MyEclipse UML (MyUML) Quickstart

Last Revision: August 31, 2005

Outline

1. Preface

This document was written using Sun JDK 1.4.2, Eclipse 3.1 and MyEclipse 4.0. All screenshots are based upon the default user interface settings for Eclipse, MyEclipse, and Windows XP. If you experience difficulty with the instruction of this document, please see the User Feedback section for how to provide feedback to the MyEclipse documentation team.

Back to Top

2. Requirements

MyEclipse UML Tools (MyUML) are supported in the following enviornments:

- Supported on Windows 2000 or XP with JDK 1.4 or Java 5
- Supported on Linux with Java 5 (but not JDK 1.4)
- Macintosh OS/X cannot be supported at this time due to an Eclipse bug (see Eclipse bug #67384)

Back to Top

3. Introduction

MyUML provides developers the MyEclipse developers with the following UML features:

- UML Diagrams: Use-case, Class, Collaboration, State, Activity, Deployment
- Integrated Diagram Editor, Outline View, and Properties View
- UML Perspective
- Free-form figure drawing tools
- Direct edit of Node and Connection details via Hot-Zones
- Diagrams stored in UML Model Repository File (e.g., cardemo.umr)
- No restriction on location or project type that may contain UML Model Repository files
- Generate Java code from models
- Export models as XMI 1.0 format
- Export diagrams in image format: GIF, PNG, PS, EPS, SVG
- Reverse-engineer UML class diagrams from any MyEclipse J2EE project or Eclipse Java project
    - Batch process reverse-engineer any combination of project, source-folder, Java package or individual class
    - Drag-n-drop any Java class or interface from the Eclipse PackageExplorer view onto any UML class diagram
    - Customizable reverse-engineering preferences
- Drag-n-drop any UML artifact from the Outline view onto any compatible UML diagram
- Auto-layout option for class diagram

UML Perspective

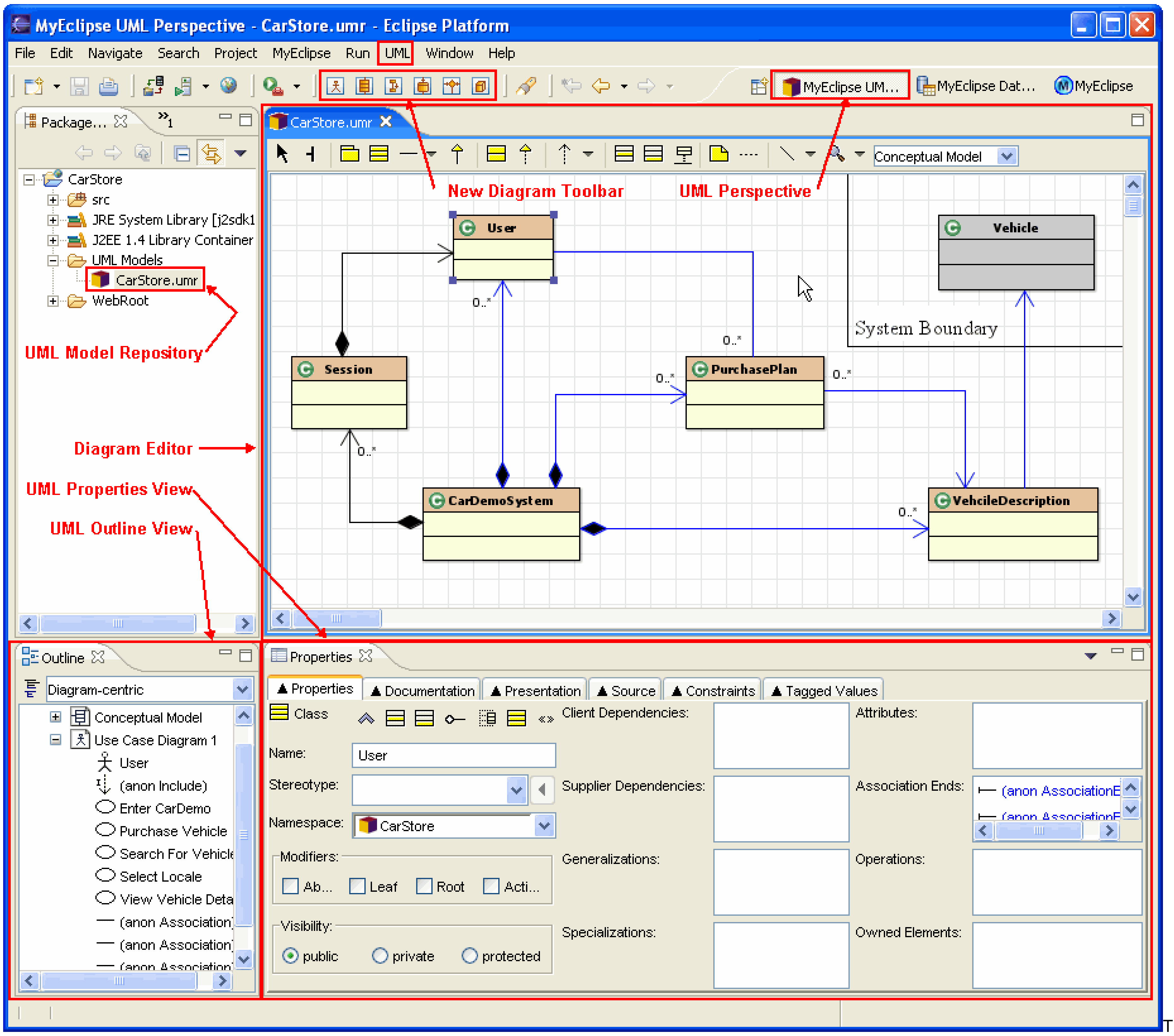


Diagram view: Conceptual Model

Outline: Diagram-centric

Name: User

Stereotype:

Namespace: CarStore

Visibility: ● public ○ private ○ protected

The MyEclipse UML Perspective provides a workbench organization tailored to activities of the modeling process. Default views include the UML Diagram Editor, Outline View, and custom Properties View. Toolbar actions are contributed to make new diagram creation quick and simple. An annotated class diagram example is shown below.

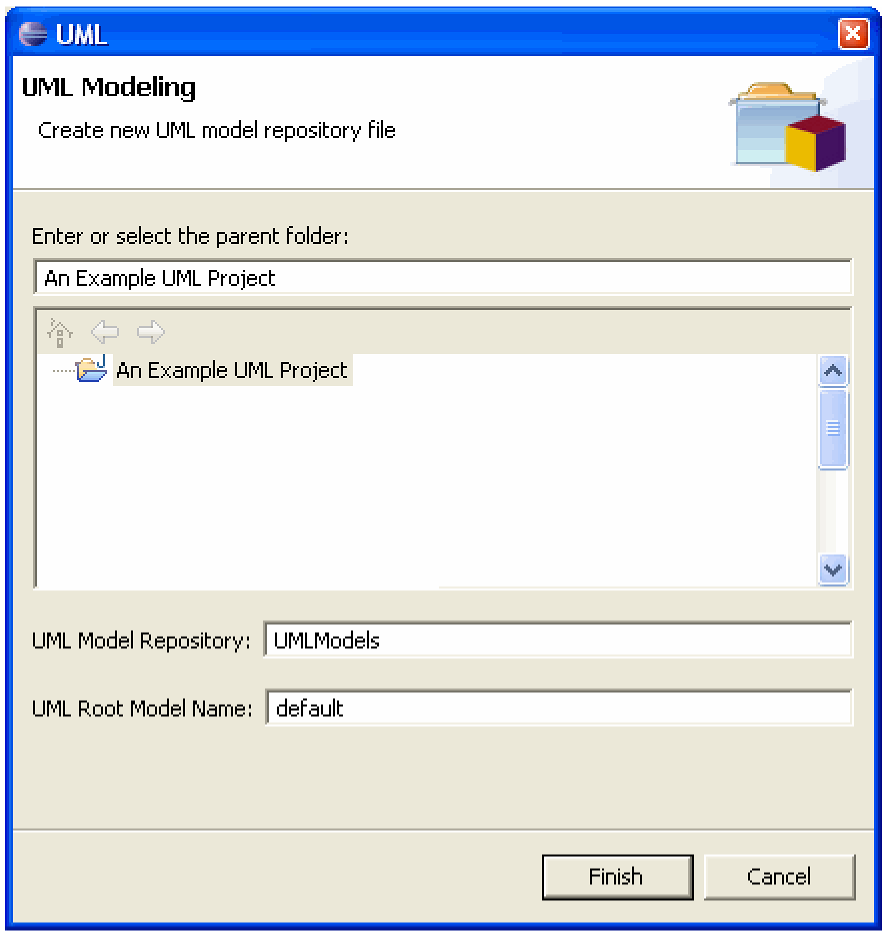
Back to Top

4. Creating a UML Model Repository

MyEclipse UML stores all UML diagrams and model elements in a special file known as a UML Model Repository (UMR). A UMR has a file extension of " .umr" and is depicted with the icon in the PackageExplorer and Navigation views. You may create any number of UMR files in any type of Eclipse project.

The 1st step in working with new MyUML diagrams is to create a UML model repository using the New UMR Wizard.

1. Launch the New UMR Wizard (see figure below) from the menu *File > New > Other... > MyEclipse > UML > UML Model Repository*, or from the MyEclipse UML Perspective at *File > New... > UML Model Repository*.
2. Complete the wizard by navigating to workspace folder that will contain the new UMR file and by specifying the name of the UML model repository.



UML Modeling

Create new UML model repository file

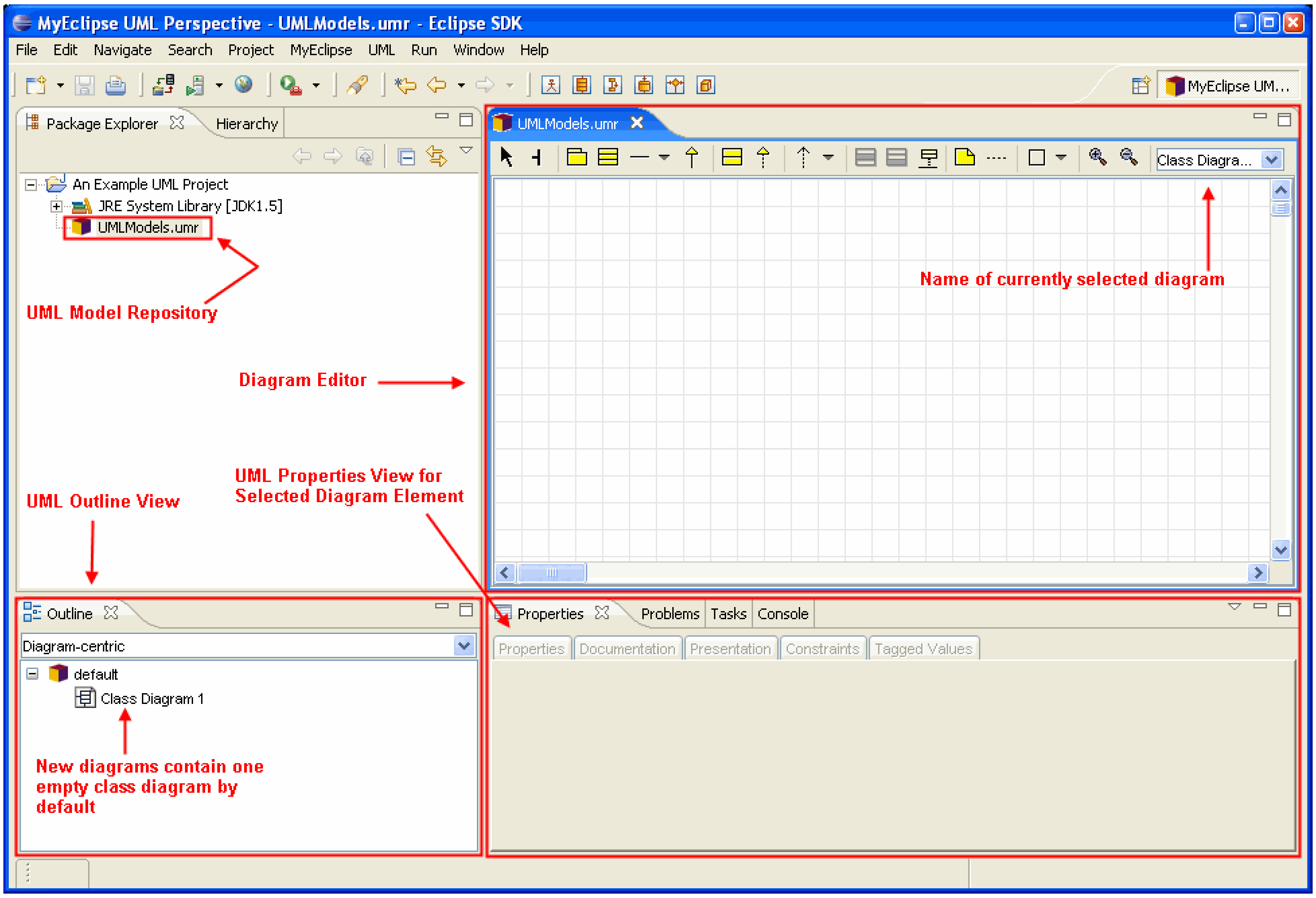
Enter or select the parent folder: An Example UML Project

An Example UML Project

UML Model Repository: UMLModels

UML Root Model Name: default

3. Select *Finish* to create the repository at the location you specified and to open MyUML Diagram Editor as shown below. Note that a new UMR file is created with an empty class diagram.



UML Model Repository

Name of currently selected diagram

Diagram Editor

UML Properties View for Selected Diagram Element

UML Outline View

New diagrams contain one empty class diagram by default

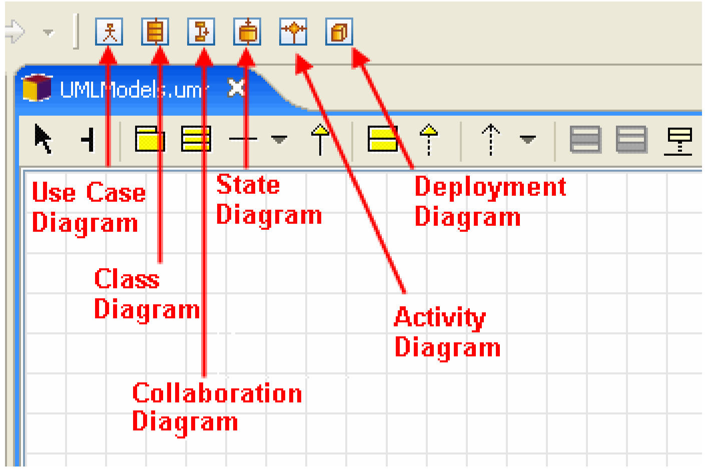
That concludes creating the steps necessary to create a UML repository. In the next section adding diagrams to the model is discussed.

Back to Top

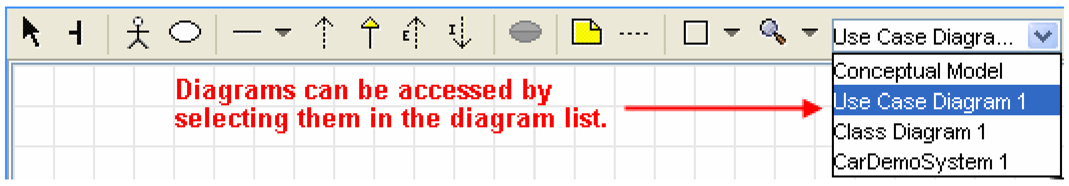
5. Creating and Editing UML Diagrams

5.1 Creating UML Diagrams

A model repository can contain any number of diagrams of any type. To add a new diagram to a model repository, open the UML repository file in the editor and then click on the appropriate new UML Diagram icon from main toolbar, as shown below.

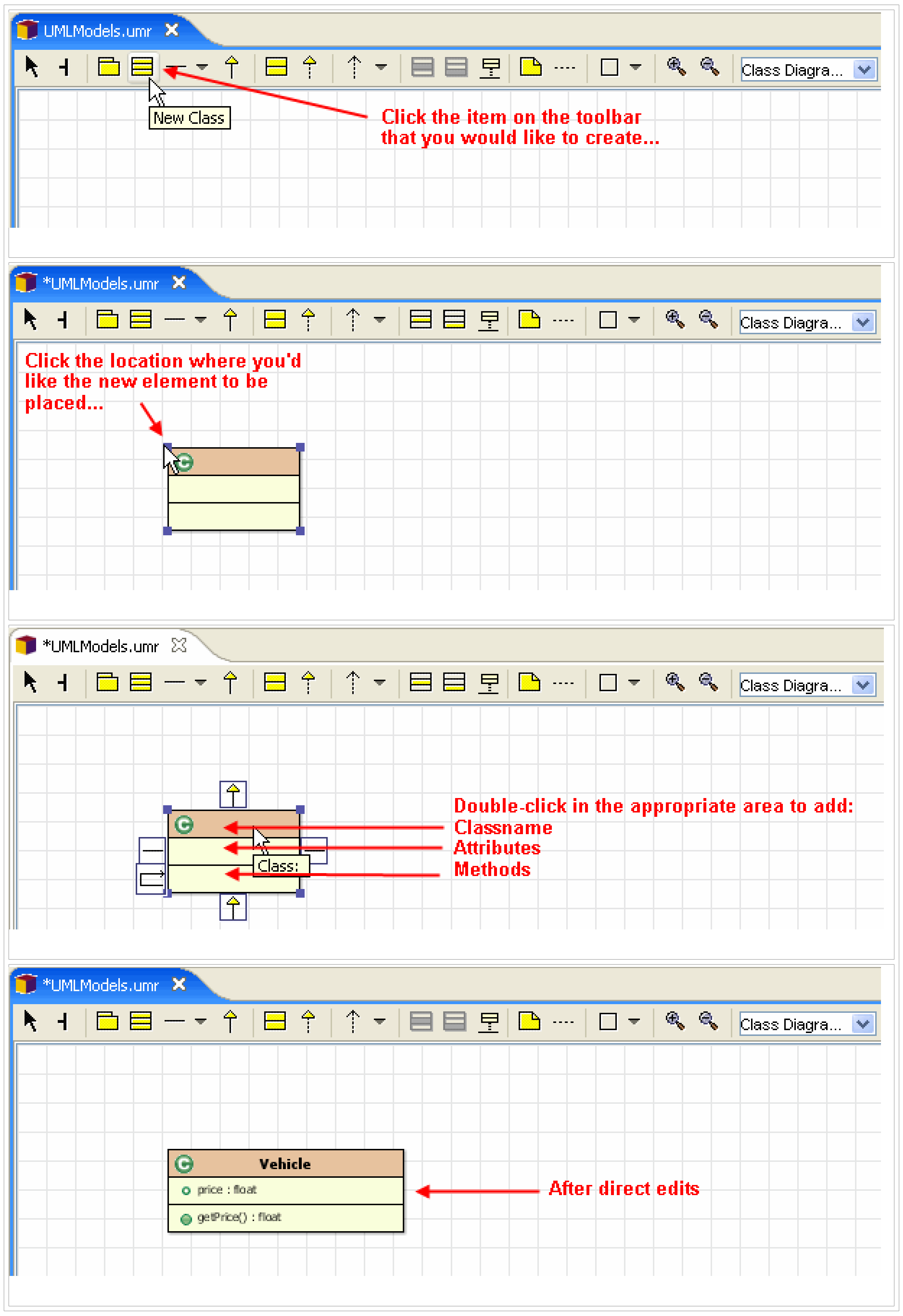


To edit or view a UML diagram select it from either the dropdown diagram list on the Diagram Editor's local toolbar or from the MyUML outline view. The Diagram Editor toolbar is shown below.


Diagrams can be accessed by selecting them in the diagram list.

5.2 Adding Model Elements to Diagrams

Model elements can be added to the current diagram from the UML editor's toolbar simply by clicking on the appropriate element button and then selecting the location on the diagram where the new element should reside. Please note that the toolbar is "diagram-specific", so its contents changes based on the active diagram's type. Below is a sequence of screenshots that show how a class can be added to a class diagram. Any element can be added to other diagrams in a similar way.


Click the item on the toolbar that you would like to create...

Click the location where you'd like the new element to be placed...

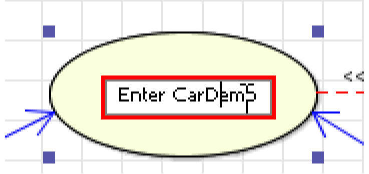
Double-click in the appropriate area to add:
Classname
Attributes
Methods

After direct edits

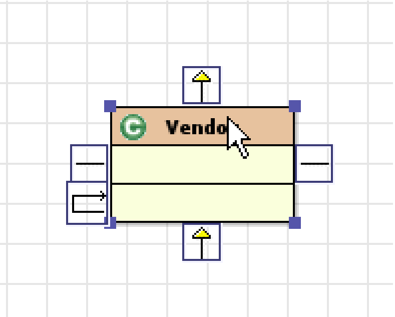
5.3 Editing Diagram Elements

The Diagram Editor enables you to directly modify nodes and connections through a concept known as a hot-zone. There are two type of hot-zones: text and connection.

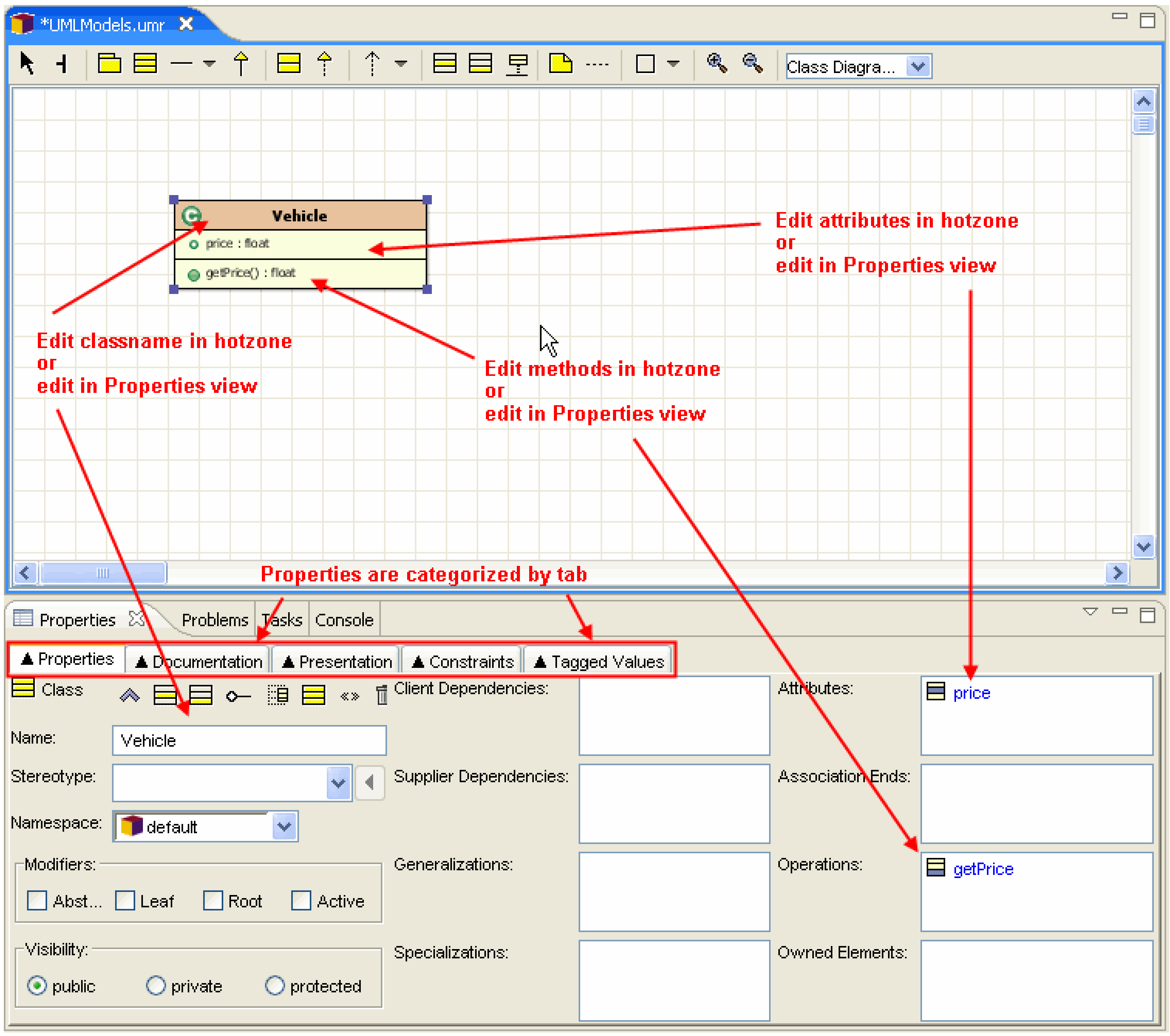
Text hot-zone is an editable region within a node or at the center and end-points of a connection, e.g., association, that is activated by double-clicking within its region. When activated, a hot-zone presents a rectangular editable type-in field. The type-in field accepts direct text entry and delete/cut/copy/paste operations. To commit the changes of a text hot-zone, select anywhere outside of the hot-zone edit region. The type-in region will disappear and be replaced with its content.



Quick Connects are a type of hot-zone that provides fly-over short-cut association and generalization connections on a node. The user can quickly choose the connection type, e.g., association, generalization, self association, and drag the connection to another node to complete the connection.



Properties editing allows any change to an element to be entered through a form-based UI called the Properties View. The Properties View has several pages for easy entry or edit of any attribute associated with an element. Some of the editable fields overlap with changes that can be made in a hot-zone, and others are only available in the Properties view.



Edit attributes in hotzone
or
edit in Properties view

Edit classname in hotzone
or
edit in Properties view

Edit methods in hotzone
or
edit in Properties view

Properties are categorized by tab

Name: Vehicle

Stereotype:

Namespace: default

Modifiers: □ Abst... □ Leaf □ Root □ Active

Visibility: ● public ○ private ○ protected

Attributes: price

Operations: getPrice

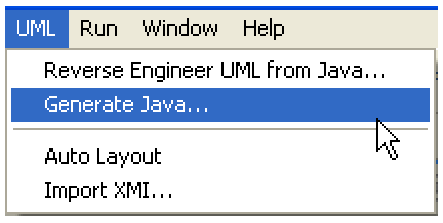
[Back to Top](#)

---

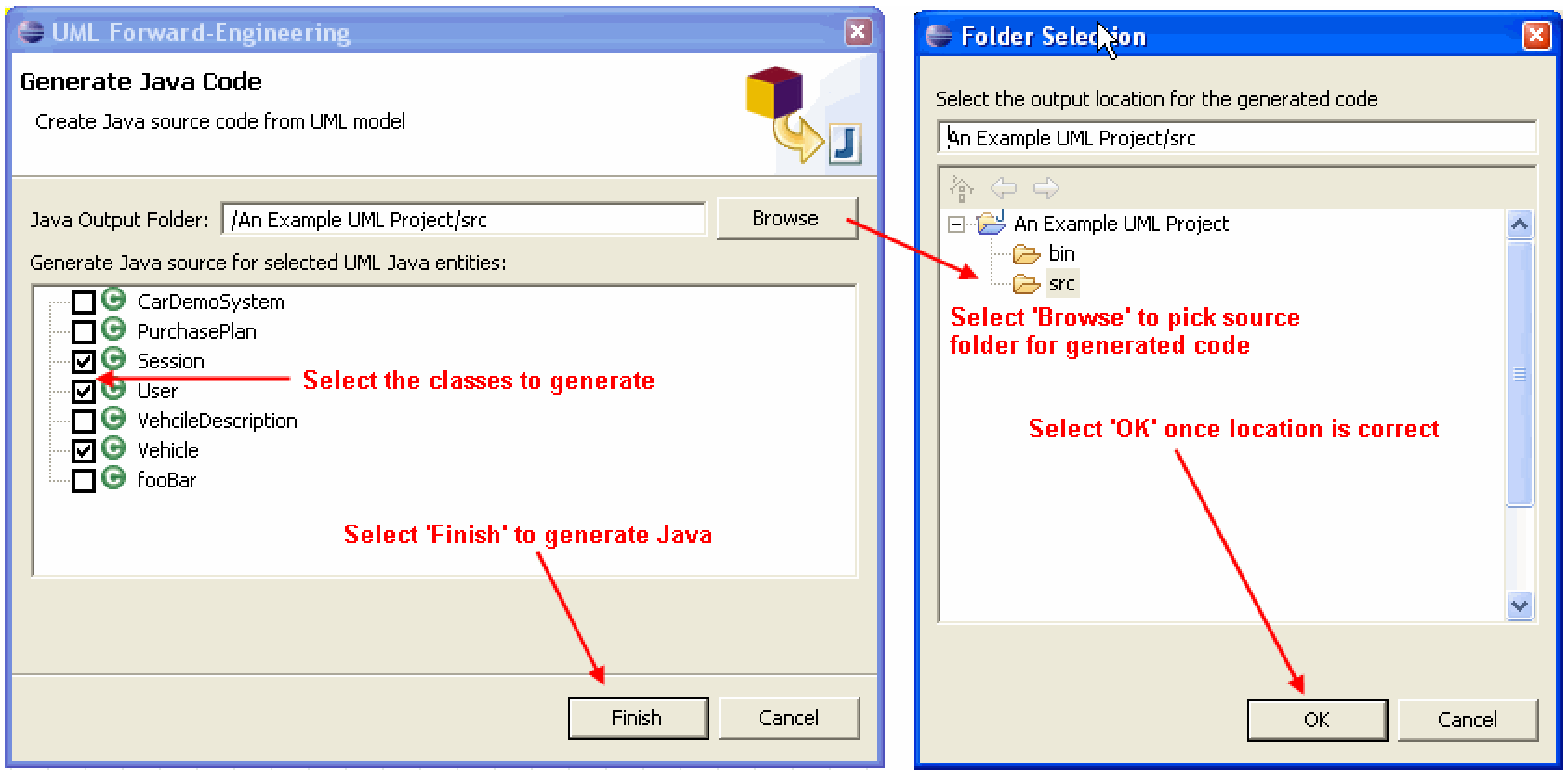
6. Forward Engineering - UML Class Diagrams to Java Code

MyEclipse UML enables you to generate Java code directly from the class diagrams of your UML model.

1. Open the UML repository with the class diagrams you wish to generate Java code
2. Launch the Forward-Engineering Wizard from the menubar, *UML > Generate Java...*



1. Select the source folder of the MyEclipse J2EE Project or any Eclipse Java project into which the new UML->Java classes will be created.
2. Select 1 or more UML classes to translate to Java code
3. Select Finish to initiate the code generation process



Generate Java Code

Create Java source code from UML model

Java Output Folder: /An Example UML Project/src

Generate Java source for selected UML Java entities:

- ☐ CarDemoSystem
- ☐ PurchasePlan
- ☑ Session
- ☑ User
- ☐ VehcileDescription
- ☑ Vehicle
- ☐ fooBar

Select the classes to generate

Select 'Finish' to generate Java

Select the output location for the generated code

An Example UML Project/src

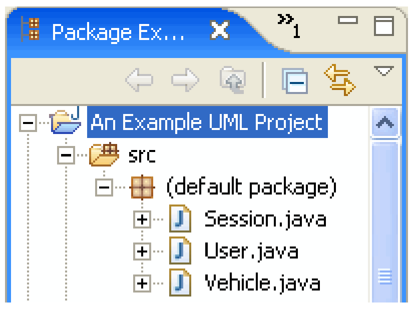
An Example UML Project

Select 'Browse' to pick source folder for generated code

Select 'OK' once location is correct

WARNING: If a Java class already exists in the output location for a forward-engineered UML class, the existing Java class will be deleted and replaced by the new Java class.

After generation is complete, the newly generated Java classes will appear in the specified source folder.



[Back to Top](#)

---

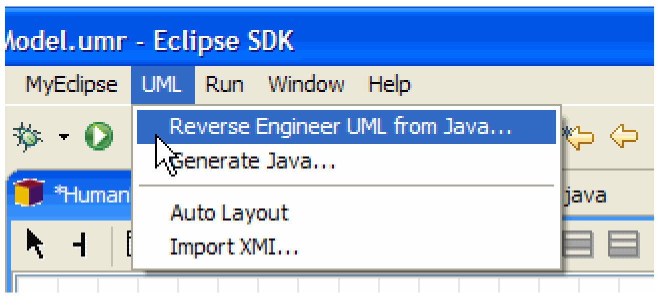
7. Reverse Engineering - Java Code to UML Class Diagrams

Reverse engineering from Java code to UML can be done in two different ways, through batch processing or through drag-and-drop. Both options will be described in this section.

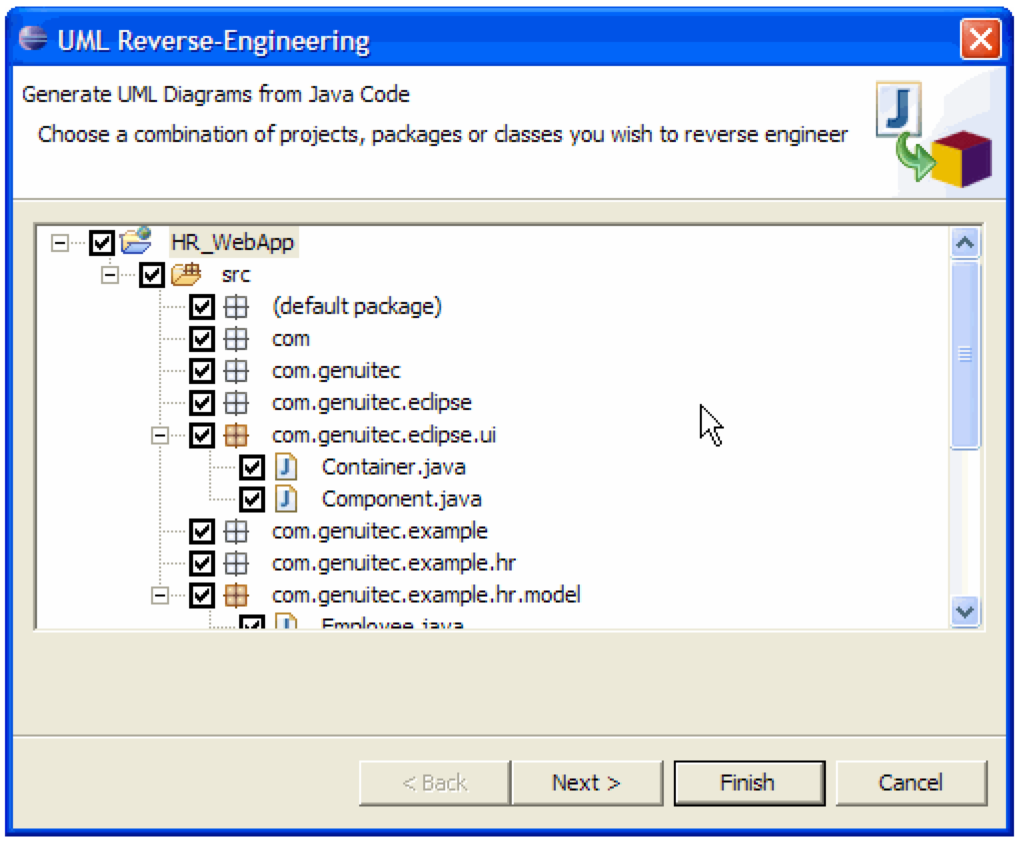
7.1 Batch Mode - Reverse-Engineering

The MyEclipse UML Reverse-Engineering tools enable you to import the Java classes and interfaces from any Java project, source folder, Java package or source file.

1. Open the UMR file
2. Launch the MyEclipse UML Reverse-Engineering Wizard. From the Eclipse menubar select *UML > Reverse Engineer UML from Java...* .



1. Navigate to and select the Java source files to reverse-engineer.



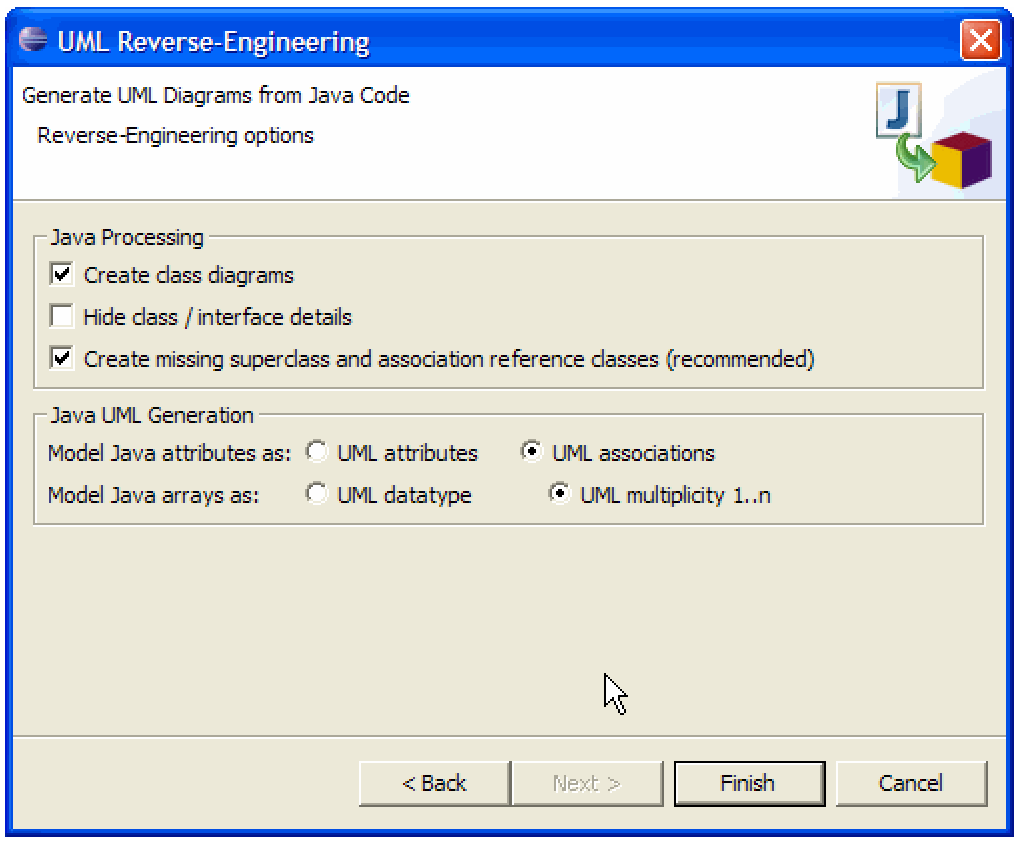
UML Reverse-Engineering

Generate UML Diagrams from Java Code

Choose a combination of projects, packages or classes you wish to reverse engineer

- ☑ HR_WebApp
  - ☑ src
    - ☑ (default package)
    - ☑ com
    - ☑ com.genuitec
    - ☑ com.genuitec.eclipse
    - ☑ com.genuitec.eclipse.ui
      - ☑ Container.java
      - ☑ Component.java
    - ☑ com.genuitec.example
    - ☑ com.genuitec.example.hr
    - ☑ com.genuitec.example.hr.model
      - ☑ Employee.java

1. Specify your processing preferences.
2. Select Finish to initiate reverse-engineering process.



UML Reverse-Engineering

Generate UML Diagrams from Java Code

Reverse-Engineering options

Java Processing

☑ Create class diagrams

☐ Hide class / interface details

☑ Create missing superclass and association reference classes (recommended)

Java UML Generation

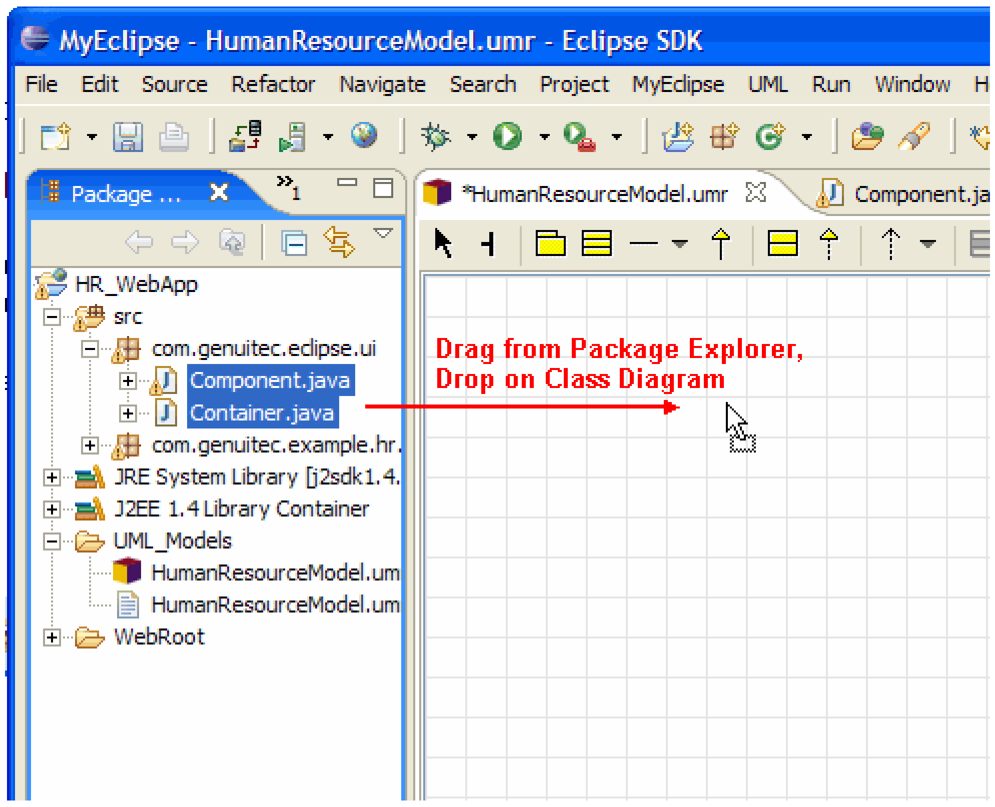
Model Java attributes as: ○ UML attributes ● UML associations

Model Java arrays as: ○ UML datatype ● UML multiplicity 1..n

7.2 Drag-and-Drop - Reverse-Engineering

You can seamlessly add Java classes and interfaces to any UML class diagram using drag-n-drop operations.

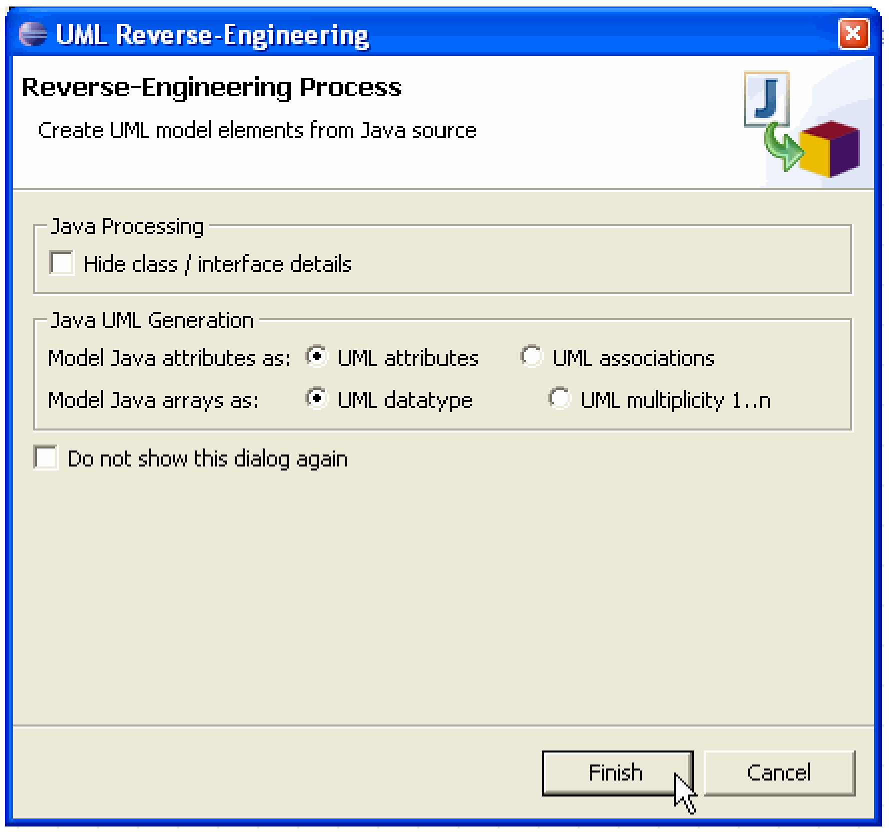
1. From the Java PackageExplorer View, select one or more classes or interfaces
2. Drag and drop the selected classes onto the active UML class diagram in the UML Diagram Editor.



MyEclipse - HumanResourceModel.umr - Eclipse SDK

Drag from Package Explorer, Drop on Class Diagram

1. After dropping the Java classes, the following dialog will allow customization of the generated content. The default settings are highly recommended.
2. Select Finish to initiate reverse-engineering of the selected Java classes.



UML Reverse-Engineering

Reverse-Engineering Process

Create UML model elements from Java source

Java Processing

☐ Hide class / interface details

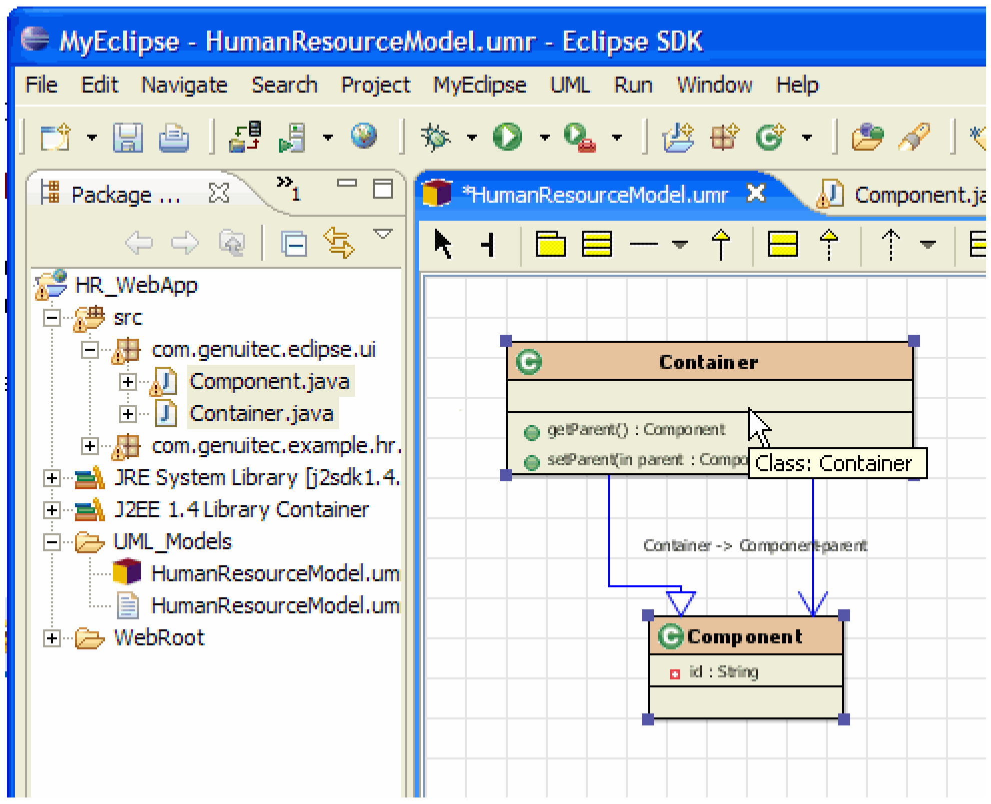
Java UML Generation

Model Java attributes as: ● UML attributes ○ UML associations

Model Java arrays as: ● UML datatype ○ UML multiplicity 1..n

☐ Do not show this dialog again

Upon completion, the class diagram will show the new additions, including any associations between them, as shown below.

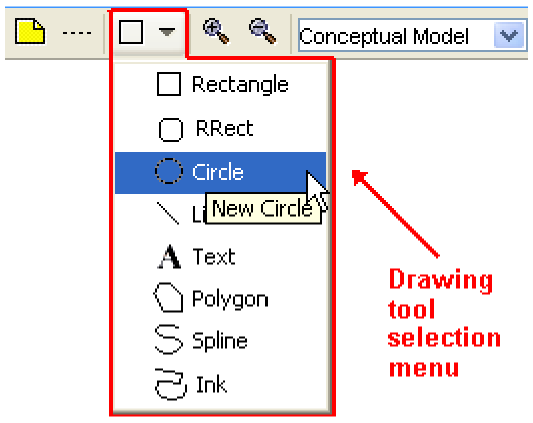


[Back to Top](#)

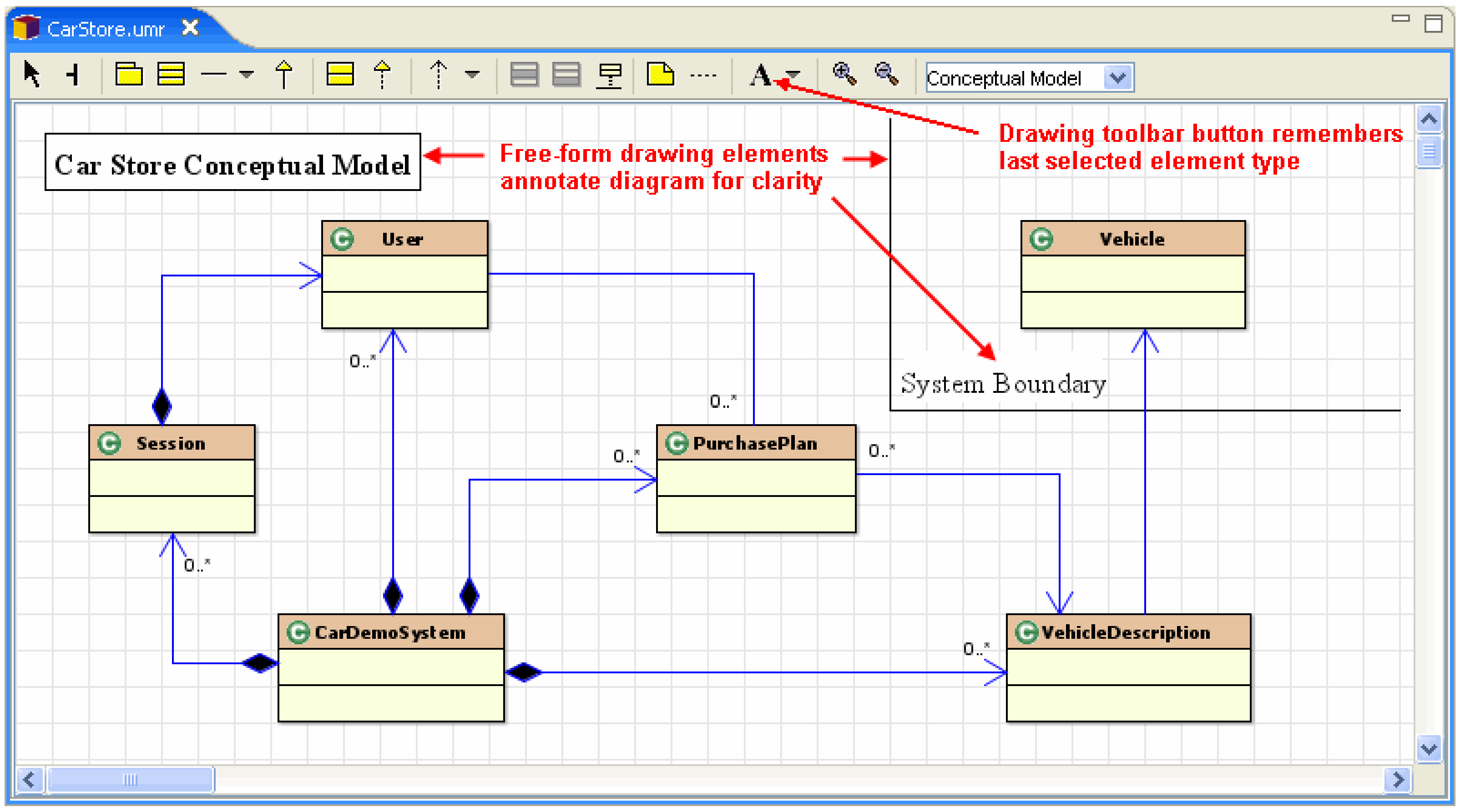
---

8. Drawing Tools

Since it's often useful to annotate diagrams with some amount of free-form drawing or text, MyUML includes a set of drawing tools for simple shapes and text which can be used on any type of diagram. The drawing tool selection menu is in the main diagram toolbar and is used to select the element type.



Drawing tool selection menu


CarStore.umr
Conceptual Model
Car Store Conceptual Model
Free-form drawing elements
annotate diagram for clarity
Drawing toolbar button remembers
last selected element type
User
Vehicle
System Boundary
Session
PurchasePlan
CarDemoSystem
VehicleDescription
0..*
0..*
0..*
0..*
0..*
0..*

[Back to Top]

9. Compatibility with Argo UML

MyUML is backward compatible with Argo UML's .zargo file format. To use an existing Argo model file with MyUML do the following:

1. Copy the Argo file into an Eclipse project.
2. Rename the file to have a .umr file extension. The .umr extension enables MyUML to recognized it as a compatible UML repository file.
3. Open the new UMR file in the Diagram Editor to begin browsing and editing diagrams.

[Back to Top]

10. References

To learn more about UML visit the UML Resource Center.

[Back to Top]

11. User Feedback

If you have comments or suggestions regarding this document please submit them to the If you have comments or suggestions regarding this document please submit them to the MyEclipse Documentation Forum .

[Back to Top]