Generating C# from Class Diagram in Visual Studio?

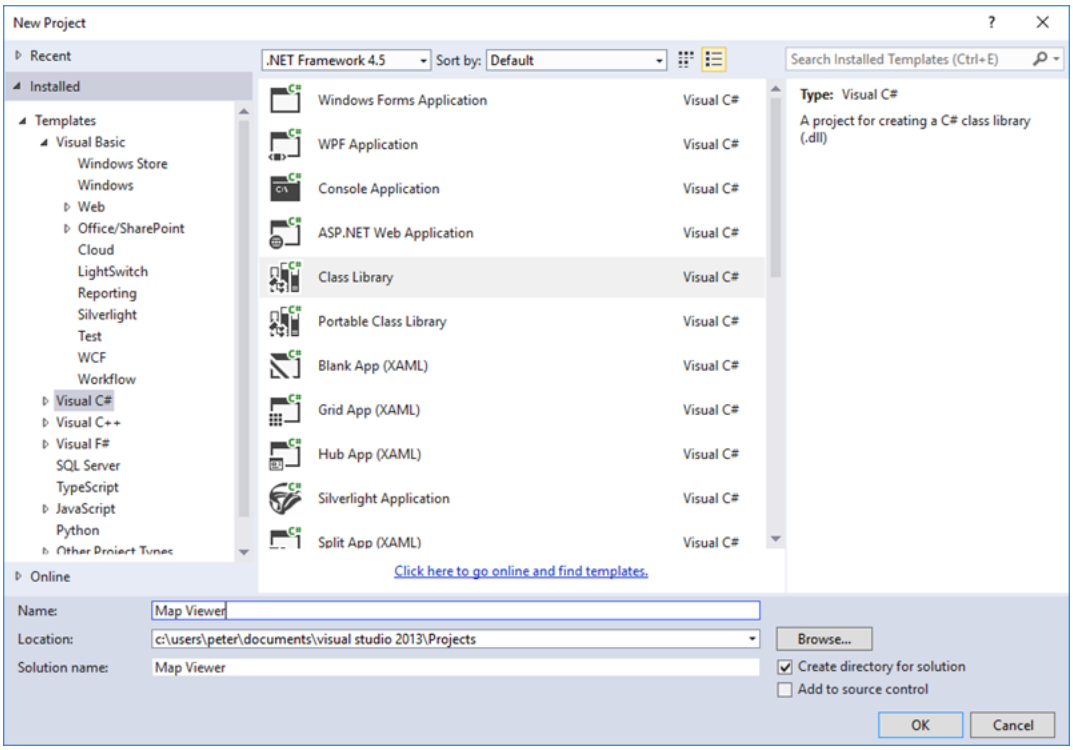
Written Date : March 6, 2016

Preparation

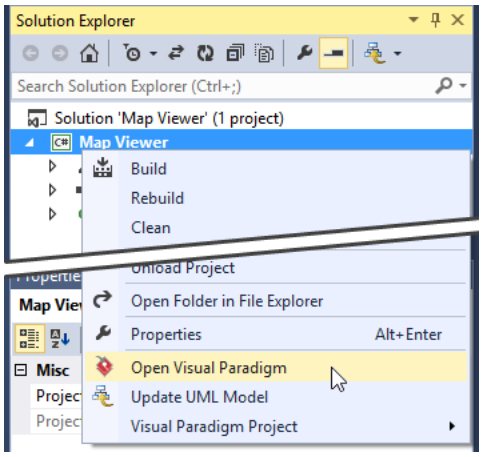
In order to follow and complete this tutorial, you must have Visual Paradigm installed, which can be downloaded from the Visual Paradigm download page. Of course, you also need Visual Studio, with the Visual Paradigm integration installed in advance. Finally, to make the tutorial easier to follow, we will not describe every minor step required to draw a class diagram in detail. We are assuming that you have the basic skills required to draw a UML class diagram in Visual Paradigm.

Designing a System with a UML Class Diagram

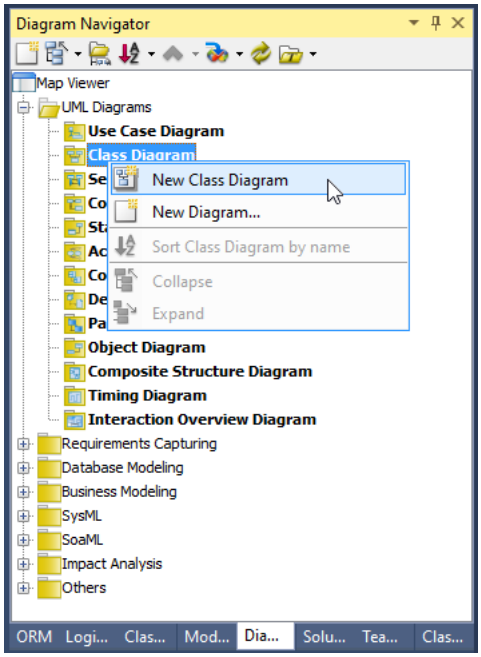
1. Create a C# library project named *Map Viewer* in Visual Studio.



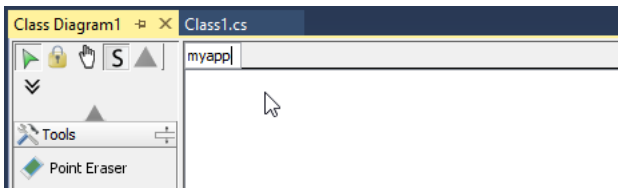
2. Right-click on the project node in **Solution Explorer** and select **Open Visual Paradigm** from the popup menu.



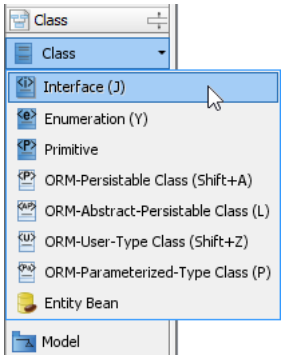
3. In **Diagram Navigator**, right-click on **Class Diagram** and select **New Class Diagram** from the popup menu.



4. A new diagram is created. You are asked to enter a package header at the top of the diagram. Enter *myapp* and press **Enter**. From now on, classes drawn in this diagram will be placed in a new package named *myapp*. In terms of code, those classes will be in the `myapp` namespace.



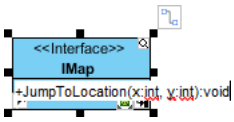
5. Click the down arrow button next to the **Class** shape selection in the diagram toolbar, and select **Interface**.



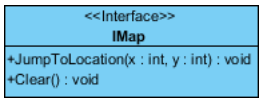
6. Click on the diagram to create an interface and name it `IMap`.

7. Create operations in `IMap`. Right-click on the `IMap` interface and select **Add** > **Operation** from the popup menu.

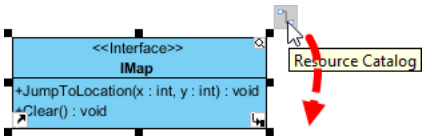
8. Enter *JumpToLocation(x:int, y:int): void* to create a public operation `JumpToLocation` with parameters x and y, and a void return type.



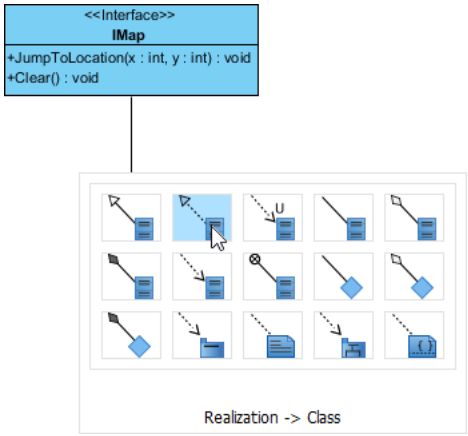
9. Press **Enter** to create another operation. Name it `Clear() : void`. Click on the diagram to confirm.



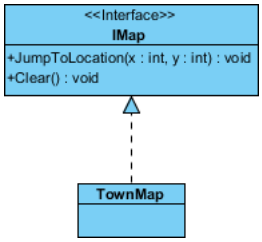
10. We need to create a class for a town map that implements `IMap`. Move the mouse pointer over the `IMap` interface, press on the **Resource Catalog** icon, and drag it out.



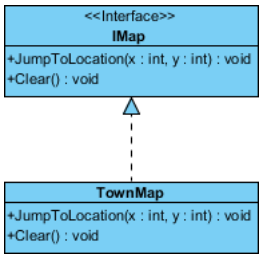
11. Release your mouse button on an empty space. Select **Realization -> Class** from the Resource Catalog to create a new class.



12. Name the class `TownMap` and press **Enter** to confirm.

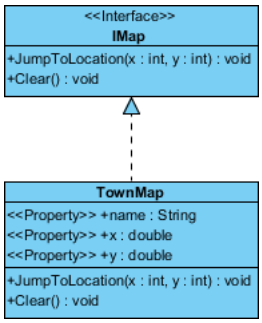


13. As the `TownMap` class is implementing the `IMap` interface, we need to implement the operations defined in `IMap`. Right-click on the `TownMap` class and select **Related Elements** > **Realize all Interfaces** from the popup menu. You can see that the operations `JumpToLocation` and `Clear` are both implemented.

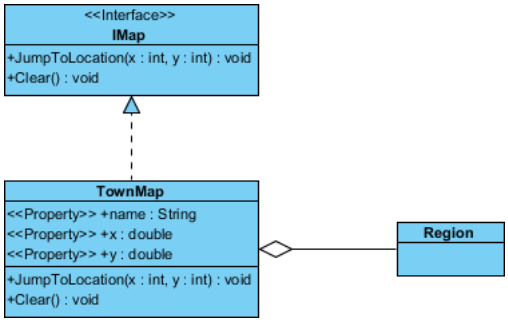


14. It is time to add properties to the class. Right-click on the `TownMap` class and select **Add** > **Property** from the popup menu.

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



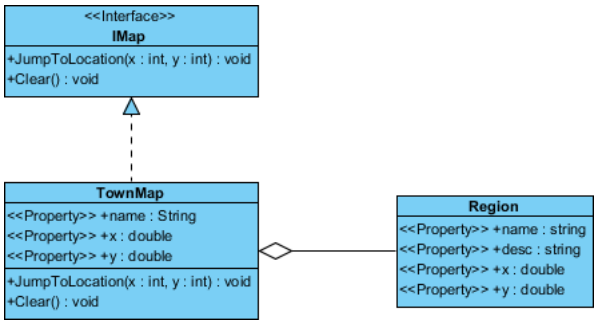
16. We need to create a new class, *Region*, with an aggregation association from the `TownMap` class. Again, use the Resource Catalog to create a class from `TownMap`. This time, use the **Aggregation -> Class** resource.



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

Class	Properties
Region	name : string
	desc : string
	x : double
	y : double

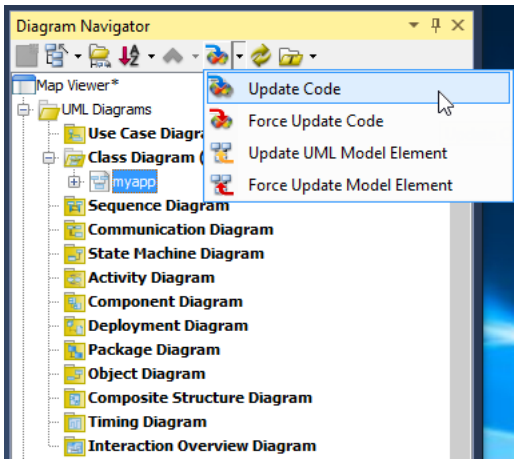
18. At this point, the diagram should look like this:



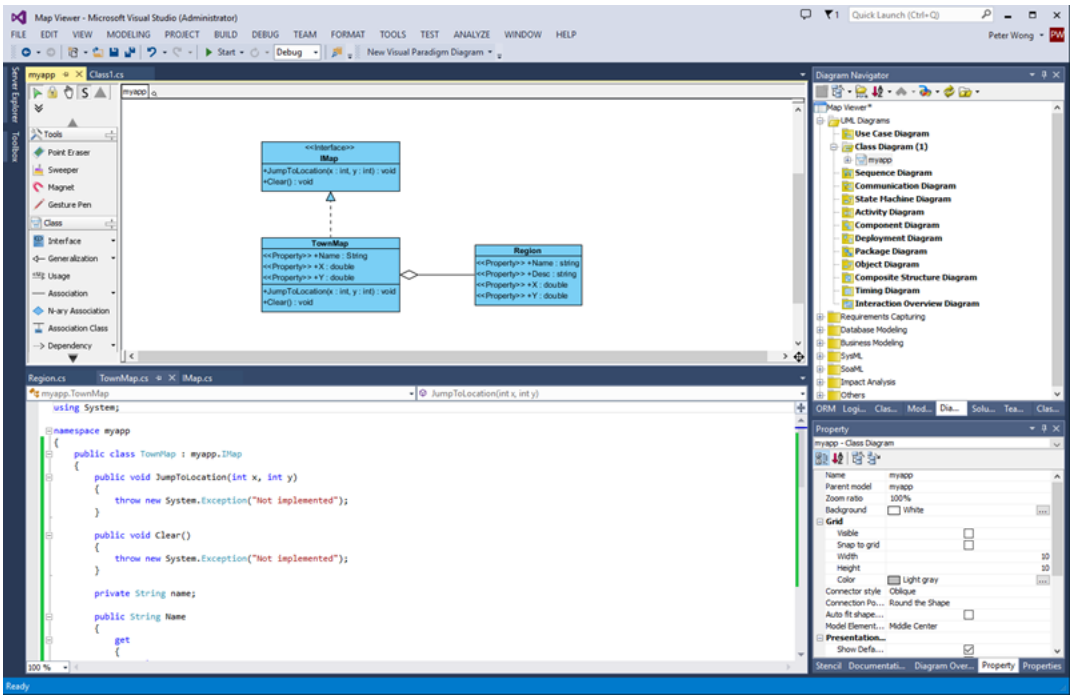
UML to C# Code Generation in Visual Studio

1. Save your work via the **File** menu.

2. Now it's time for code generation. Click the **Update Code** button in the **Diagram Navigator**.



3. Check the **Solution Explorer**. You should see a list of generated files. You can open them to fill in the method bodies.



4. This is the end of the tutorial. Instead of closing Visual Studio now, you may try something more by editing the code, like adding, renaming, or deleting classes, properties, and operations. Then select **Update UML Model** from the **Diagram Navigator** and observe the changes that are made to the class model. Enjoy!

Related Links

- User's Guide - Overview and installation of Visual Studio integration
- Full set of UML tools and UML diagrams

Visual Paradigm

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

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