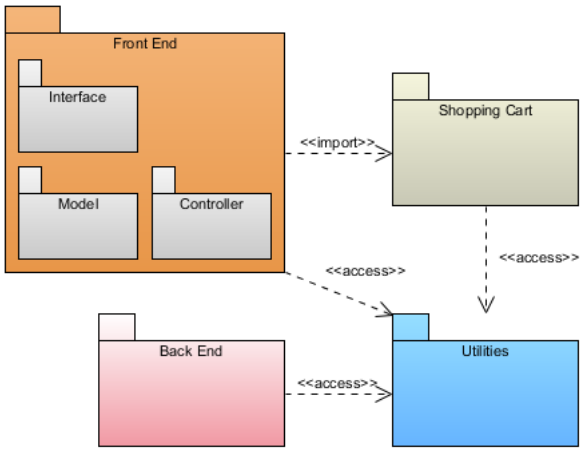
UML Package Diagram

Written Date : July 6, 2011

When modeling a large scale system, you would probably be working with a high volume of model elements. They describe a model from different views and different phases, hence are in different types. UML package helps to organize and arrange model elements and diagrams into logical groups, through which you can manage a chunk of project data together. You can also use packages to present different views of the system's architecture. In addition, developers can use package to model the physical package or namespace structure of the application to build. With this, they follow the convention of naming package like using lower case for Java package.

Package diagram visualizes packages and depicts the dependency, import, access, generalization, realization and merge relationships between them. Package diagram enables you to gain a high level understanding of the collaboration among model elements through analyzing the relationships among their parent package. This also helps explain the system's architecture from a broad view.



Here are some of the notations you can use in package diagram.

Notation	Description
Package	Represents a group of model elements and/or diagrams.
Subsystem	A system that is part of a large scale system.

Dependency	When a package (e.g. client) requires another (e.g. supplier) for specification or implementation, a dependency would be established to model that. Dependency can also be found among model elements contained in packages.
Import	A relationship that shows the model elements in a package which are to be imported from another package. It allows a package to refer to a model element in another package without qualification.
Access	Model elements in a package that can be used by the model elements in another package but cannot be further imported from that package.
Generalization	A relationship that shows a package inheriting from another by sharing its properties and adding its own properties to produce a package with more specific specification.
Realization	An abstraction relationship between two packages showing that the supplier package provides specification while the client package implements it.
Merge	A relationship between two packages showing that their contents are to be combined. This should be used when elements in packages have the same name and represent the same concept.

In this tutorial, we'll use Visual Paradigm for UML to model a simple system with several UML classes in it. We are going to create packages and move the classes into them. We'll also draw package diagram to show the relationships between packages.

Before starting this tutorial, you may want to familiarize yourself with the following resources.

1. Unified Modeling Language (UML) Specification (ref: *7.3.38 Package (from Kernel)* in OMG Unified Modeling Language Superstructure Version 2.4)
2. Users Guide - Resource centric interface

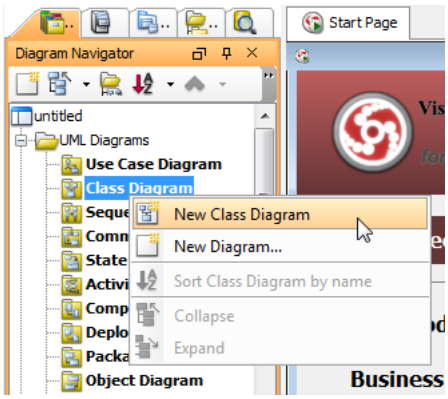
Drawing a Simple Class Diagram

Since we are using package with classes in the following example, let's begin by creating a few classes.

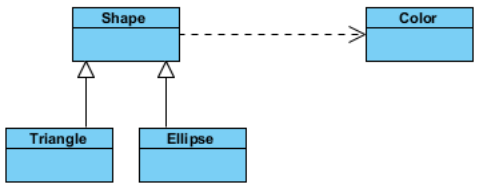
Note: In practice, you may use package with any type of model elements. There is no special requirement.

1. Start **Visual Paradigm for UML.**
2. Create a new project by selecting **File > New Project** from the main menu.
3. In the **New Project** window, name the project *Package Diagram Tutorial*. Click **Create Blank Project**.

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



5. Make use of the diagram toolbar and resource-centric interface to draw a class diagram like this:

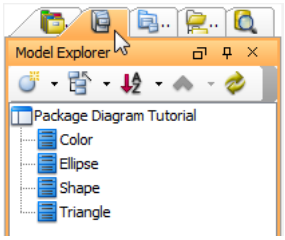


Note that the relationship between Shape, Triangle and Ellipse classes is generalization. The relationship between Shape and Color classes is dependency.

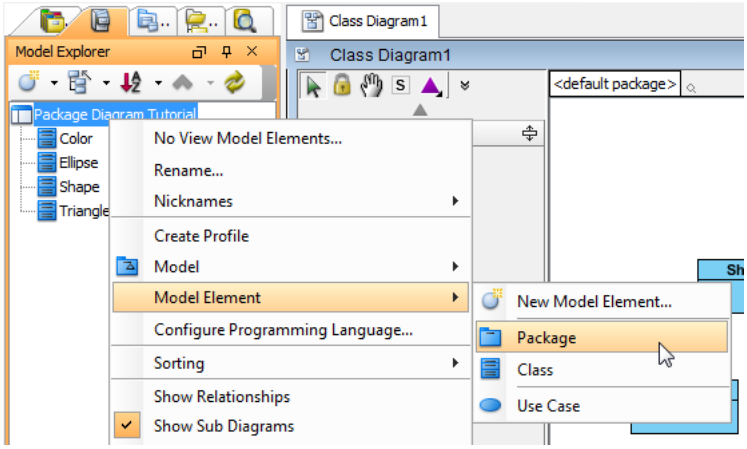
Creating Packages in Model Explorer

We now have four classes. Shape, Triangle and Ellipse belong to model, while Color is a type, required by the Shape class. In this section, we'll create two packages to group the four classes. There are many possible ways for you to create packages. One way is to create them directly in a class diagram containing the classes. In this tutorial, we'll try something different - creating package in the **Model Explorer**.

1. Open the **Model Explorer**, which is the tab next to that of the **Diagram Navigator**.

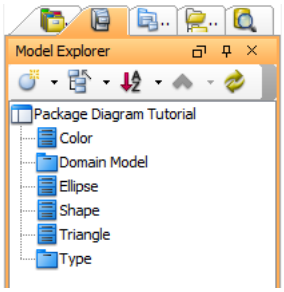


2. Right click at the project root node and select **Model Element > Package** from the popup menu.



3. In the **Package Specification** window, name the package *Domain Model* and click **OK**.

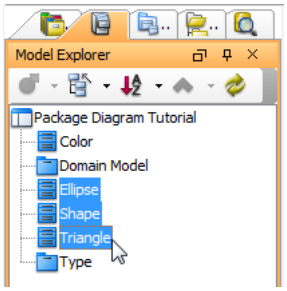
4. Create another package *Type*.



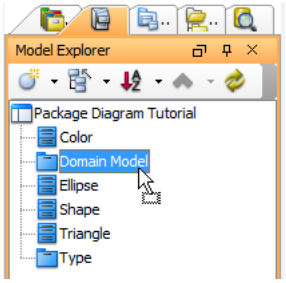
Moving Classes into Packages

You can move model elements into a package through the **Model Explorer** via simple drag-and-drop. In this section, we'll move the classes into the appropriate packages.

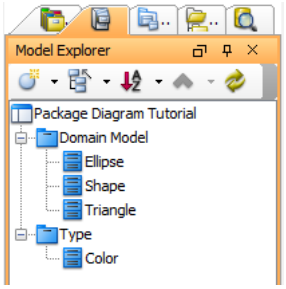
1. Select the classes *Shape, Ellipse* and *Triangle*. You can perform multiple selections by pressing the **Ctrl** key.



2. Drag the selected class into the package *Domain Model*.



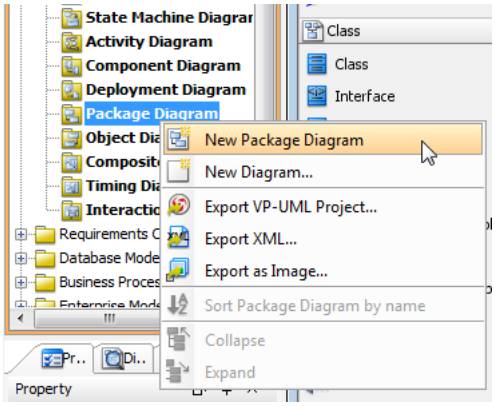
3. Similarly, drag the class *Color* into package *Type*. This forms a hierarchy.



Drawing Package Diagram

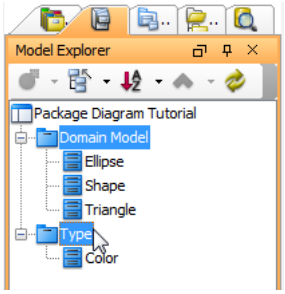
Now, you can draw a package diagram to show the relationships among the classes from a high-level view, through modeling with their parent packages.

1. Go back to the **Diagram Navigator**.
2. Right click on **Package Diagram** and select **New Package Diagram** from the popup menu.

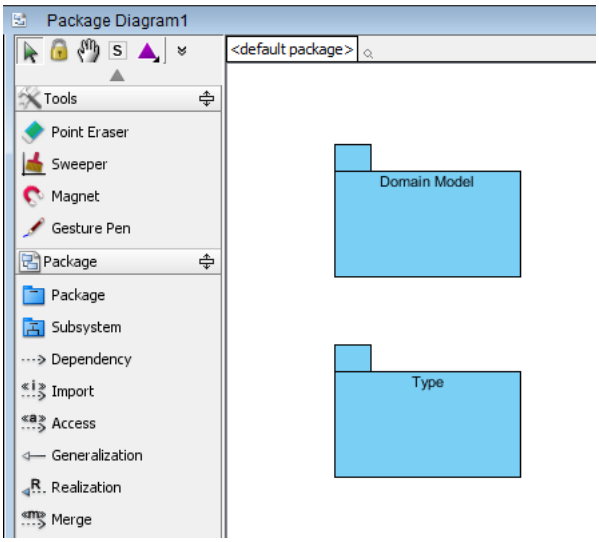


3. Open the **Model Explorer**.

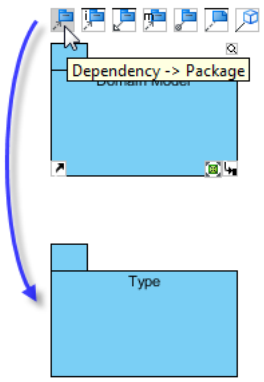
4. Select the two packages. Again, press the **Ctrl** key for multiple selections.



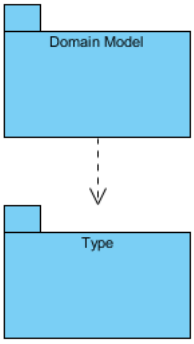
5. Drag the selected packages onto the package diagram just created. Release the mouse button. You should obtain a diagram similar as below:



6. You know that the *Shape* class in *Domain Model* depends on the *Color* class in *Type*. Therefore, add a dependency between the parent packages to represent this fact. Move the mouse pointer over the package *Domain Model*. Press on the resource icon **Dependency -> Package** and drag to the package *Type*.



7. Release the mouse button within *Type*. The diagram should now become:



Related Links

1. YouTube Video: Package Diagram in UML
2. Know-how: Organize Project Data with Model

Visual Paradigm for UML home page
(http://www.visual-paradigm.com/product/vpuml/)

UML tutorials
(http://www.visual-paradigm.com/product/vpuml/tutorials.jsp)

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