that you have the basic skills required to draw UML class diagram in Visual Paradigm.

**Design system with UML Class Diagram**

1. Create a C# library project Map Viewer in Visual Studio.
2. Right click on the project node in **Solution Explorer**, and select **Open Visual Paradigm** from the popup menu.

3. In **Diagram Navigator**, right click on class diagram and select **New Class Diagram** from the popup menu.
4. A new diagram is created. You are asked to enter a package header on top of the diagram. Enter myapp and press Enter. From now on classes to be drawn in this diagram will be placed in a (new) package named myapp. In code level, those classes will be in myapp namespace.

5. Click on the down arrow button near the shape selection Class in diagram toolbar, and select Interface.

6. Click on the diagram to create an interface class and name it as IMap.

7. Create operations in IMap. Right click on the class IMap and select Add > Operation from the popup menu.

8. Enter JumpToLocation(x:int, y:int): void to create a public operation JumpToLocation with parameter x, y and return void.

9. Press Enter to create another operation, Name it Clear() : void. Click on diagram to confirm editing.
10. We need to create a class for town map which inherits IMap. Move the mouse pointer over interface IMap, press on the Resource Catalog icon and drag it out.

11. Release your mouse button on the empty space. Select Realization -> Class from Resource Catalog to create a new class.

12. Name the class as TownMap and press Enter to confirm.

13. As the TownMap class is implementing the interface IMap, we need to implement the operations defined in IMap. Right click on class TownMap and select Related Elements > Realize all Interfaces from the popup menu. You can see that operations JumpToLocation and Clear are both inherited.
14. It is time to add properties to class. Right click on class *TownMap* and select **Add > Property** from the popup menu.

15. Enter *name : string* to name the property as *name*, and set the type as *string*. Press **Enter** to proceed to the next property. Enter *x : double* as property name and type. Then, press **Enter** and create property *y : double*.

16. We need to create a new class *Region* with an aggregation (association) from class *TownMap*. Again, use Resource Catalog to create a class from *TownMap*. This time, use the resource **Aggregation -> Class**.

17. Follow the previous steps to create properties in class *Region*.

<table>
<thead>
<tr>
<th>Class</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region</td>
<td>name : string</td>
</tr>
<tr>
<td></td>
<td>desc : string</td>
</tr>
<tr>
<td></td>
<td>x : double</td>
</tr>
<tr>
<td></td>
<td>y : double</td>
</tr>
</tbody>
</table>
18. Up to now, the diagram should look like:

```
<<Interface>>
INap
+JumpToLocation(x : int, y : int) ; void
+Client() ; void
```

**UML to C# code generation in Visual Studio**

1. Save your work via the File menu.
2. Now comes the code generation. Select the menu **Update Code** on **Diagram Navigator**.
3. Check the **Solution Explorer**. You should see a list of generated file. You can open them to fill in the code body.

4. This is the end of the tutorial. Instead of closing Visual Studio now, you may try something more by editing the code like to add, rename or delete class, properties and operations, and select **Update UML Model** from **Diagram Navigator**, and observe the changes that will make in the class model. Enjoy!

**Related Links**
- [Full set of UML tools and UML diagrams](https://www.visual-paradigm.com/tutorials/visual-studio-uml-to-csharp-tutorial.jsp)

**Visual Paradigm home page**
(https://www.visual-paradigm.com/)

**Visual Paradigm tutorials**
(https://www.visual-paradigm.com/tutorials/)