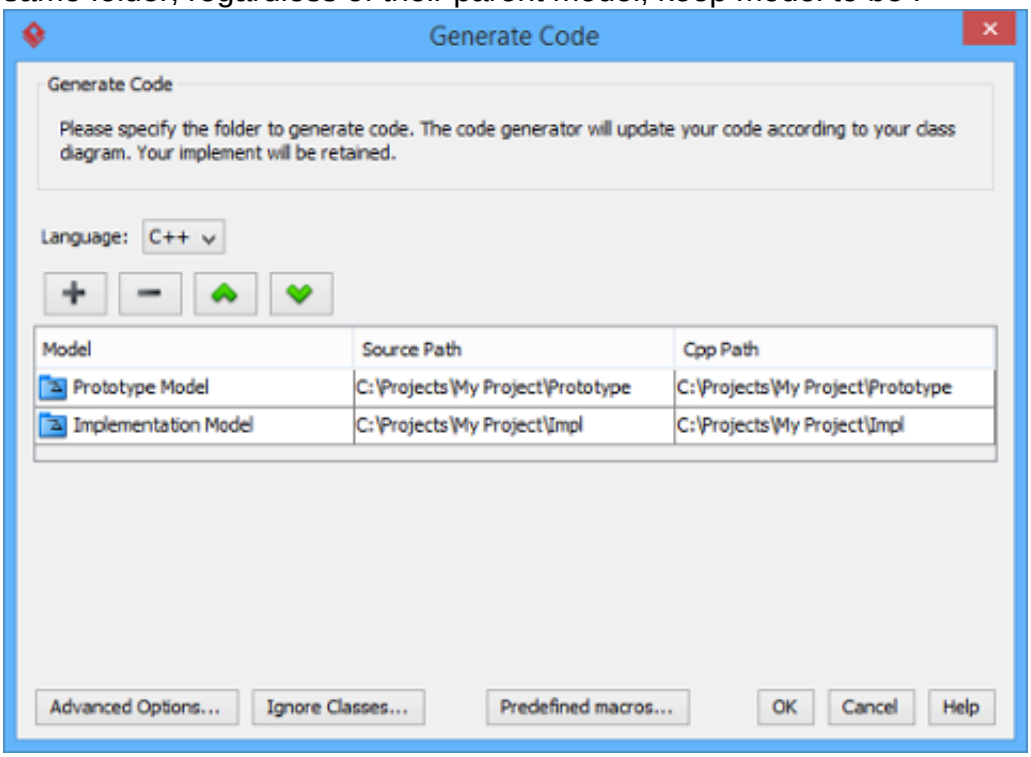
How to generate C++ from UML in Round-Trip

Round-trip engineering is the ability to generate model from source code and generate source code from UML model and keep them synchronized. You can make use of round-trip engineering to keep your implementation model and source code up-to-date, so as to produce up-to-date description on your model.

Generating/Updating code from whole project

You can generate C++ code from all classes in current project. To generate code from project:

1. Select **Tools > Code > Generate Java Code...** from the toolbar. .
2. Select **C++** as the **Language**.
3. In the **Generate Code** dialog box, specify the mapping between model and source path. Model is a UML element that acts as a container of other elements. Classes and packages under a model will be generated to the mapped source path. You can add multiple model-to-source-path mapping by pressing the **+** button. If you are not using model to structure your project or if you want to generate all classes in project to the same folder, regardless of their parent model, keep model to be .



4. Optionally configure the advanced code generation options by clicking **Advanced Options...**. Read the section Advanced Options in this chapter for details about the options.
5. Click **OK** to proceed with generation.

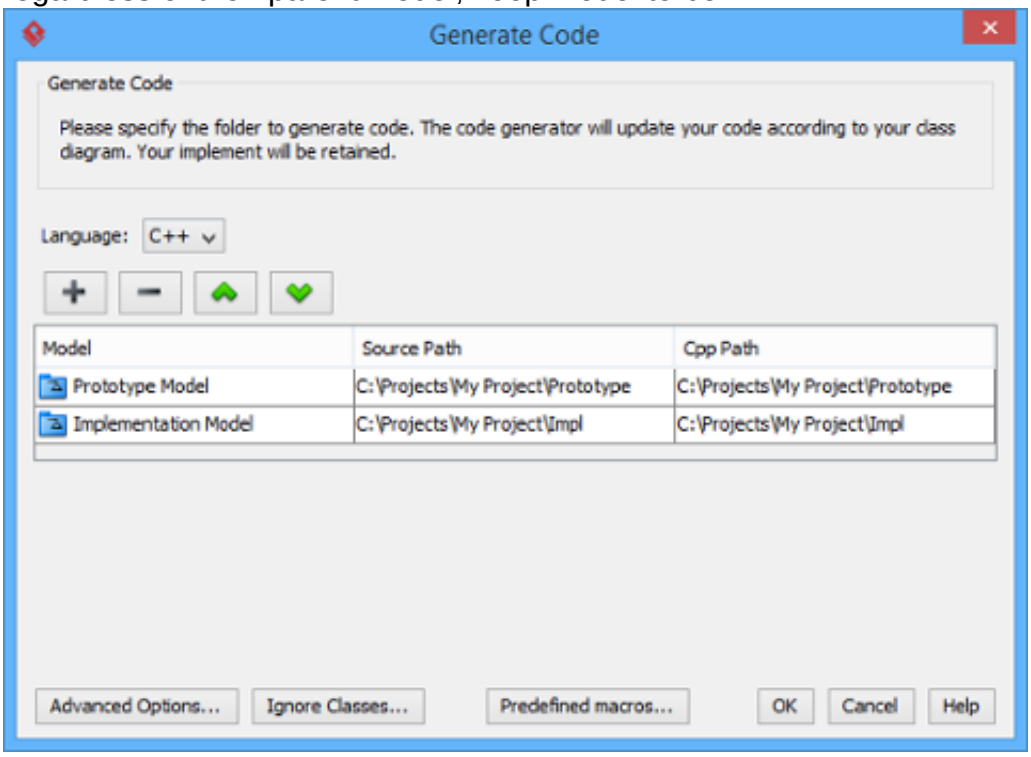
Note: Description in model elements is generated as comment in code.

Generating/Updating code from opening class diagram

You can generate C++ code from an opening class diagram that contains the class(es) you

want to generate code. To generate code from class diagram:

1. Right click on the class diagram background and select **Utilities > C++ Round-trip > Generate Code** from the popup menu.
2. In the **Generate Code** dialog box, specify the source path where you want the code to be generated. Model is a UML element that acts as a container of other elements. Notice that source path is set for model, not for diagram. Classes and packages under a model will be generated to the mapped source path. You can add multiple model-to-source-path mapping by pressing the **+** button. If you are not using model to structure your project, or if you want to generate all classes in project to the same folder, regardless of their parent model, keep model to be .



   Note: If you have generated code for once, the **Generate Code** window will not appear next time when you generate/update code, for any diagram. If you want to configure the model-to-source-path mapping or to configure options, you can run a code generation for project (refer to the previous section for detail).
3. Optionally configure the advanced code generation options by clicking **Advanced Options...**. Read the section Advanced Options in this chapter for details about the options.
4. Click **OK** to proceed with generation.

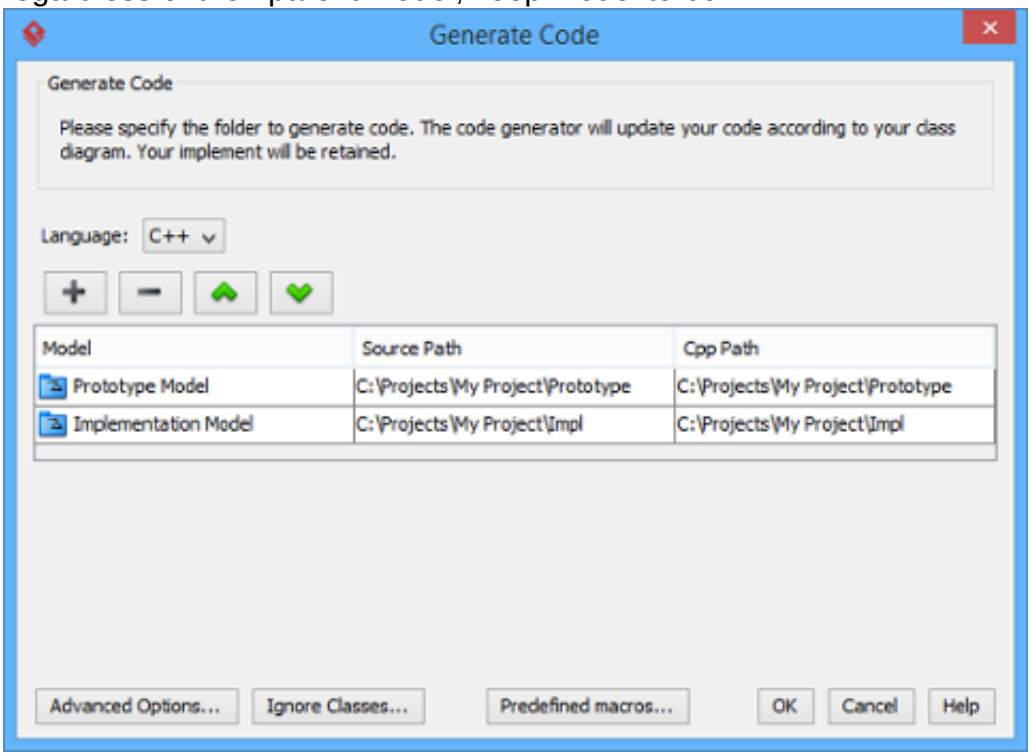
Note: Description in model elements is generated as comment in code.

Generating/Updating code from chosen classes

You can generate C++ code from specific class or classes. To generate code from class/classes:

1. Select the class(es) and right click on them, then select **C++ Round-trip > Generate Code** from the popup menu.
2. In the **Generate Code** dialog box, specify the source path where you want the code to be generated. Model is a UML element that acts as a container of other elements. Notice that source path is set for model, not for diagram. Classes and packages under a

model will be generated to the mapped source path. You can add multiple model-to-source-path mapping by pressing the **+** button. If you are not using model to structure your project, or if you want to generate all classes in project to the same folder, regardless of their parent model, keep model to be .

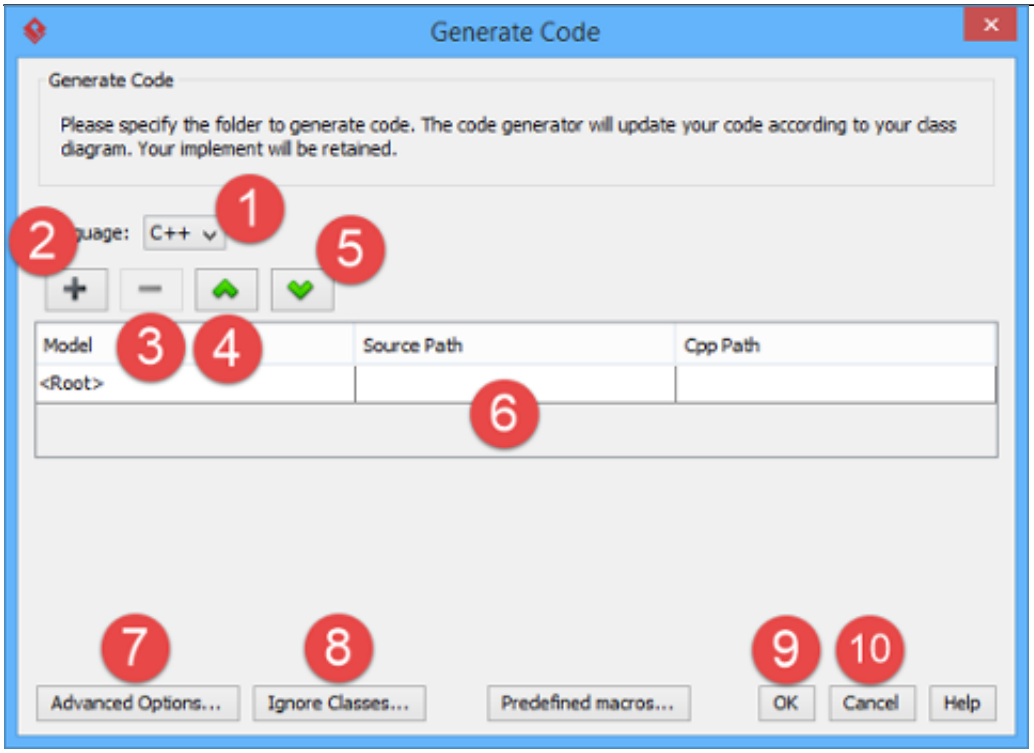


Note: If you have generated code for once, the **Generate Code** window will not appear next time when you generate/update code, for any class selection. If you want to configure the model-to-source-path mapping or to configure options, you can run a code generation for project (refer to the previous section for detail).

3. Optionally configure the advanced code generation options by clicking **Advanced Options...**. Read the section Advanced Options in this chapter for details about the options.
4. Click **OK** to proceed with generation.

Note: Description in model elements is generated as comment in code.

An overview of Generate Code dialog box

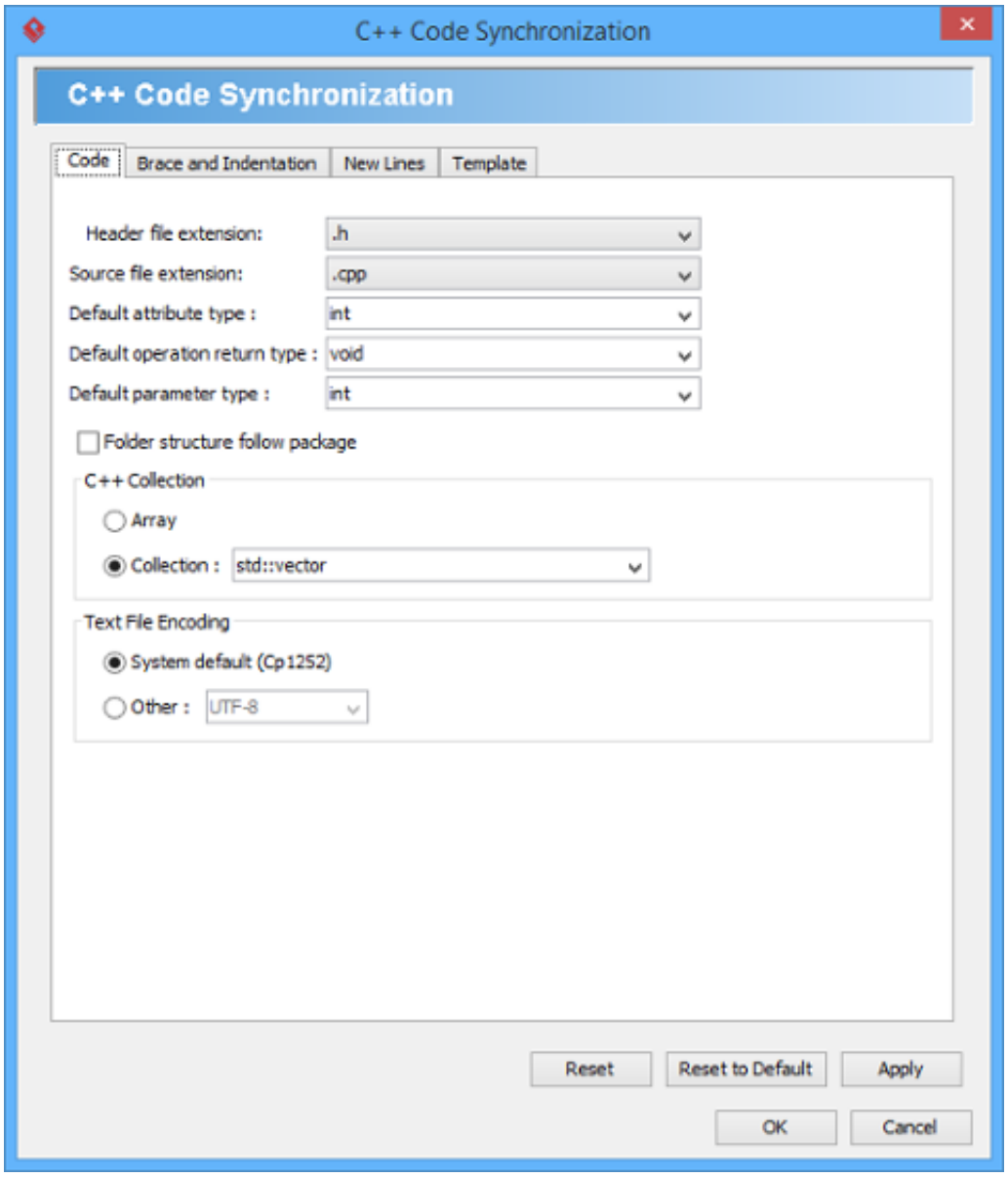


No.	Name	Description
1	Language	The programming language of the source code to generate.
2	Add model-to-source-path mapping	Click to add a new mapping between UML model and the source path where code will be generated to.
3	Remove model-to-source-path mapping	Click to remove chosen model-to-source-path mapping.
4	Move model-to-source-path mapping up	Click to move chosen model-to-source-path mapping one item upward.
5	Move model-to-source-path mapping down	Click to move chosen model-to-source-path mapping one item downward.
6	Model-to-source-path mapping	A list of mapping between UML model and source path.
7	Advanced options	Click to configure advanced code generation options. For details, read the section *Advanced Options* in this chapter.
8	Ignore classes	Click to organize the ignore list of classes to ignore in code generation. For details, read the section *To ignore classes in generation* in this chapter.
9	OK	Click to start generation.
10	Cancel	Click to close the **Generate Code** dialog without generating code.

Advanced options

You can configure the advanced options for more control of the code by clicking the **Advanced Options...** button in **Generate Code** dialog box. In the **Code Synchronization** dialog box popped up, there are four categories (tabs) of settings you can configure. Below is a description.

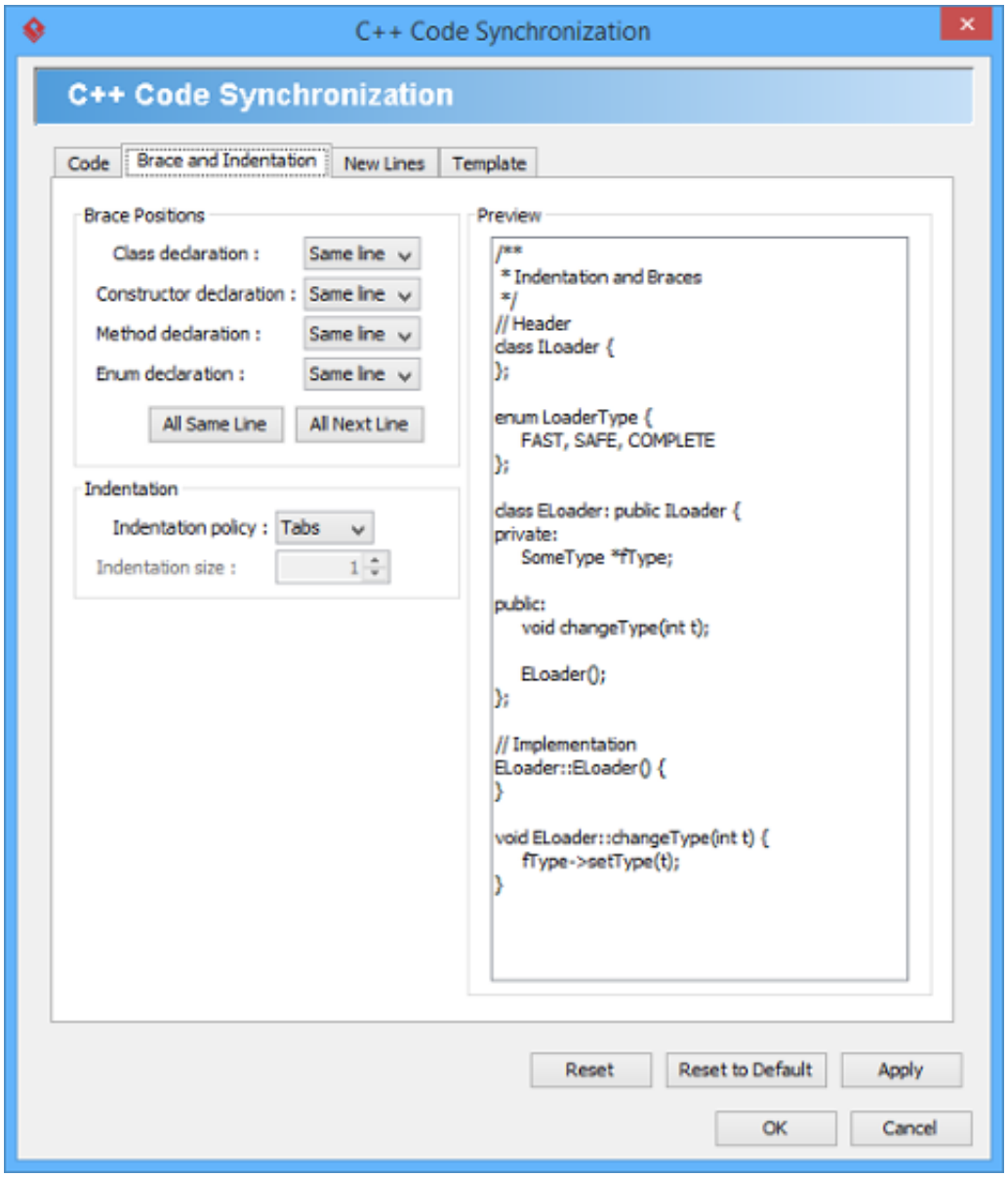
Code



Option	Description
Default attribute type	(default int) Type that will be assigned to Attribute upon code generation when type is unspecified
Default operation return type	(default void) Return Type that will be assigned to operation upon code generation when return type is unspecified
Default parameter type	(default int) Type that will be assigned to Parameter upon code generation when type is unspecified

Option	Description
Text File Encoding	• System default - (default) The default system encoding will be selected as encoding for source files • Other -Specify an encoding for source files

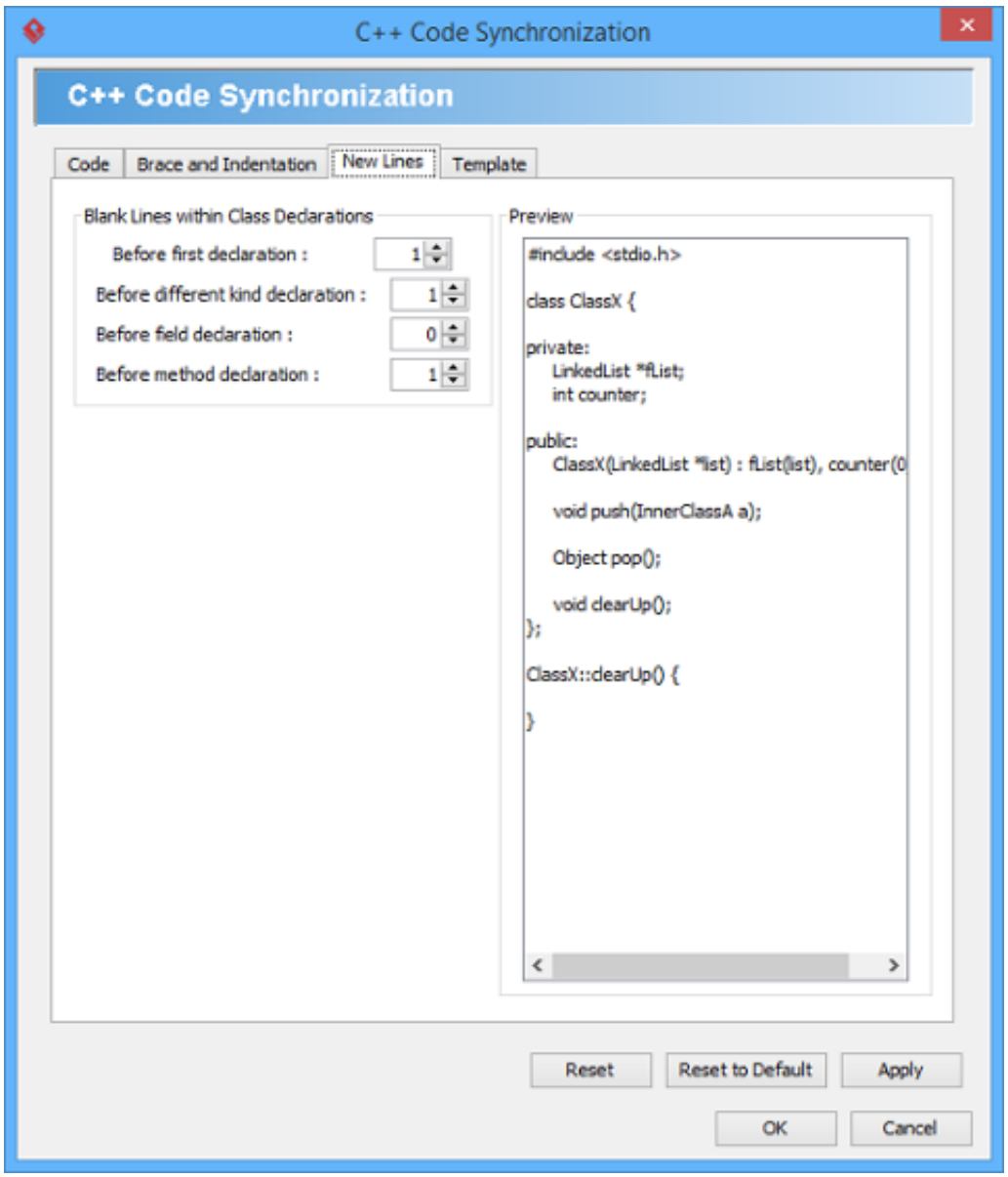
Brace and Indentation



Option	Description
Class declaration	• Same line - (default) Brace for class declaration appear at the same line as the declaration • Next line - Brace for class declaration appear at the line after the declaration
Constructor declaration	• Same line - (default) Brace for constructor appear at the same line as the declaration • Next line - Brace for constructor appear at the line after the declaration

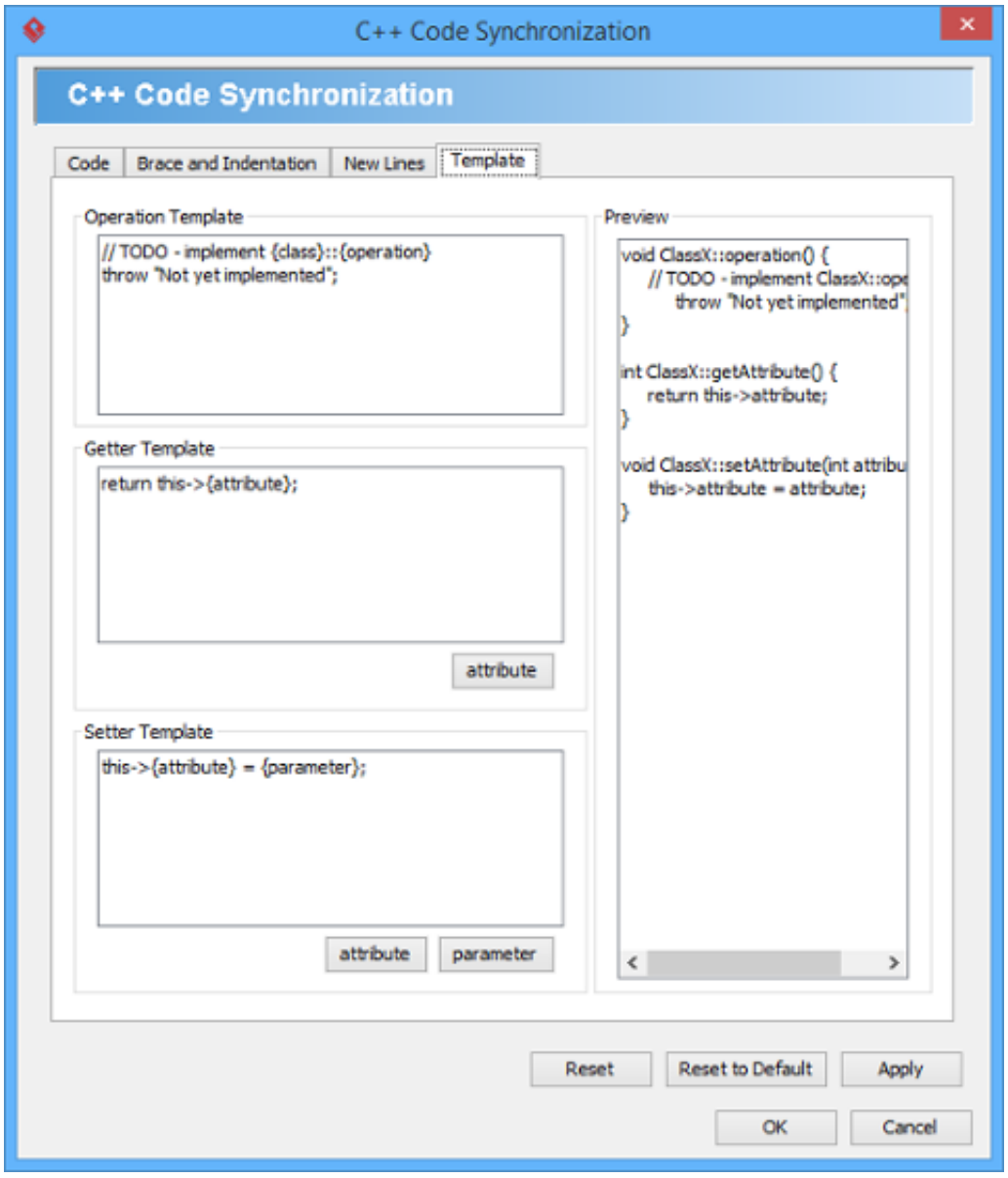
Option	Description
Method declaration	• Same line - (default) Brace for method appear at the same line as the declaration • Next line - Brace for method appear at the line after the declaration
Enum declaration	• Same line - (default) Brace for enumeration appear at the same line as the declaration • Next line - Brace for enumeration tor appear at the line after the declaration
Indentation policy	• Tabs - (default) Use a tab of space as indentation • Spaces - Use spaces as indentation. The number of spaces can be defined below
Indentation size	The number of spaces to indent

New Lines



Option	Description
Before first declaration	Number of blank lines to appear before the first declaration within Class declarations
Before different kind declaration	Number of blank lines to appear before a different kind of declaration
Before field declaration	Number of blank lines to appear before field declaration
Before method declaration	Number of blank lines to appear before method declaration

Template



Option	Description
Operation Template	Defines a template of method body that will be applied when generating operations.
Getter Template	Defines a template of getter that will be applied when generating getter methods. Getter will be generated to attribute stereotyped as Property, or with property getter selected.

Option	Description
Setter Template	Defines a template of setter that will be applied when generating setter methods. Setter will be generated to attribute stereotyped as Property, or with property setter selected.

To ignore classes in generation

You can make certain UML class not to generate code against code generation by ignoring them. To ignore class(es), click **Ignore Classes...** in **Generate Code** dialog box. In the second Generate Code dialog box that popped up, select the class(es) to ignore and click **>** to move them to the ignore list. Click **OK** to confirm.