**Modelio Tutorial**

Download & Install

1. Download Modelio Open Source: <https://www.modelio.org/downloads/download-modelio.html>
2. I can’t remember the installation process, but if there is a choice, you need to include/install the “Java Designer” module and the “Modeler” module. These may be automatic.

Workspace & Projects

1. When you run the program, it will ask you for a workspace. By default it will use: users/userID/modelio/workspace. You can redirect this wherever you like and name the “workspace” anything you like.
2. Create a *project* named *Company* in your *workspace*: Choose: File, Create Project. Give it the name, *Company* and be sure and check “Java project”. (We are going to model several Java classes: Company ->\* Employee.) The Model Explorer looks like this:



1. Notes:
2. Why it shows “Company” three times is beyond me. The highlighted one above is where all our artifacts will be stored.
3. You can only switch workspaces (File, Switch workspace) when you first run the program. Once a project is open, you can’t switch workspaces.
4. You can have only one project open at a time.
5. Projects are stored in folders under the workspace folder.

Workspace & Projects

1. Right-click the “company” folder and choose: Create diagram, class diagram. The empty class diagram will be displayed. Notice that there is a toolbox with icons for creating an interface, class, attribute, operation, etc.
2. Create a class: Click the “Class” icon once, then click on drawing surface. It creates a class named “Class” in the Model Explorer and display an rectangle for the class on the class diagram. Change the name to “Company.”
3. Right-click the “Company” class in the Model Explorer or the diagram and choose: Edit element
4. Choose the Java tab and check “Java Element”. This is important. If not checked, when you go to generate code, this class will not have code generated.
5. Generate the code for the class: right-click the “Company” class in the Model Explorer and choose: Java Designer, generate. This will create a “src” folder inside your “Company” folder in the file system (doesn’t show this in Modelio)
6. View the code: right-click the “Company” class in the Model Explorer and choose: Java Designer, Edit. The marked-up code will be displayed. The markup is for the diagram. You can turn that off as we will do later.

**import** com.modeliosoft.modelio.javadesigner.annotations.objid;

*@objid ("20a23023-e57b-47d9-b99c-c1828e7989a1")*

**public** **class** Company {

}

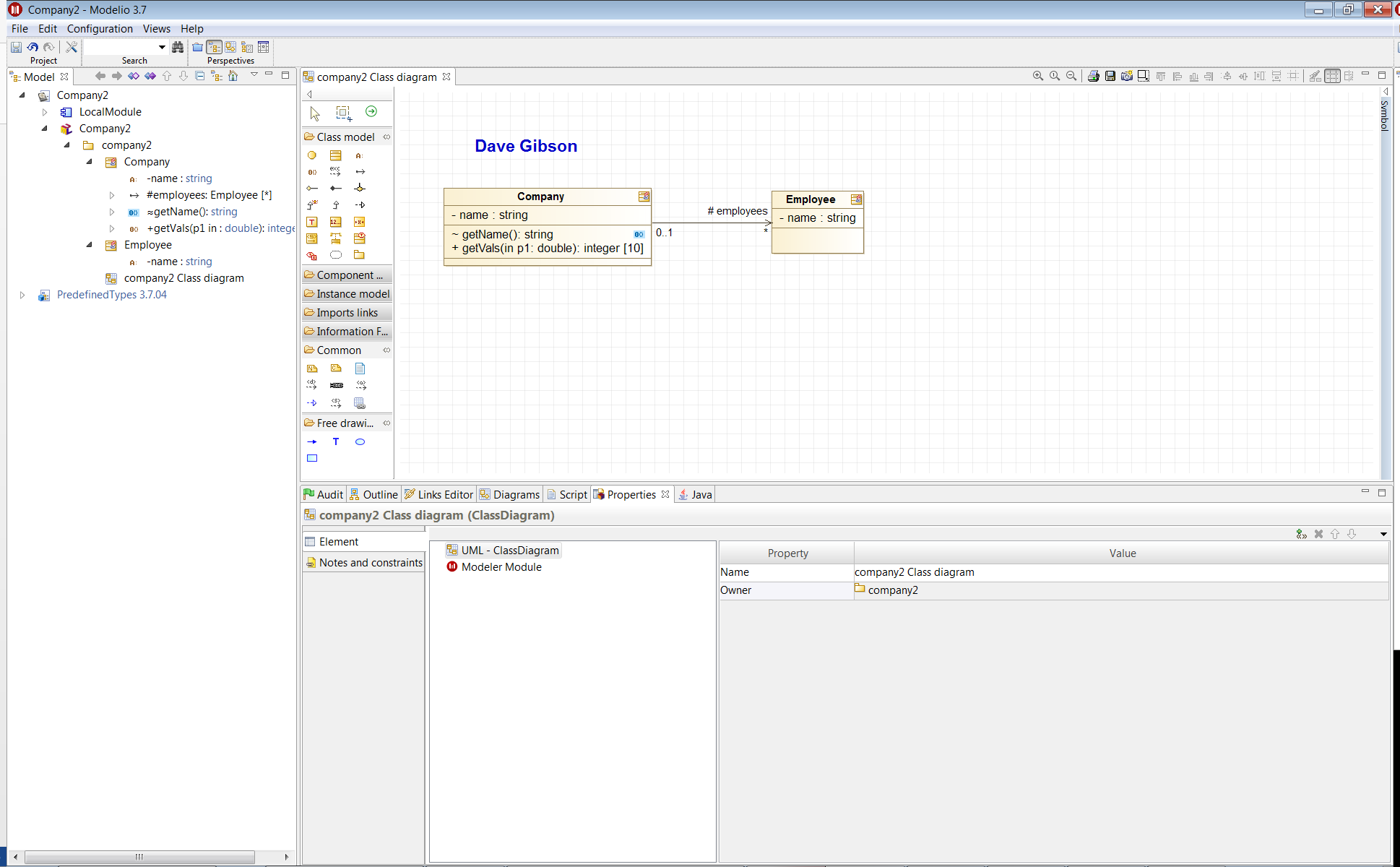
1. Add an attribute to the *Company* class: Click the attribute icon once, then click in the Company rectangle anywhere. An attribute is added: *Attribute: string.*
2. Select the attribute in the drawing or in the Model Explorer and then activate the “Property” tab in the lower portion of the screen, below the drawing. Change the Name to “name”, and the Visibility to “Private”.

Note that Access mode is set to “Read/Write”. Thus, when you click the drawing, a getter and setter will be displayed. It may not show until we do the next step.

1. Add an operation that accepts a double and returns an array of 3 integers. (This is just to illustrate an array, among other things. Do the following
2. Click the “Operation” icon and then click in the *Company* rectangle. An operation named *Operation* will be displayed.
3. Right-click the the *Operation* method in the Model Explorer and choose: Edit element.
4. The Operation tab should be active. Change the Name to, “getVals”
5. In the “Operation parameters” section below, click the icon to create a parameter. It will be named *p1* which you can change if you want.
6. Change the type of the parameter to double.
7. Click the icon to “Create a return parameter”. It will create one of type string. Change the type to “int” and the Multiplicity to “10”. Notice a preview of the signature is shown in the bottom left of the dialog. Press close and your diagram is updated. Stretch the box if needed.
8. Generate the code for the class: right-click the “Company” class in the Model Explorer and choose: Java Designer, generate. Then, view the code: right-click the “Company” class in the Model Explorer and choose: Java Designer, Edit. The marked-up code will be displayed. Note that it implemented the integer array as List<Integer>. There is probably an option to force it to use an array.
9. Add an “Employee” class. Don’t forget to “Edit element”, Java tab, and check “Java Element”. There is some way to make all classes “Java Elements”, but don’t know right now. You cannot generate java code unless you have done this. For example, if you create a class, then right-click, there is no “Java designer” choice.
10. Add an association: click the association icon, then click on the Company class then on the Employee class.
11. Select the association in the diagram (or right-click association in Model Explorer and choose Edit element, or simply double-click the association in the Model Explorer) and in lower portion of the screen you will see a bunch of tabs. Choose the “Properties” tab. Note the things you can change: multiplicity, navigable, visibility, etc. Do the following:
12. Set the “Multiplicity max” for “To: Employee” to “\*”. Thus, changing from the default 1-1 relationship to a 1-many.
13. Change “Role” to “employees”. This is the name of the instance variable that will be in the Company class (collections should always have a plural name).
14. Change “Visibility” to “Protected”
15. Generate the code for all the classes: right-click the “company” node in the Model Explorer (not the class, the node above the class) and choose: Java Designer, generate. Then, view the code: right-click the “Company” class in the Model Explorer and choose: Java Designer, Edit. View the code for the Employee class.
16. Do the following:
17. In the Employee code, add this instance variable:

private String name;

1. Save the Employee code: The only way to save the Employee class (as best I can tell), is to close the window and it prompts you to save. Do that now.
2. Inspect the diagram: you will not see the attribute. However, check the Model Explorer and you will see it there.
3. Drag the Employee’s name attribute in the Model Explorer to the Employee class and you will see it appear in the diagram.
4. Periodically, you’ll want to save the project: File, Save project. Interestingly, it doesn’t save the class file if it has been modified. Do that now.
5. To generate the code without the markup: Configuration, Modules, Java Designer, General, Generation mode. Change “round-trip” to “release”.
6. Generate the code, then edit one of the classes to see that the markup is removed.
7. Open the code for either class. Notice that you cannot edit the code – that is because we are not in “round-trip” generation mode.
8. To save the diagram, in the upper-right select the disk icon. Strangely, the default location is not your workspace. Do that now and save in your “Company” folder (above the “src” folder).
9. Add your name to the drawing: From the Toolbox, choose the “Free drawing” divider and the Text icon. Type your name, stretch box to accommodate.
10. **Do the following:**
11. Make a screen shot similar to the one on the below. Crop it so it shows just what I have shown.



1. The image should be saved as a *jpg* or *png*.
2. The image should easily readable without zooming in or out.
3. Place the image in the *HW CDM* dropbox on Blazeview.
4. Continue with the tutorial.

Notes:

1. If you delete an element from your diagram it deletes it from the model! You probably don’t want that. Instead, right-click the element in the drawing and choose “Mask”. It will be removed from drawing, but not from the Model Explorer. To put it back: hold shift and select “Company” and each of the members of the class and drag back to drawing.
2. To reverse engineer existing code, create a new java project, right-click the project name, choose: Java designer, Reverse, Reverse java application from sources, find the code, select the classes to reverse engineer. It will create the classes in the model explorer. Then you can:
3. Create a class diagram
4. Select classes and all attributes and methods and drag onto drawing surface.
5. Generate the code. This will generate a copy of the classes in your Modelio workspace. Edit a class and you will see that all the code implementation/logic is there.