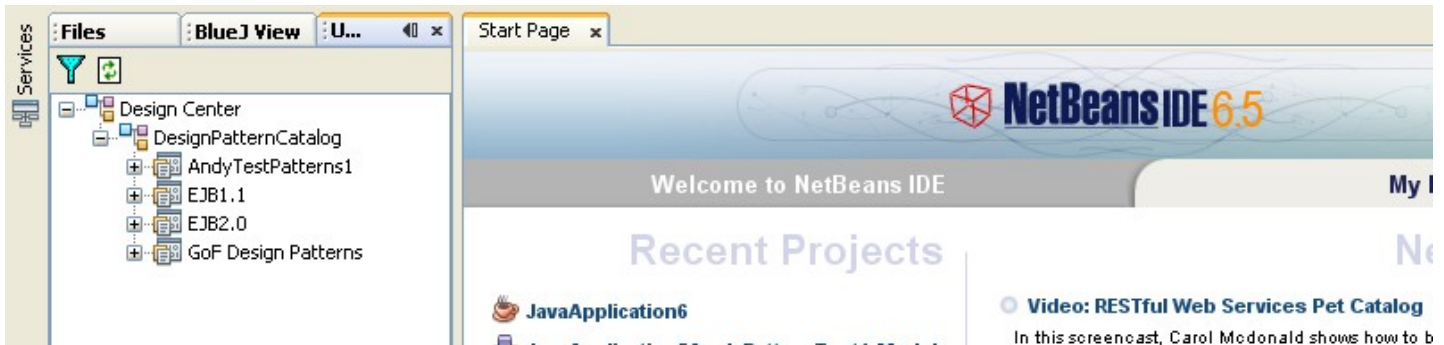


- [Home](#)
- [Design Patterns](#)
- [Blog](#)
- [Products](#)
- [Sitemap](#)
- [Contact](#)

## AndyPatterns

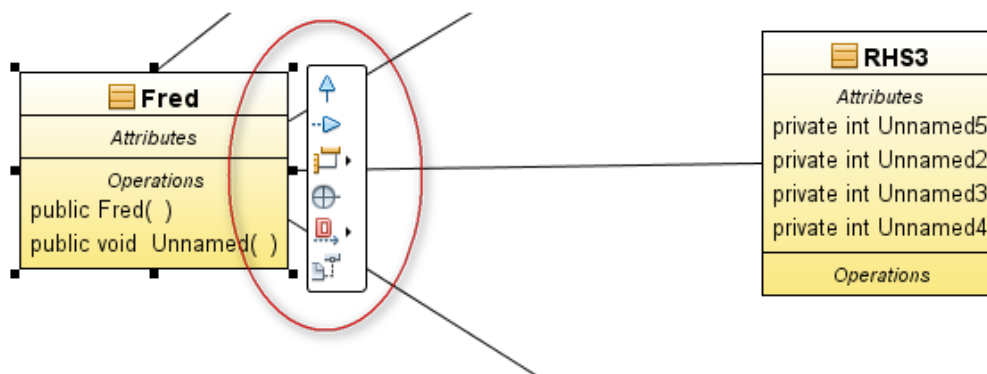


## UML and Design Pattern Support in Netbeans 6.5

Netbeans is an opensource IDE from Sun and is gaining a lot of developer support. I've certainly abandoned Eclipse and switched, as the java facilities are just as good (for me) and importantly Netbeans offers a wonderful GUI form designer plus UML facilities, which Eclipse doesn't come bundled with for free. Let's take a look at the UML and design pattern support in Netbeans and investigate Netbean's cool ability to build "custom" design patterns - visually.

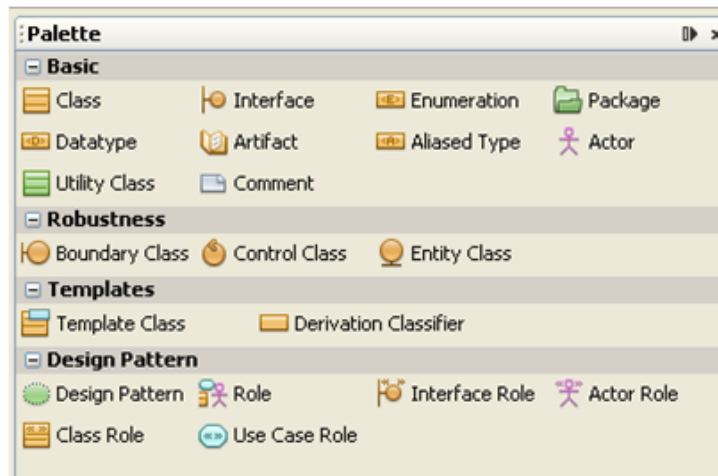
### The UML diagrammer

The UML diagramming is quite good. You drag out objects from a palette or use the "hover palette" next to a class to drag out links to either existing or new classes. This gui metaphor is nice to use and saves you constantly returning to a palette on the side of the screen in order to add objects and connections.



*Use the "hover palette" next to a class to drag out links to either existing or new classes.*

Of course you need a nice palette of UML objects too:



*Netbean's UML palette*

## The UML "Model"

Any UML you do in Netbeans requires you to create a special "UML project". This contains a model of your classes and methods etc. as objects. The UML view is driven off the model.

- You can generate code OUT into another project
- You can reverse engineer code IN from another project

UML markers in the code help code sync e.g.

```
package javaapplicationuml2;

+ UML Marker
public interface Target {

+ UML Marker
    public void request ();

}
```

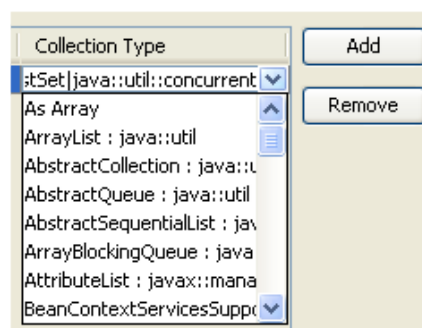
*Markers in generated code - careful not to delete them  
unless you know what you are doing!*

which actually comprise comments with a unique guid e.g.

```
//
// #[regen=yes,id=DCE.548B84AB-2DF4-1944-BA69-85F33CA61C99]
//
```

## UML code generation

UML generates classes, interfaces, enums, methods and attributes. No substantive code is generated. Attributes can be one to many etc. When you click on an association line, select properties and select multiplicity, you get a choice of implementation e.g.



### *A choice of one to many implementations*

A lot of language make do with one list and one hash type and build it into the language (e.g. python, ruby, groovy, javascript). Hey but this is Java and you can't make it too easy! Of course there are special needs for different collection types, so this is a pretty cool way of picking the implementation that you want.

Note that you must specify the association as “navigable” to get an "arrow" on your line and to get any code generated for that association.

Attribute code generation seems a bit buggy and often doesn't compile E.g. Here is a class X having a one to many association with class Z

```
import java.util.ArrayList;      // ok
private Z mZ;      // surely Z should be ArrayList

public ArrayList getZ () {
    return mZ;
}
public void setZ (ArrayList val) {
    this.mZ = val;
}
```


And in the above code you don't get your X.Add(z) method you instead get some strange setter which accepts an entire arraylist - as if anyone does that.

## UML code generation using Freemarker templates

You can build code generation templates to generate more substantial code. Netbeans uses this templating system throughout itself e.g. the default java class stub you get when you create a new java class is a freemarker template.

To generate code using freemarker via UML you need to mark your UML classes with stereotypes and associate those stereotypes with a template.

Unfortunately the whole code generation aspect is currently poorly documented and there is no API published except a code class interface that one is meant to decipher. The one [tutorial](#) uses a 441 line freemarker template to generate a couple of lines of code !! And the Sun developers in this area claim there are no resources to provide any more documentation.

 View the forum discussion [here](#).

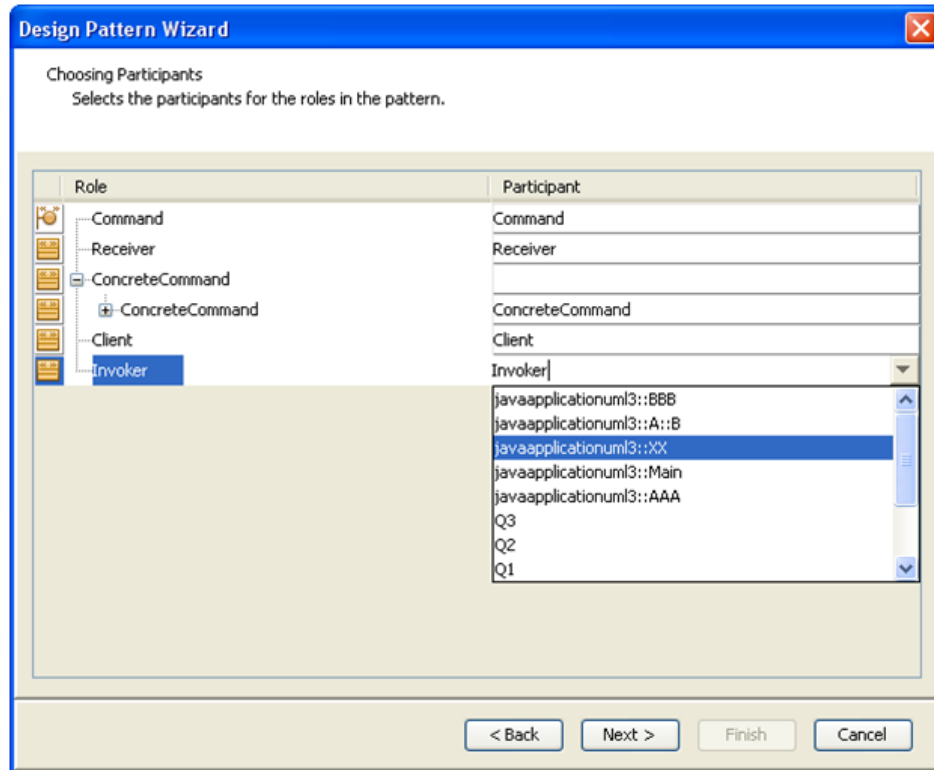
## Design Pattern Support

All GOF patterns supplied as templates which you can graft into your workspace:



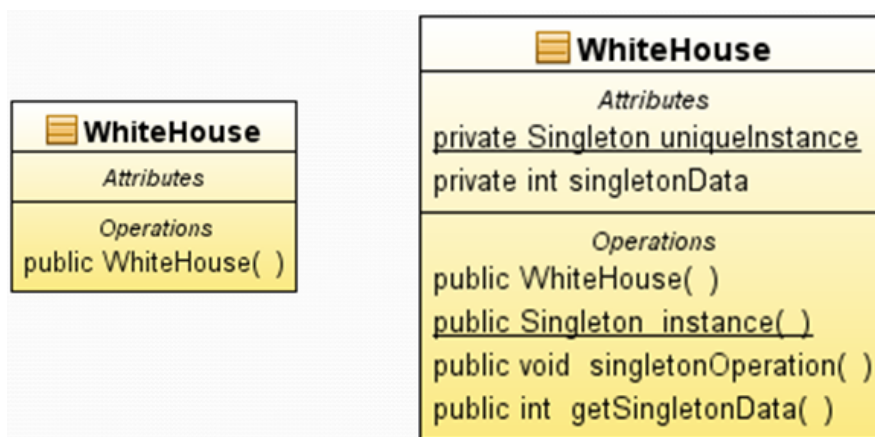
*Invoking the design pattern wizard.*

You can specify existing classes as fulfilling roles, or have the wizard create new classes for those roles:



*Mapping design pattern roles to existing or new classes.*

Again, only the basic structure of the pattern is created – you get no actual substantive code. For example with the Singleton pattern you get the getInstance() method generated but there is no code inside! Here is the class before and after:



*Before and after applying the Singleton design pattern.*

And the code that was generated:

```
public class WhiteHouse {

    private static WhiteHouse uniqueInstance; // correct

    private int singletonData;

    public WhiteHouse ( ) {          // should have been made private
    }                                // to prevent multiple instantiations

    public static WhiteHouse instance ( ) {          // correct declaratation
        return null; // but sorely needs functional code here...instead of returning null !
    }

    public void singletonOperation ( ) {
    }

    public int getSingletonData ( ) {
        return 0; // huh? Shouldn't this return this.singletonData ?
    }
}
```

```

    }
}

```

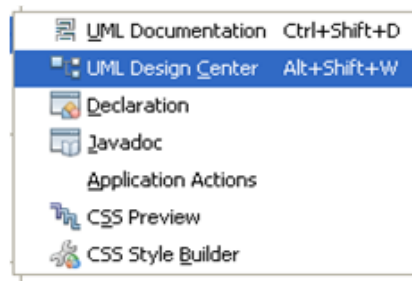
It seems that when generating sub methods, if the method returns a value, there is some default code generation which returns null or 0 or something that will allow the generated code to compile. However this might confuse newcomers who perhaps see this generate code as properly functional in some way. Let me repeat, there is no code generation of body code from design patterns except for the default return values that you normally get with UML code generation.

Yes you can generate code from UML but that requires that you write FreeMarker templates and associate these templates with stereotypes (just set the stereotype property of a uml class or method). The design patterns don't apply these stereotypes, nor do they provide any freemarker code generation templates.

Without code generation the design patterns are of limited value. Code generation is extremely difficult, relying on Freemarker template technology and next to no documentation. I am hoping that "marking" design pattern related classes with stereotypes will solve the code generation problem - but nobody seems to have actually done it. The code generation that exists is buggy anyway esp. with multiplicity of attributes.

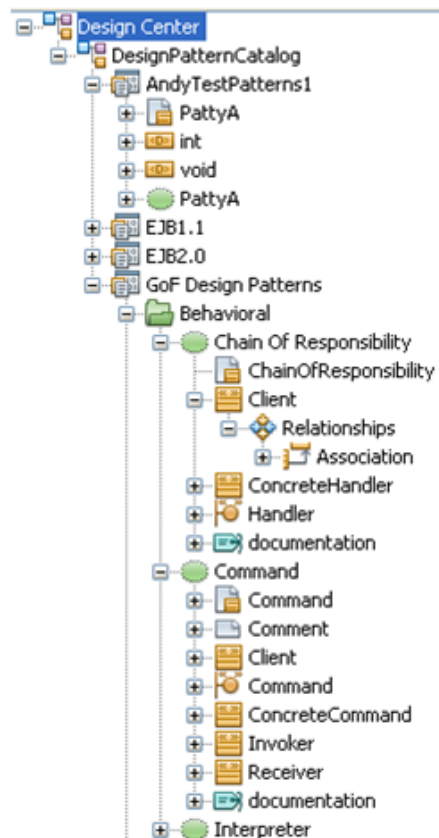
## Custom Design Patterns

One really nice thing is being able to create your own design patterns. No coding or compilation is needed, just draw the diagram and save. From the Tools/Other menus select the UML design centre:



*Invoking the UML Design Centre from a menu*

Here you can view the existing patterns in the patterns "design centre". It is very easy to create a folder and add your own patterns.



*The UML Design Centre, where you can add your own patterns.*

To create your own pattern simply draw a diagram of your pattern classes using a few special diagram objects - the collaboration and role objects:



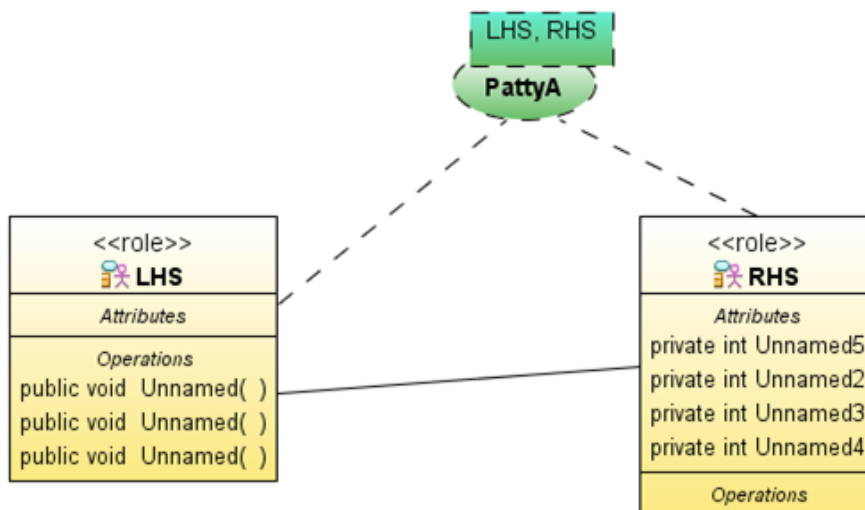
*The UML palette relating to design patterns.*

You can only use the UML palette relating to design patterns whilst in the UML design centre, despite it being visible during normal UML operations. This is a little confusing, especially as I'd love to be able to document my patterns once they have been applied, using the collaboration object - dragging a named role line to each class involved in the pattern...

When building a design pattern, instead of adding a class you add a "class role" object from this special design pattern palette. The role objects are placeholders for a new or existing classes (or use cases or interfaces etc.), which you can fill in using combo drop downs etc. when the pattern is finally applied via the design pattern wizard.

You link all the roles together with a collaboration object and you are done. The collaboration object lets you name your pattern and becomes the anchor for all the roles. Its a pity you don't get collaboration object in regular diagramming in order to document your patterns once they have been applied. You can instead use notes with multiple links (though this doesn't let you label the links) or make do with tagging your UML classes with stereotypes, indicating the roles e.g. Adapter, Adaptee, Client etc.

Here is a pattern I created:



*An example custom design pattern!*

The example is a bit contrived, but you get the idea. Once you build your pattern you can immediately go back to your normal workspace and apply it:



*Applying my new pattern to my workspace.*

## Design Pattern Gotchas

Be careful when using design pattern "automation". The particular pattern implementation may not suit your situation. For example "Adapter" is an object adapter not a class adapter. You need to understand patterns anyway to use these tools.

## Appendix: UML annoyances

- In UML new classes are always created in the package in which the diagram resides, so you may need to move them to the correct package
- Right click shows disabled items in popup menu then right click again at slightly different pixel position and items are enabled again
- Can't copy and paste methods and attributes
- Weird keyboard combinations e.g. CTRL-ALT-SHIFT-N for pan
- Newly generated code/java files don't always appear in the project so you have to close and reopen the target project or wait till some emergency timer refresh kicks in.
- Synch to and from is a pain and error prone. Whilst code rarely gets clobbered you can get unexpected multiple declarations of variables, loss of associations in your UML diagrams when re-reverse engineering (esp if generated by design pattern workflow)
- Code generation concepts "Add Merge Markers to Existing Source Elements" and "Generate Markers for Source File Merging" too complex, IMO.
- Apply a design pattern twice and you get two generalisation relationships between classes and other silly bugs

## Final Thoughts

I found Netbeans to be generally very cool, fast and stable. The UML tool is generally nice to use. It's really a pity that the UML code generation is a bit buggy and that UML custom code generation is undocumented and uber complex. The Design Patterns support and customization in Netbeans is outstanding (not at the level of IBM Rational though). Again it's a pity there is no design patterns code generation support, and that this part of Sun's roadmap seems to be a dead end. They even say on their UML plugin home page that if you want more - use the Visual Paradigm plugin for Netbeans. I live in hope that this opensource tool can be improved though. Maybe I should start looking at the code base myself... :-)

[concrete5 - open source CMS](#) © 2020 [AndyPatterns](#). All rights reserved. [Sign In to Edit this Site](#)

### Become A Data Innovator

Flip The 80/20 Rule and Spend Most Of Your Time Uncovering Brilliant Insights.  
Alteryx