Reverse Options

Demo

- Code Reverse

On this page

A reverse is the opposite of code generation. Existing code can be converted to UML models with the help of the MagicDraw reverse feature. In order to perform a reverse operation, prepare the sets the same way you did for code generation, see Code Engineering Sets.

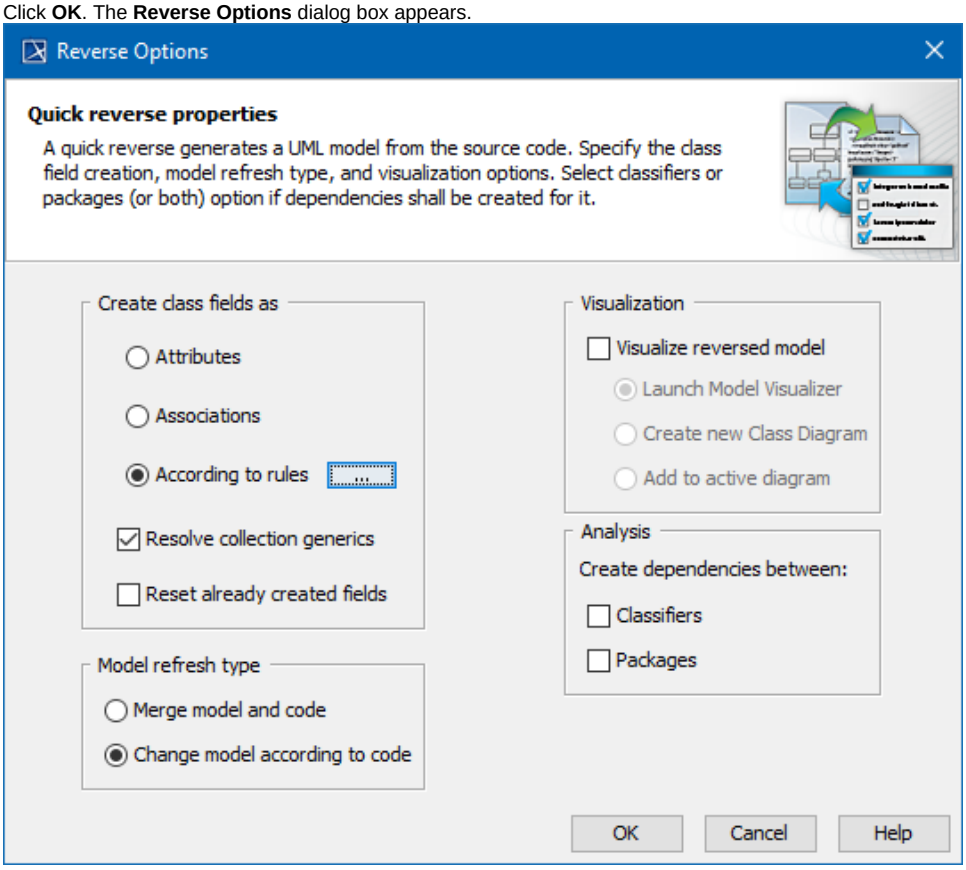
- Choose **Reverse** from the **Code engineering sets** item shortcut menu.
- Choose **Reverse** from the selected set shortcut menu.

The UML model for the component can be reversed in the same way. Simply select the component you are interested in from the browser and click Reverse on its shortcut menu. Models can be reversed without creating a set.

To reverse a model without creating a set

---

1. Choose **Quick Reverse** from the **Tools** menu. The **Round Trip Set** dialog box appears. Quick Reverse is available only in Professional and Enterprise editions.
2. Select the files from the **Round Trip Set** dialog box, **Add Files** tab.
3. Click **OK**. The **Reverse Options** dialog box appears.



The following table outlines Dialog descriptions:

Area	Element name	Function
Create class fields as	**Attributes**	Class fields are represented in a model as attributes.
	Associations	Class fields are represented in a model as association ends.
	According to rules	Association or Attribute creation on reverse is the ability to enter rules that help decide if an association or attribute must be created on reverse.

	Resolve collection generics	Reverse engineering can create associations when one class has a collection of other classes and uses Java generics (for example, *List<String>*). If selected, types of collections will be resolved (property type will not be a collection, but a real type). Predefined container types in Java language properties will be appended by all the same containers in this form: *·java .util.List<$$type$$>* where *$$type$$* replaced by the value of "type" property when the code is generated.
	Reset already created fields	Select this option if you want to keep previously created UML representation (attribute or association) for class fields.
Model refresh type	**Merge Model and Code**	The model elements are updated by code. Elements that do not exist in the code will not be removed from the model.
	Change Model According to Code	The model will be created strictly by code. Entities in the model that do not match entities in the code will be discarded.
Vizualization	**Visualize reserved model**	Classes created while reversing can be added to a diagram.
	Launch Model Visualizer	After reversing, the **Model Visualizer** dialog box appears. It will assist you in creating a Class diagram or Sequence diagram (Java only) for newly created entities.
	Create new Class Diagram	After reversing, the **Create Diagram** dialog box appears. Create a new diagram where the created entities will be added.
	Add to active diagram	After reversing, all created entities will be added to the current opened diagram.
Analysis. **Create dependencies between**	**Classifiers**	Dependencies between classes will be analyzed and created.
	Packages	Dependencies only between packages will be created.

Method bodies are not parsed on dependency search. Only static information is used.

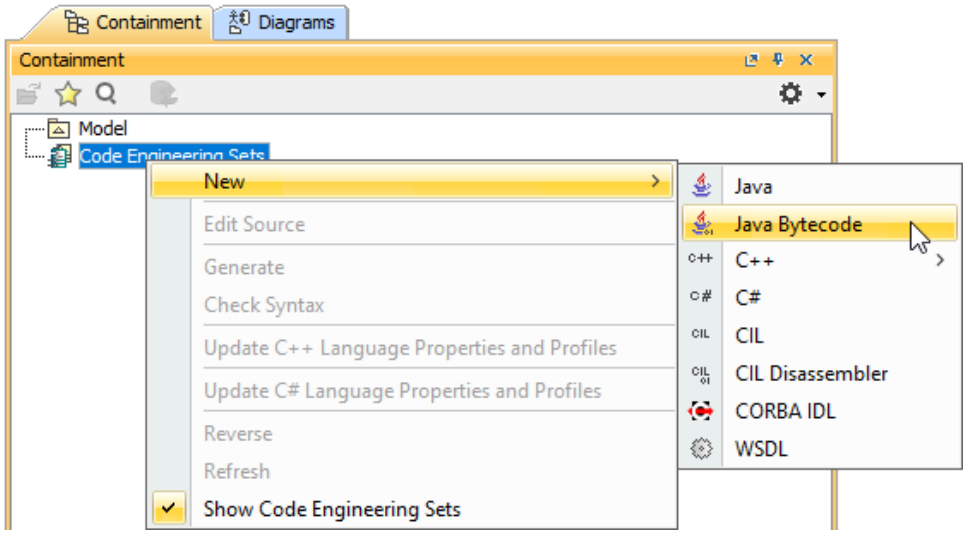
If you have a code set combined from several files, you may see changes you wish to model without reversing all the code. Only changed files should be reversed. You can do this type of reversing by clicking the **Refresh** button on the set shortcut menu, or by performing model refresh from the Code Engineering Sets dialog box.

Importing JAVA .JAR packages

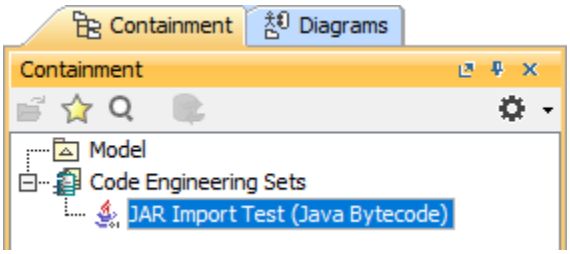
Apart from reversing code operation, **Code Engineering Sets** also allows you to import JAVA .JAR files and their Packages using **Java Bytecode** on the **Code Engineering Sets** shortcut menu.

To include (or import) a JAVA .JAR package with Java code generation

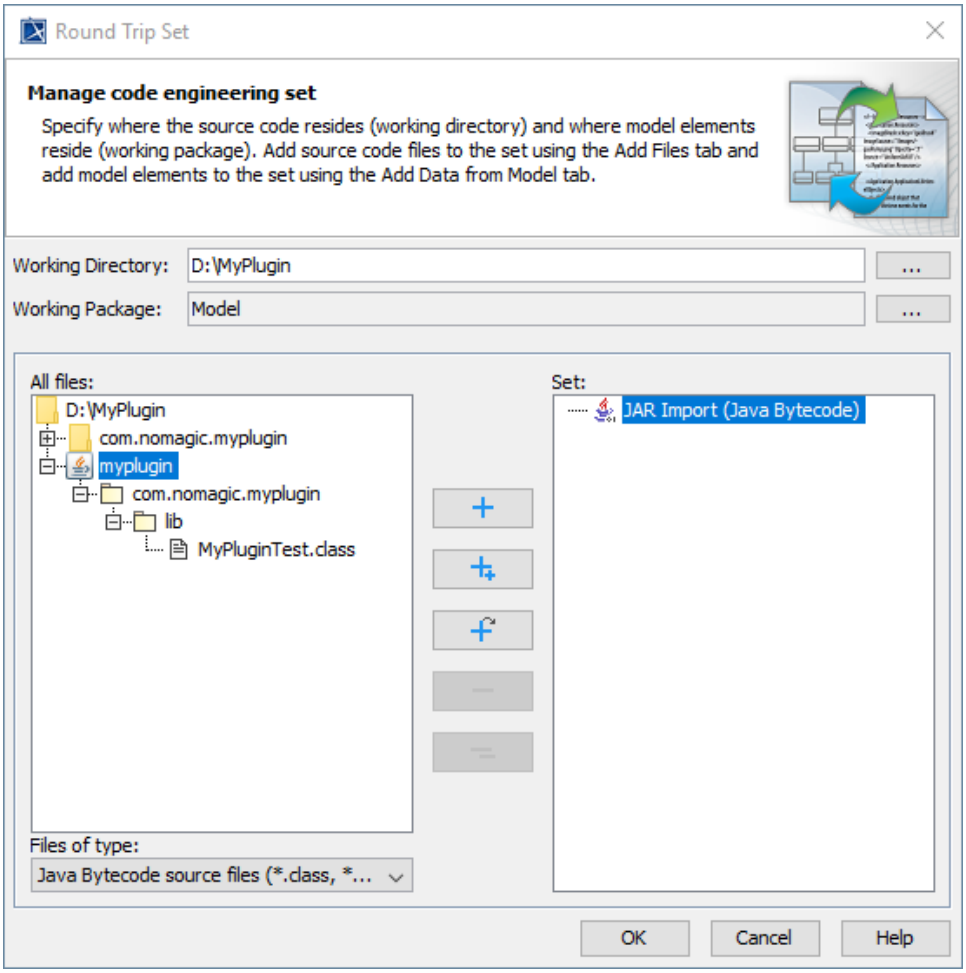
1. Create a new UML project or open an existing one.
2. In the Containment tree, right-click **Code Engineering Sets**. Select **New** > **Java Bytecode**.



3. Name your new Java Bytecode engineering set and click **OK**. The new Java Bytecode engineering set is created under **Code Engineering Sets** in the Containment tree.



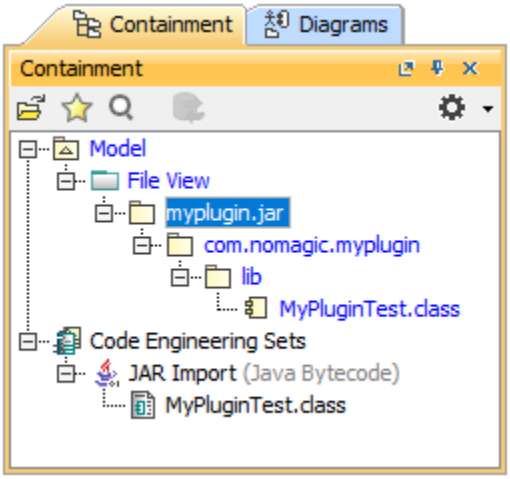
4. Right-click your newly created Java Bytecode engineering set. Select **Edit**.
5. The **Round Trip Set** dialog opens. In the **Working Directory** box, specify the root directory of the .JAR file that you want to import.



6. In the **All files** list, select the .JAR file you want to import.

> **Note**
> The file hierarchy of the imported .JAR file in the Containment tree will start from the working directory you set. For example, if you set the working directory as the MagicDraw installation directory, the whole file hierarchy that the imported .JAR file is in the installation directory will be also imported into the Containment tree.

7. Select  and click **OK**. The imported .JAR file is now in the Containment tree under the **Model** Package, **File View** Package, and the rest of the file hierarchy of the imported .JAR file.



Related Pages:

Unknown macro: 'list-children'

Unknown macro: 'list-children'