Java Reverse Engineering Overview

Overview

Java Reverse Engineering supports Java 1.2 as described in *The Java Language Specification* by James Gosling, Bill Joy and Guy Steele and continues the reverse engineering model used for C++.

- Java classes and interfaces generate class symbols in the class diagram.
- Java class members and methods are added as attributes and operations within class symbols.
- The GD*Pro* system which is created during reverse engineering contains 100% of the structural information present in the Java source code, except comments and white space.
- The implementation diagram contains a source file symbol for each Java source file.
- File dependencies are derived from Java "import" statements (similar to C++ #includes).
- Java Reverse Engineering builds systems for UML.

Overview - Packages

Java uses a code packaging system which has no equivalent in C++.

C++: New name spaces can be created explicitly at any time using the "namespace" construct, and implicit namespace exist for:

- Method definitions
- Class definitions
- The "global" scope of a file and all of the files it #includes

Java: Java has no equivalent to C++'s "namespace" construct, but implicit namespaces exist for:

- Method definitions (equivalent to C++)
- Class definitions (including interfaces) (equivalent to C++)
- Files (bound by the start and end of file)
- Packages (collections of files in a directory)

Java Reverse Engineering fully supports Java's implicit namespaces.

Overview - Interfaces

Interfaces in Java provide some of the features of multiple inheritance without the headaches of full MI. A class "extends" only a single superclass, but it can "implement" one or more interfaces. Interfaces themselves can "extend" other interfaces.

An interface is not a definition of a class - it's a definition of a set of methods that one or more classes will implement. An important issue of interfaces is that they declare only methods and constants. Variables can not be defined in interfaces.

The syntax of an interface is a subset of the syntax for a class.

During reverse engineering:

- Each interface declaration generates a class symbol.
- Interfaces which inherit from other interfaces will be linked to their super-interface's symbol using a generalization link.
- A class's implementation of an interface results in a generalization link between the class's symbol and the interface's symbol.

Overview - Exceptions/Access Control Modifier

Exceptions

Java exceptions work much like C++ exceptions. A method can declare a list of exceptions it may throw, e.g:

```
public static int c(int i) throws MyException, MyOtherException {...

};
```

Access Control

Java supports the following levels of access control:

"package visibility" - the default, with no equivalent in C++. Members are accessible to all objects within the same package.

public - equivalent to C++'s public

private - equivalent to C++'s private

protected - equivalent to C++'s protected

Java Reverse Engineering fully supports all levels of Java access control for classes, attributes, and operations.

Modifiers

These Java modifiers can be used, variously, on class, method, and variable declarations.

final The final keyword is a modifier that can be applied to classes, methods, and variables. It has a similar, but not identical meaning in each case.

A final class can never be subclassed.
A final method can never be overridden.
A final variable can never have its value set.

Modifiers:

operation: constant set to true

attribute: constant set to true

Access Permission is "read-only"

Example:

public final void writeDouble(double v) throws IOException {};

native The native keyword is a modifier that can be applied to method declarations. It indicates that the method is implemented elsewhere in C, or in some other platform-dependent fashion.

Modifiers:

native is not capture

will set "native" to true

Example:

private native void socketCreate(boolean isServer);

synchronized The synchronized keyword can be used as a modifier for class or instance methods. It indicates that the method modifies the internal state of the class or the internal state of an instance of the class in a way that is not thread-safe.

Modifier:

sets synchronized to true

private synchronized native String initializeLinkerInternal();

transient The transient keyword is a modifier that can be applied to instance fields in a class. It indicates a field that is not part of an object's persistent state and thus needs not be serialized with the object.

Modifier:

Sets transient to true.

Example:

transient private int pData;

volatile The volatile keyword is a modifier that can be applied to fields. It specifies that the field is used by synchronized threads and that the compiler should not attempt to perform optimizations with it.

Modifier:

Sets volatile to true.

Example:

volatile private int pData;

Overview - ClassPath and Import

When Java classes are referenced using the "import" construct, the Java interpreter seeks out these classes in the standard system classes and in classes referenced relative to the current directory.

Additionally, Java uses the CLASSPATH environment variable to locate Java source located in other directories. This is similar to C++'s use of "include paths" to locate files for expanding "#include <>" directives.

When the CLASSPATH environment variable is defined and a class is referenced using Java's "import" statement, a search for the class is launched:

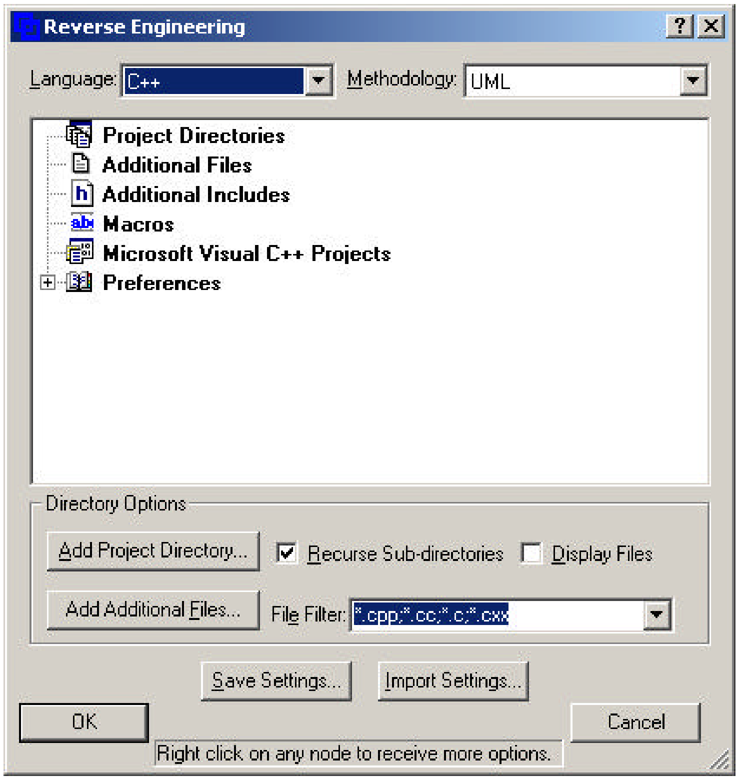
1. First search each directory listed in CLASSPATH.
2. Then search the standard Java classes.

In addition to the environment variable, Java users can specify a class path to the Java compiler using the -classpath command-line argument. Setting this option overrides any path specified in the CLASSPATH environment variable. If this option is used, the compiler does not append the location of the system classes to the end of this path.

Java Reverse Engineering fully implements Java's CLASSPATH and "import" behavior.

Starting the Reverse Engineering Process

1. Choose TOOLS->REVERSE ENGINEER from the menu. The Reverse Engineering dialog box is displayed.



Language: C++

Methodology: UML

☑ Recurse Sub-directories ☐ Display Files

File Filter: *.cpp;*.cc;*.c;*.cxx

Right click on any node to receive more options.

2. Make a choice from the "Language" drop down list. Your options are C++, Java or IDL. Select the language "Java". The default setting for the language is C++.

   The following information then appears in the Directory List Box:

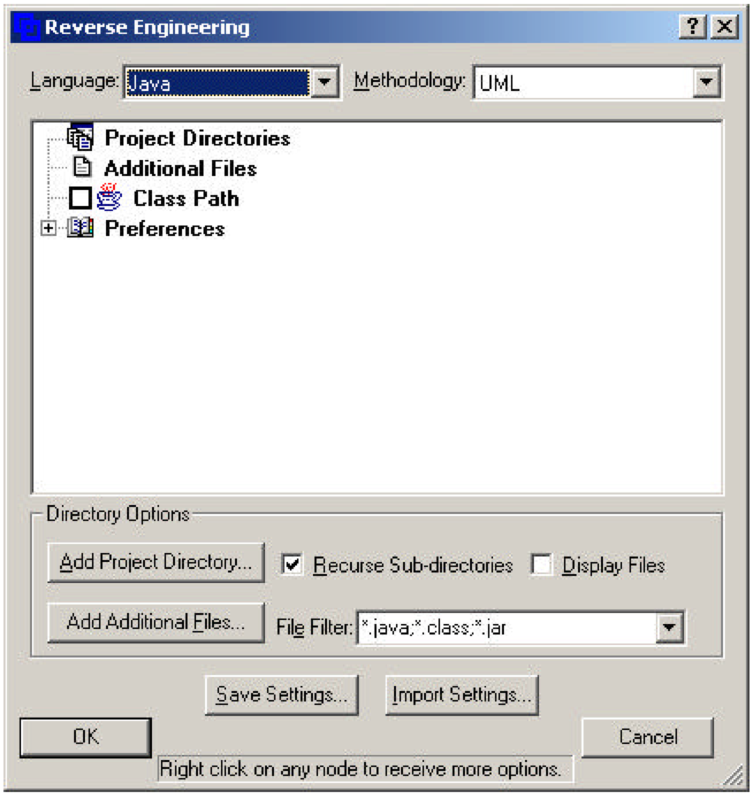
   Project Directories This shows the various directories that will be interrogated for files with extensions indicated in the File Filter text box. You can add directories by selecting the "Add Project Directory" button.

   Additional Files Right-click this option to add any files not found in the Project Directories tree that you would like to reverse engineer. You can also delete all additional files.

   Class Path This shows the various entries found in your CLASSPATH environment variable.

   Preferences The following preferences can be set:
   - Hide Class Attributes
   - Hide Class Operations
   - Enable Comment Capture

3. Select UML from the "Methodology" drop down list. Currently UML is your only choice.



Reverse Engineering

Language: Java Methodology: UML

- Project Directories
- Additional Files
- □ Class Path
- ⊞ Preferences

Directory Options

☑ Recurse Sub-directories □ Display Files

File Filter: *.java;*.class;*.jar

Right click on any node to receive more options.

4. Check the Recurse Subdirectory option if you want added directories (both Project and Additional Includes directories) to recurse subdirectories when creating new nodes. If a recursion is done, only directories with files matching the filter are added.

5. Check the Display Files option if you want all the files displayed in the Project Directories tree. A ⊞ appears next to all the project directories. Click this icon to expand the tree and display all the files in the directory.

Additional Files

You can add project directories containing files to be reverse engineered. However, for purposes of this tutorial we are only going to reverse engineer three files, not the entire project directory. So to do this we are going to use the Additional Files option in the Reverse Engineering dialog box.

You can select one or multiple files for inclusion from a file selection dialog box. The files selected are added whether or not they match the file filter.

1. Right-click on the Additional Files title in the files list box and a background menu opens.

2. Select the Additional Files command from the menu and the Open dialog box appears. Notice that the types of files listed in the Files of Type text box correspond to the Language selection you made in the Reverse Engineering dialog box.

Note: You can also click the [Add Additional Files...] button.

3. Select the following path:

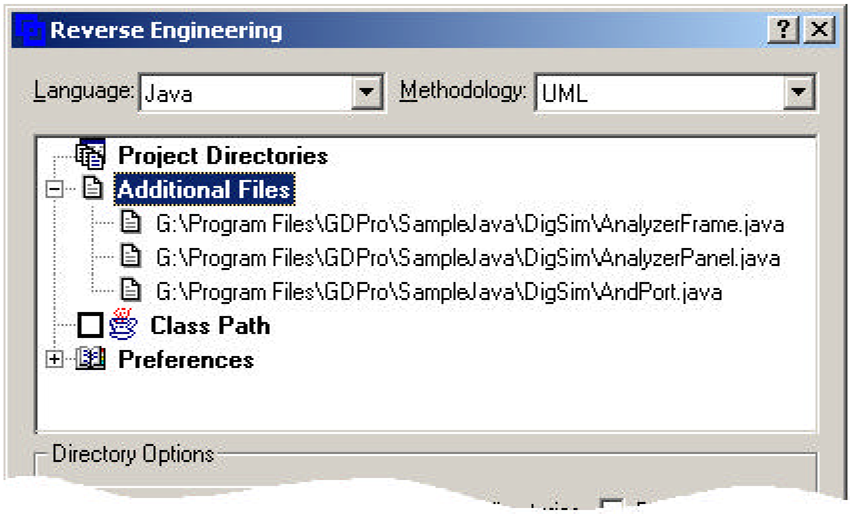
   x:/Program Files/GDPro/SampleJava/DigSim

4. For this tutorial select the following files:

AnalyzerFrame.java
AnalyzerPanel.java
AndPort.java

5. After you have selected the files you want, click Open. The dialog box closes and the files you selected appear in the list box of the Reverse Engineering dialog box.

Note: Use the Control key to select multiple, non-adjacent files from the file list. The Shift key allows you to select a group of adjacent files in the list.



Language: Java
Methodology: UML

G:\Program Files\GDPro\SampleJava\DigSim\AnalyzerFrame.java
G:\Program Files\GDPro\SampleJava\DigSim\AnalyzerPanel.java
G:\Program Files\GDPro\SampleJava\DigSim\AndPort.java

Drag and Drop - Additional Files

You can also "drag and drop" files directly from the Windows Explorer into the Additional Files tree.

Note: This option does not respect the file filter set.

Additional Commands - Additional Files Background Menu

When you right-click on the Additional Files title in the Reverse Engineering dialog box, the following commands appear on the background menu.

Add Additional Files This command allows you add individual files not found in the Project Directory.

Remove All Additional Files This command allows you to remove all the additional files in the Additional Files Directory.

Right-click on the Additional Files title in the Reverse Engineering dialog box and choose the Remove All Additional Files command. A prompt dialog box opens asking if you want to remove all the additional files.

Remove Additional Files

The Remove Additional Files command is active if you have placed individual files in the Additional Files directory.

This command allows you to remove individual files from the Additional Files directory.

Right-click on a selected file and choose the Remove Additional Files command. A prompt dialog box opens asking if you want to remove the selected file.

Reference Classes

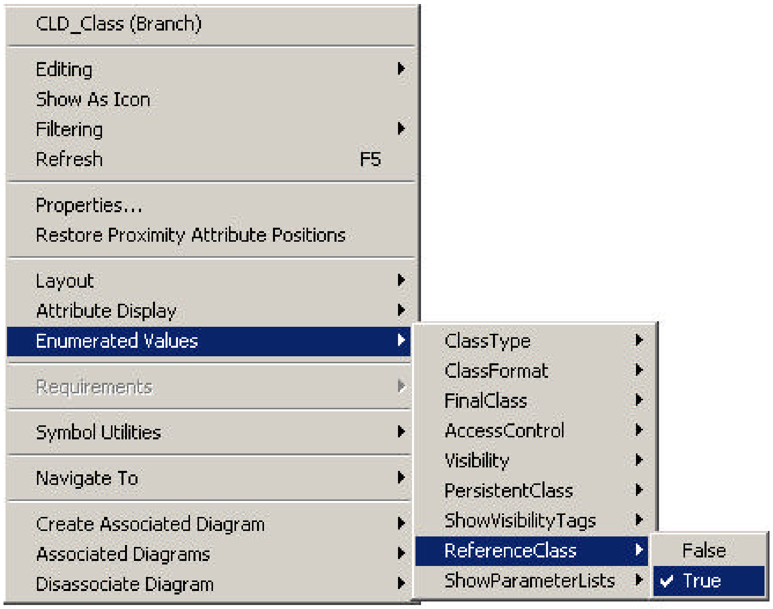
Note: The following section on Reference Classes is for information only and not part of the tutorial.

Reference classes are classes that do not participate in the code generation process. They are automatically created by reverse engineering when classes in the reverse engineering set of files have associations or inheritance relationships with classes that are not defined in the code that is being reverse engineered. Reference classes complete the relationships, but do not have any content (no attributes or operations defined), and will not have any code generated for them.

Once you have your library of reference classes, you can use them as components in creating a new system by opening the system with the reference classes, going to the navigator, clicking on the class you want to use in your new system, zoom to the class, and copy and paste it into your new class diagram. You can inherit or create associations to the reference class, and GD*Pro* knows not to generate the header and body files for the reference class.

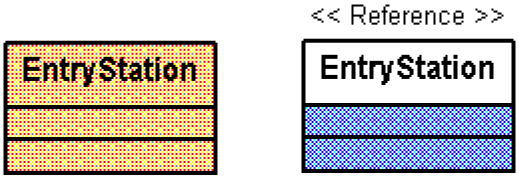
Setting Reference Classes

1. Open the class diagram in the view area.
2. Select the class you want to make a reference class.
3. Right-click on the class symbol. A background utilities menu opens.



4. Select ENUMERATED VALUES->REFERENCE CLASS->TRUE.

5. The selected class is now a reference class. You can also toggle the reference classes back to "live" classes by selecting ENUMERATED VALUES->REFERENCE CLASS->FALSE.



Reference Include Libraries

Reference include directories are available for reverse engineering. They allow reverse engineering to "locate source files" from reference class libraries, like Java Foundation Class Library, without fully reverse engineering the classes in the reference library. You can specify as many reference include directories as you wish to make sure that reverse engineering locates source files properly.

Note: This section on the Reference Include Libraries is for information only. Currently the sample code for the tutorial does not include any reference libraries.

Reference Include directories are available for reverse engineering. They allow reverse engineering to process macro definitions from reference class libraries, like Rogue Wave or the Microsoft Foundation Class Library, without fully reverse engineering the classes in the reference library. You can specify as many reference include directories as you wish, to make sure that reverse engineering is able to locate all header files.

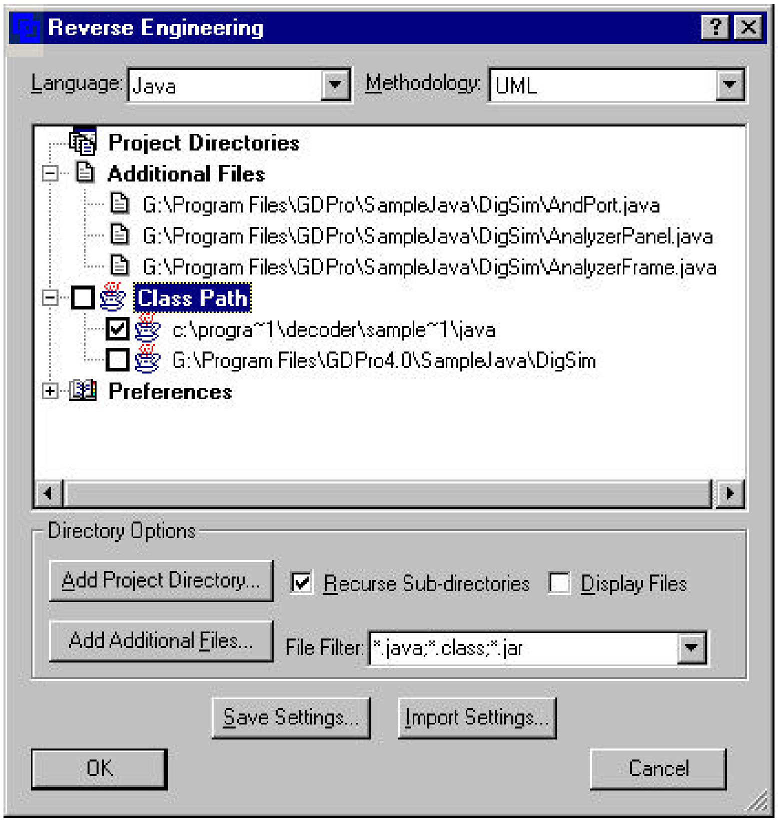
Reference and Normal Include Directories

- Reverse Engineering uses a pre-processing phase to process include dependencies
- The pre-processor searches the Include Directories for source files included by the files being reverse engineered

- Included source files that are found in "normal" include directories are also reverse engineered, and the class structure is added to the model
- Included source files that are found in "reference" include directories are not reverse engineered, and their class structure is not added to the model

To add a reference include directory to the reverse engineering process:

1. Add an include directory to the Class Path portion of the tree. When this directory is added, the option box to the left of the directory path is checked.



Reverse Engineering

Language: Java Methodology: UML

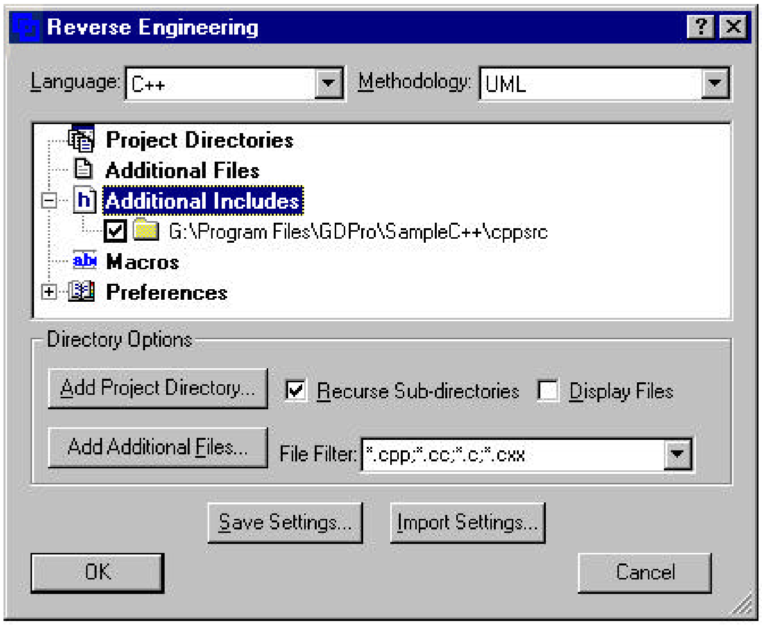
```
Project Directories
Additional Files
    G:\Program Files\GDPro\SampleJava\DigSim\AndPort.java
    G:\Program Files\GDPro\SampleJava\DigSim\AnalyzerPanel.java
    G:\Program Files\GDPro\SampleJava\DigSim\AnalyzerFrame.java
□ Class Path
    ☑ c:\progra~1\decoder\sample~1\java
    □ G:\Program Files\GDPro4.0\SampleJava\DigSim
Preferences
```

Directory Options

☑ Recurse Sub-directories □ Display Files

File Filter: *.java;*.class;*.jar

2. To have this directory included as a reference library only, remove the checkmark from the option box to the left of the directory path.



Language: C++
Methodology: UML

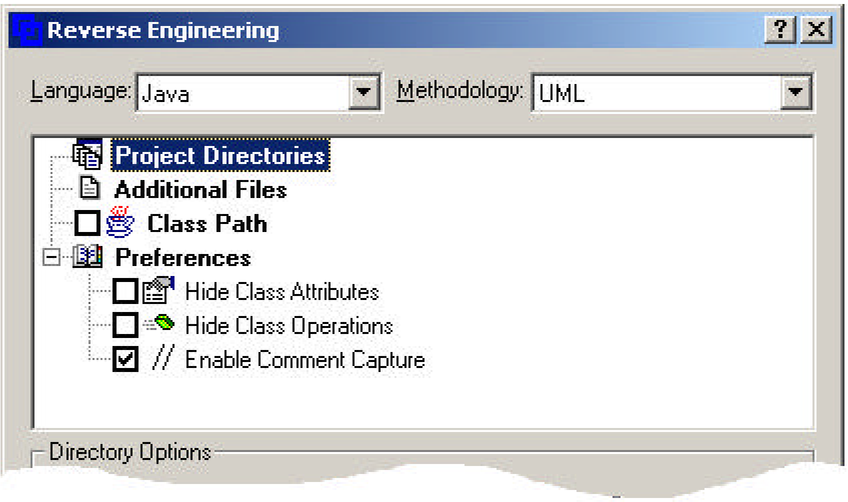
- Project Directories
- Additional Files
- Additional Includes
  - ☑ G:\Program Files\GDPro\SampleC++\cppsrc
- Macros
- Preferences

☑ Recurse Sub-directories
□ Display Files
File Filter: *.cpp;*.cc;*.c;*.cxx

The reverse engineering process identifies these directories as referenced.

Setting Preferences

1. Click the ⊞ located to the left of Preferences title. The list expands showing the available preferences.



Language: Java
Methodology: UML

- Project Directories
- Additional Files
- □ Class Path
- Preferences
  - □ Hide Class Attributes
  - □ Hide Class Operations
  - ☑ // Enable Comment Capture

Hide Class Attributes Hides the class attributes. These class attributes are captured and can be accessed through the reporting function.

Hide Class Operations Hides the class operations. These class operations are captured and can be accessed through the reporting function.

Enable Comment Capture The reverse engineering process will capture comments from the source code and add them to the model as class, attribute and operation descriptions. The forward engineering process applies the current descriptions from the model to the source when doing code generation.

2. The "Hide Class Attributes", and "Hide Class Operations" preferences should not be selected.
3. Make sure the "Enable Comment Capture" is selected.

   The preferences you set are implemented when you start the reverse engineering process.

Note: When reverse engineering large diagrams, you should hide the class attributes and operations because of the size of the diagram generated.

Save Settings

Once you have set your preferences, includes, reference include libraries, and definition, you can save these settings for future use.

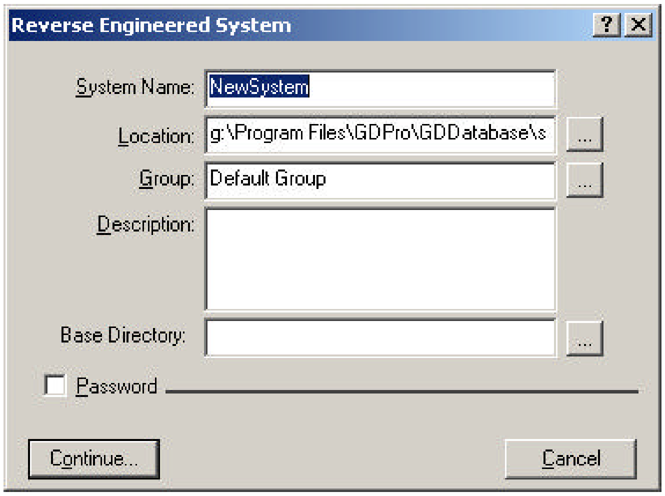
1. Click the "Save Settings" button. The Save As dialog box opens.
2. Select the directory and then type the name "Tutorial_Java" in the file name text box. The file extension is ".re" and it is the default setting in the Save as Type text box.
3. Click Save, the Save As dialog box closes and a prompt dialog box opens telling you that your file was saved successfully.
4. Click OK to close the prompt dialog box.

Import Settings

You can also use a previously created file by clicking the Import Settings button. The Open dialog box appears with all the available settings files shown in the list box. Select the file you want to use and click Open. The dialog box closes and the imported settings are in effect.

Naming the New System

1. Click at the bottom left of the Reverse Engineering dialog box. The Reverse Engineered System dialog box opens.



System Name: NewSystem

Location: g:\Program Files\GDPro\GDDatabase\s

Group: Default Group

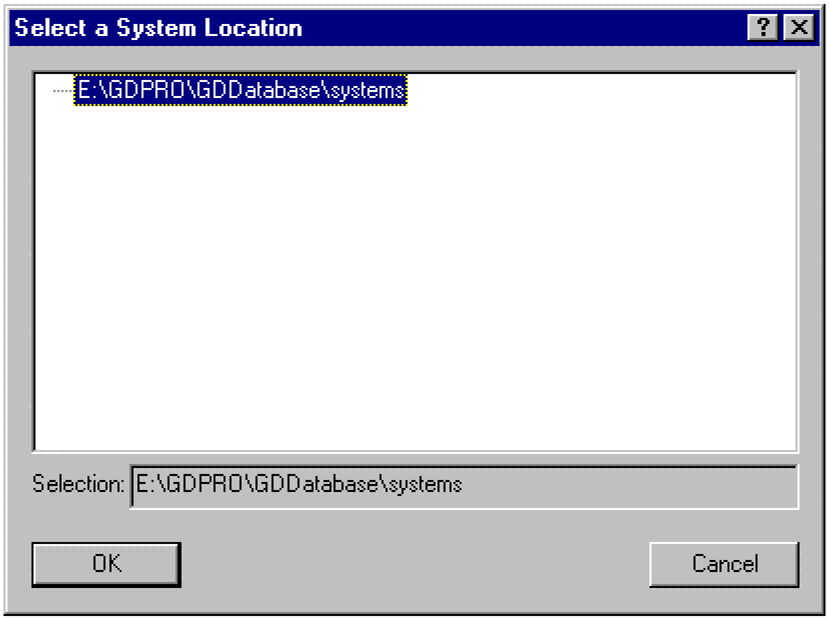
Description:

Base Directory:

□ Password

2. Type the name "RE Tutorial" for your new system. The default name is NewSystem.

3. The default location for your system is GDPro\GDDatabase\systems. If you want to store your system in another location click ... next to the Location text box. The Select a System Location dialog box opens and all available system directories

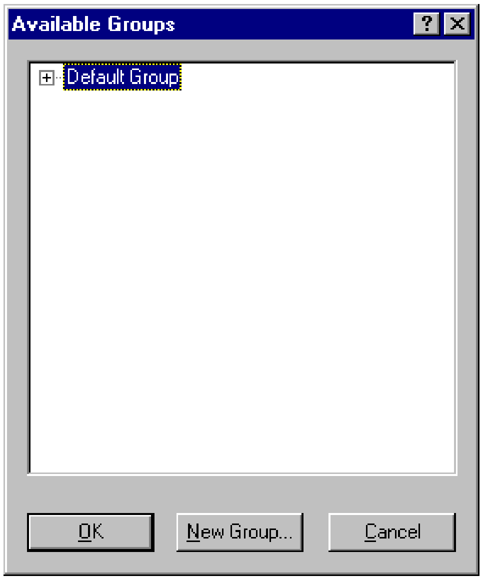


E:\GDPRO\GDDatabase\systems

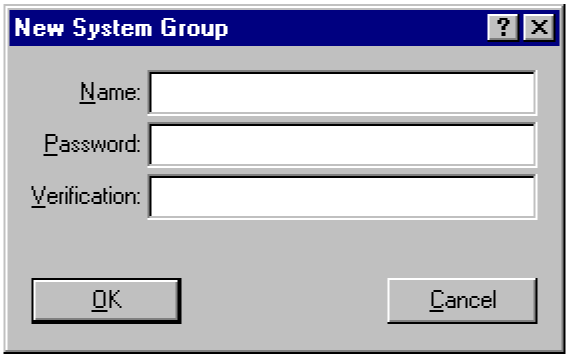
Selection: E:\GDPRO\GDDatabase\systems

4. For this tutorial we will not change the system directory. Click OK and the Select a System Location dialog box closes.

5. The reverse engineering system is automatically placed in a group named Default Group. We will put the system in another group so click ... located next to the Group text box. The Available Groups dialog box opens.



6. Create a new group for your reverse engineered system, by clicking New Group... . The New System Group dialog box opens.

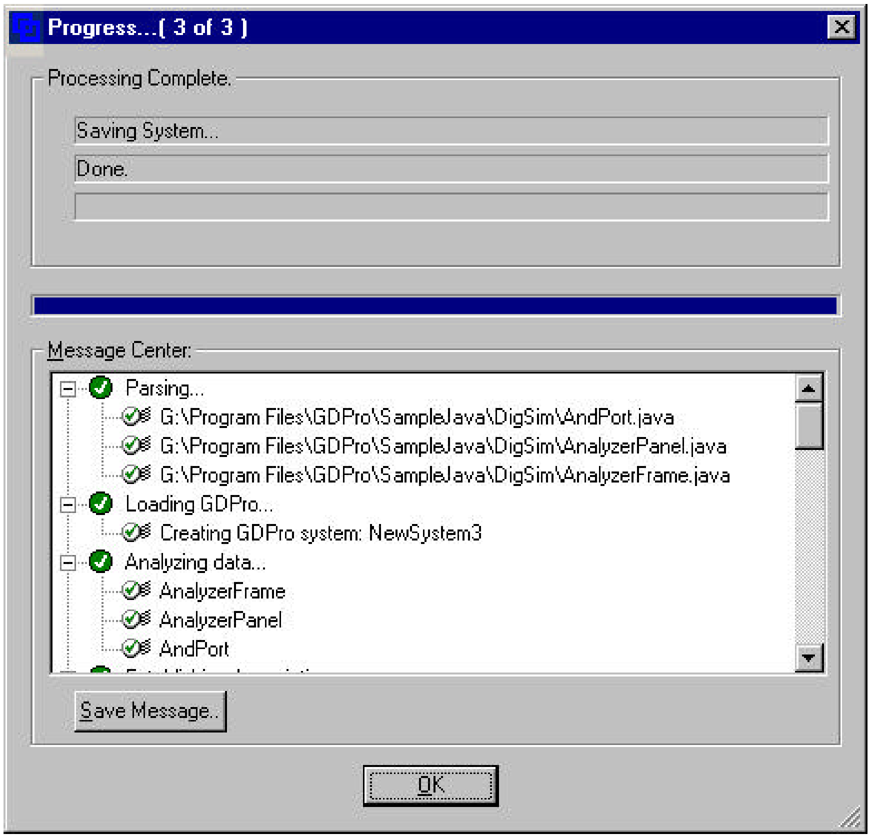


7. Type the name "Java_RE_Test" in the Name text box.
8. Type the password "tutorial" in the Password text box. Reenter the same password in the Verification text box and click OK. The New System Group dialog box closes.
9. Select the group name "Java_RE_Test" from the Available Groups list box and click OK. The Available Groups dialog box closes and the name Java_RE_Test appears in the Group text box.
10. In the Reverse Engineered System dialog box type the following description: "This is an example of Reverse Engineering". This field is optional.
11. Enter a base directory for this system. If you click the browse button, a Search for Folder dialog box opens. You can select your base directory from this dialog box.

Reverse Engineering Progress

1. After you have completed all the requested information click Continue... . The Progress dialog box opens. This displays messages of the reverse-engineering process.

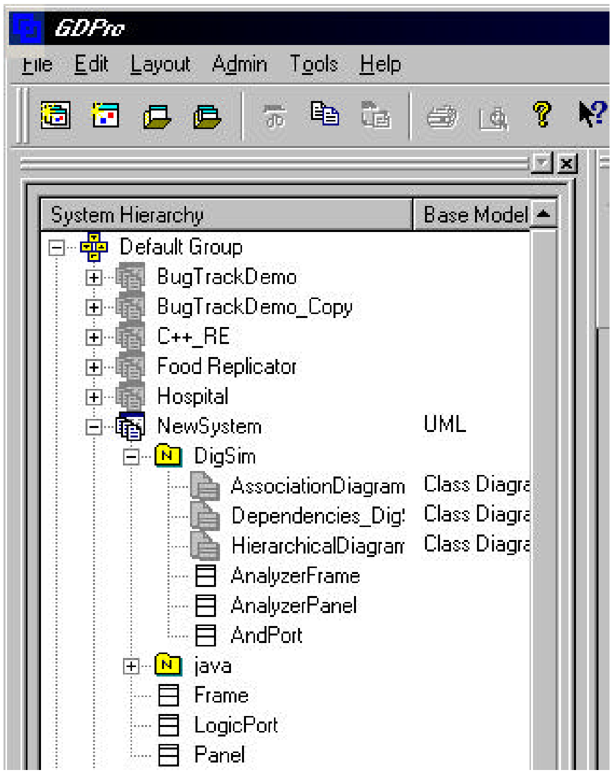
Note: The following files are included in the reverse engineering process: (1) files in the selected system directory; and (2) files located using Java import statements and the CLASSPATH or GDPro Include Path statements.



```
Saving System...
Done.
```

```
Parsing...
    G:\Program Files\GDPro\SampleJava\DigSim\AndPort.java
    G:\Program Files\GDPro\SampleJava\DigSim\AnalyzerPanel.java
    G:\Program Files\GDPro\SampleJava\DigSim\AnalyzerFrame.java
Loading GDPro...
    Creating GDPro system: NewSystem3
Analyzing data...
    AnalyzerFrame
    AnalyzerPanel
    AndPort
```

2. You can save the messages/error file by clicking Save Message... The Save As dialog box opens.
3. Name the file "System Messages" and click OK. The Save As dialog box closes and the message file is saved in the .txt format. The default file name is GDResults.txt.
4. Click OK to close to Progress dialog box. The generated diagram names appear in the System Hierarchy Window under the group and system name.



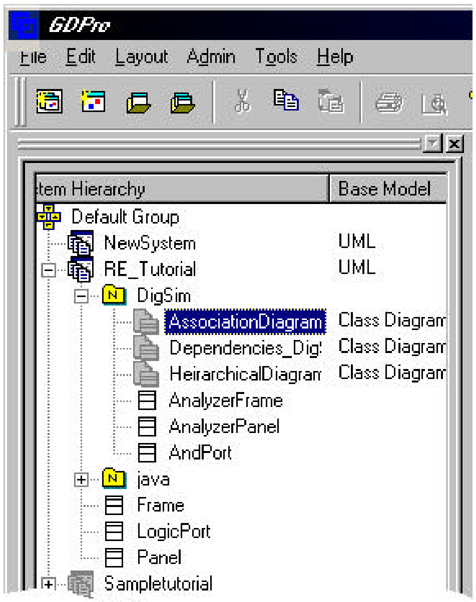
System Hierarchy	Base Model
Default Group	
BugTrackDemo	
BugTrackDemo_Copy	
C++_RE	
Food Replicator	
Hospital	
NewSystem	UML
DigSim	
AssociationDiagram	Class Diagra
Dependencies_Dig!	Class Diagra
HierarchicalDiagram	Class Diagra
AnalyzerFrame	
AnalyzerPanel	
AndPort	
java	
Frame	
LogicPort	
Panel	

When the reverse engineering is completed, you should have three UML Object models that describe the Java source code you reverse engineered: the Association Class Diagram; the Dependencies Class Diagram; and the Hierarchy Class Diagram

Association Class Diagram

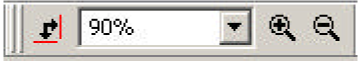
Let's take a closer look at some of the objects on your diagrams. The Association Class diagram organizes the classes and interfaces by focusing in on the associations found between the various types found during reverse engineering.

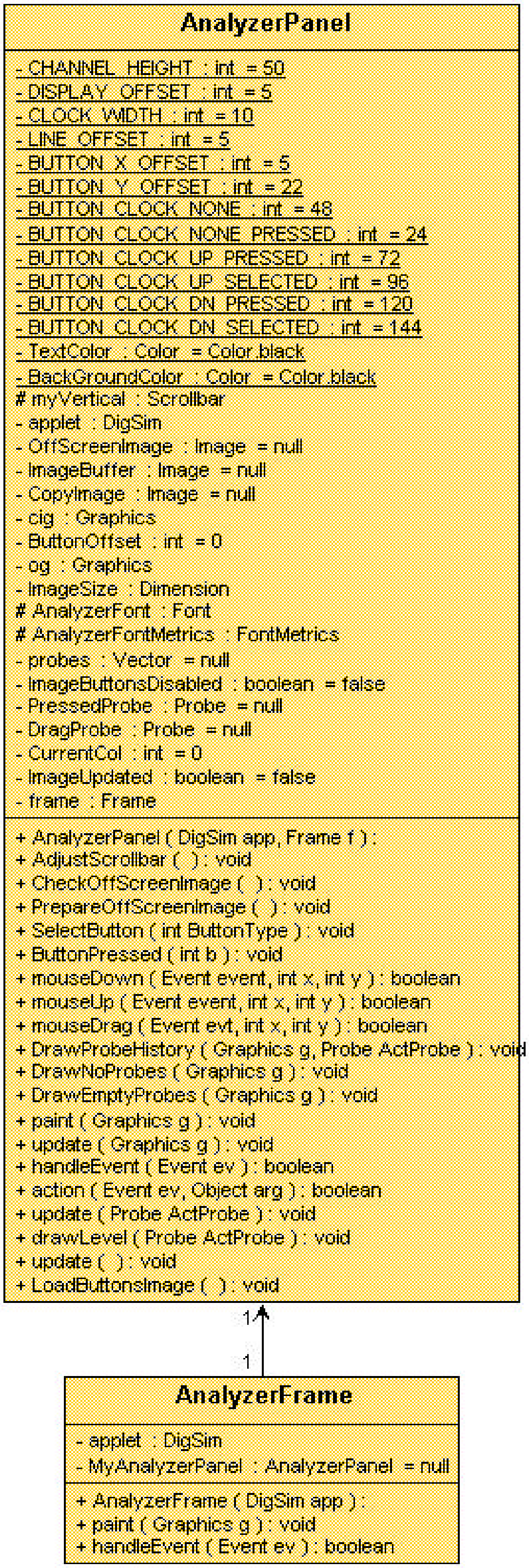
1. When the reverse engineering process is complete the created diagrams appear in the System Hierarchy window.



tem Hierarchy	Base Model
Default Group	
NewSystem	UML
RE_Tutorial	UML
DigSim	
AssociationDiagram	Class Diagram
Dependencies_Dig!	Class Diagram
HeirarchicalDiagram	Class Diagram
AnalyzerFrame	
AnalyzerPanel	
AndPort	
java	
Frame	
LogicPort	
Panel	
Sampletutorial	

2. Double-click on the Association Diagram name in the System Hierarchy window and the associated diagram opens.

Note: Since most designs are large, you can zoom out on a design to better view the entire design. Click the drop down arrow on the menu . A drop menu opens listing available zoom percentages. Select the percentage you want to reduce your design. Or you can also select the Zoom to Fit option from the list. You can then select 100% to return to actual size.


AnalyzerPanel
- CHANNEL_HEIGHT : int = 50
- DISPLAY_OFFSET : int = 5
- CLOCK_WIDTH : int = 10
- LINE_OFFSET : int = 5
- BUTTON_X_OFFSET : int = 5
- BUTTON_Y_OFFSET : int = 22
- BUTTON_CLOCK_NONE : int = 48
- BUTTON_CLOCK_NONE_PRESSED : int = 24
- BUTTON_CLOCK_UP_PRESSED : int = 72
- BUTTON_CLOCK_UP_SELECTED : int = 96
- BUTTON_CLOCK_DN_PRESSED : int = 120
- BUTTON_CLOCK_DN_SELECTED : int = 144
- TextColor : Color = Color.black
- BackGroundColor : Color = Color.black
# myVertical : Scrollbar
- applet : DigSim
- OffScreenImage : Image = null
- ImageBuffer : Image = null
- CopyImage : Image = null
- cig : Graphics
- ButtonOffset : int = 0
- og : Graphics
- ImageSize : Dimension
# AnalyzerFont : Font
# AnalyzerFontMetrics : FontMetrics
- probes : Vector = null
- ImageButtonsDisabled : boolean = false
- PressedProbe : Probe = null
- DragProbe : Probe = null
- CurrentCol : int = 0
- ImageUpdated : boolean = false
- frame : Frame
+ AnalyzerPanel (DigSim app, Frame f) :
+ AdjustScrollbar ( ) : void
+ CheckOffScreenImage ( ) : void
+ PrepareOffScreenImage ( ) : void
+ SelectButton (int ButtonType) : void
+ ButtonPressed (int b) : void
+ mouseDown (Event event, int x, int y) : boolean
+ mouseUp (Event event, int x, int y) : boolean
+ mouseDrag (Event evt, int x, int y) : boolean
+ DrawProbeHistory (Graphics g, Probe ActProbe) : void
+ DrawNoProbes (Graphics g) : void
+ DrawEmptyProbes (Graphics g) : void
+ paint (Graphics g) : void
+ update (Graphics g) : void
+ handleEvent (Event ev) : boolean
+ action (Event ev, Object arg) : boolean
+ update (Probe ActProbe) : void
+ drawLevel (Probe ActProbe) : void
+ update ( ) : void
+ LoadButtonsImage ( ) : void
1
1
AnalyzerFrame
- applet : DigSim
- MyAnalyzerPanel : AnalyzerPanel = null
+ AnalyzerFrame (DigSim app) :
+ paint (Graphics g) : void
+ handleEvent (Event ev) : boolean

There are a variety of class types in Java:

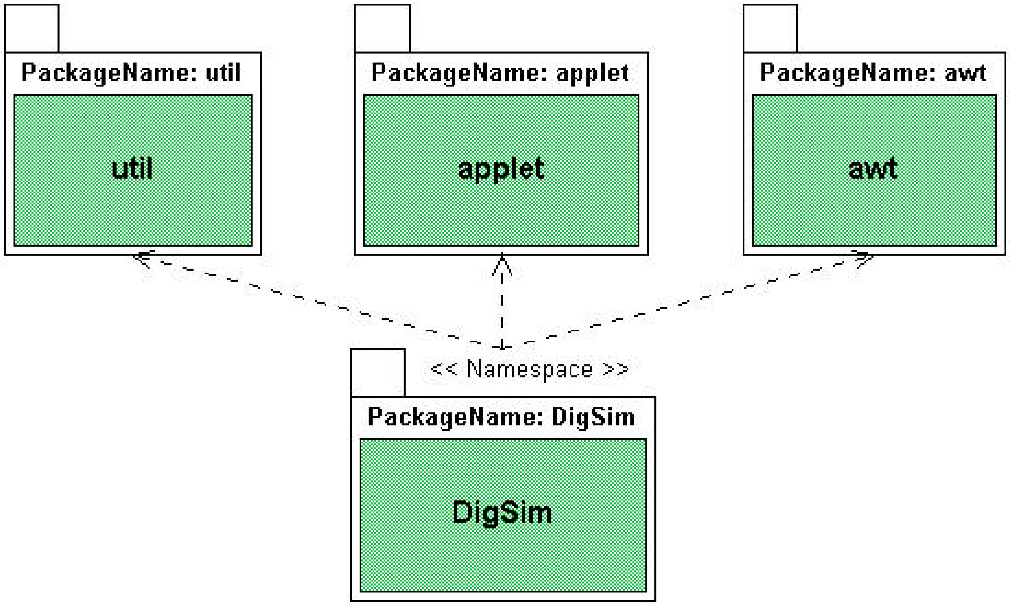
1. Package member class or interface
2. Nested top-level class or interface
3. Member class
4. Local class
5. Anonymous class

During Java Reverse Engineering, class symbols are created only for classes of the first four types. Definitions for anonymous classes are stored as "data" elements.

Dependencies Class Diagram

This diagram shows the packages that the current package is immediately dependant on.

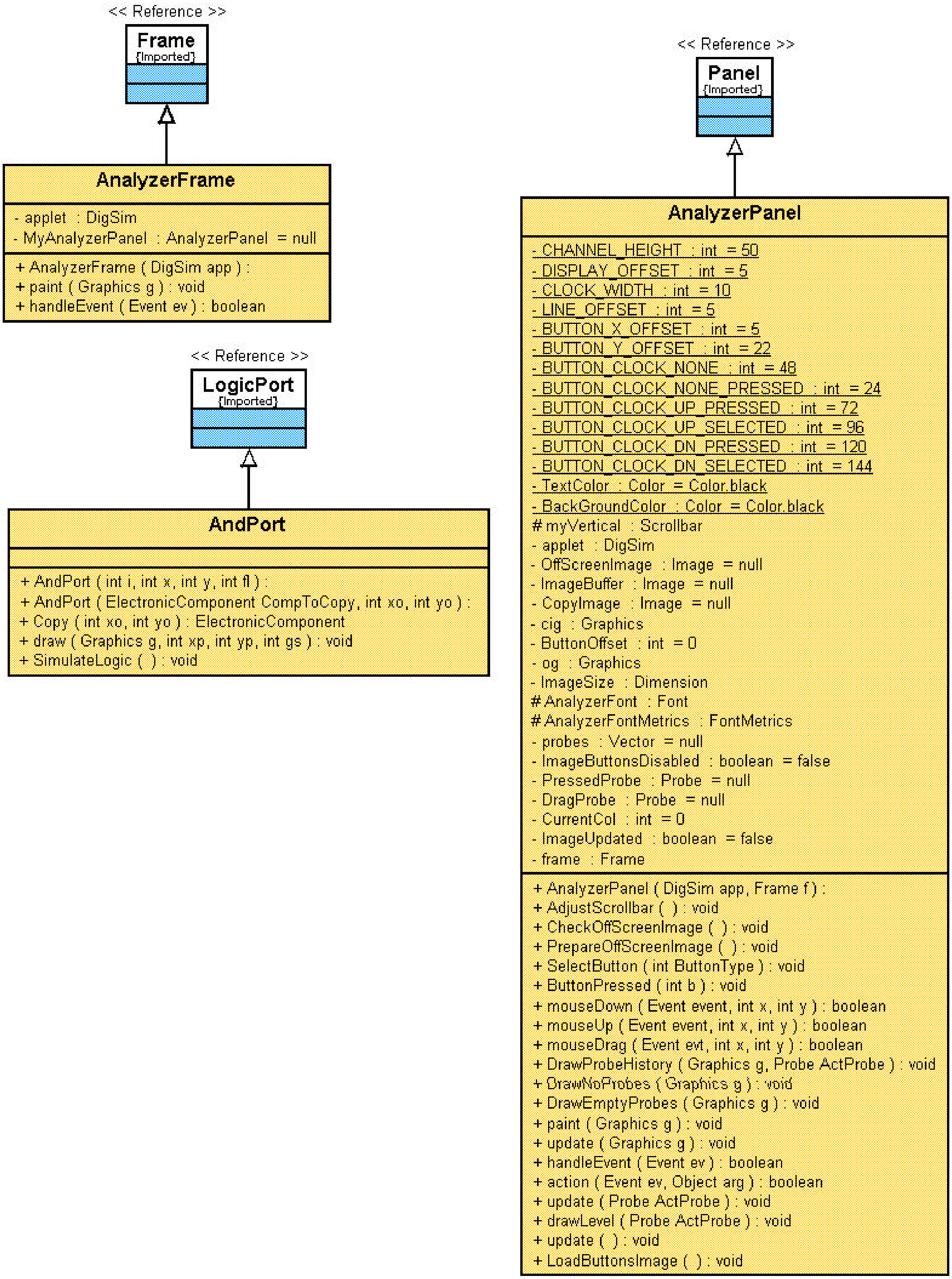
1. Double-click on the Dependencies Diagram name called "Dependencies_DigSim" in the System Hierarchy window and the associated diagram opens.



Hierarchy Class Diagram

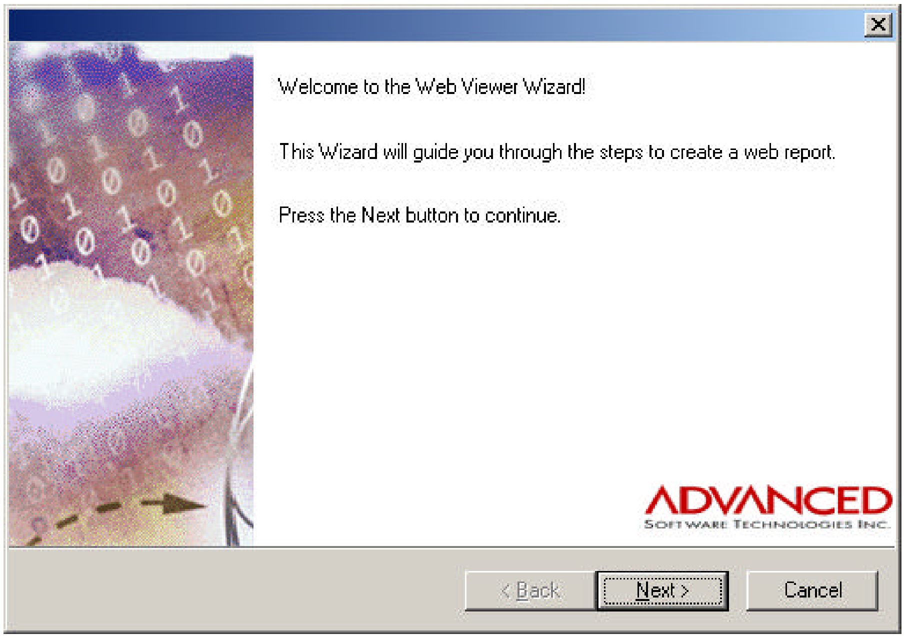
This diagram organizes the classes and interfaces found during processing based on inheritance relationships. This diagram will be created if any classes in the particular package are found to be in an inheritance or generalization relationship with another class.

1. Double-click on the Hierarchy Diagram name in the System Hierarchy window and the diagram opens.


<< Reference >>
Frame
{Imported}
AnalyzerFrame
- applet : DigSim
- MyAnalyzerPanel : AnalyzerPanel = null
+ AnalyzerFrame (DigSim app) :
+ paint (Graphics g) : void
+ handleEvent (Event ev) : boolean
<< Reference >>
LogicPort
{Imported}
AndPort
+ AndPort (int i, int x, int y, int fl) :
+ AndPort (ElectronicComponent CompToCopy, int xo, int yo) :
+ Copy (int xo, int yo) : ElectronicComponent
+ draw (Graphics g, int xp, int yp, int gs) : void
+ SimulateLogic ( ) : void
<< Reference >>
Panel
{Imported}
AnalyzerPanel
- CHANNEL_HEIGHT : int = 50
- DISPLAY_OFFSET : int = 5
- CLOCK_WIDTH : int = 10
- LINE_OFFSET : int = 5
- BUTTON_X_OFFSET : int = 5
- BUTTON_Y_OFFSET : int = 22
- BUTTON_CLOCK_NONE : int = 48
- BUTTON_CLOCK_NONE_PRESSED : int = 24
- BUTTON_CLOCK_UP_PRESSED : int = 72
- BUTTON_CLOCK_UP_SELECTED : int = 96
- BUTTON_CLOCK_DN_PRESSED : int = 120
- BUTTON_CLOCK_DN_SELECTED : int = 144
- TextColor : Color = Color.black
- BackGroundColor : Color = Color.black
# myVertical : Scrollbar
- applet : DigSim
- OffScreenImage : Image = null
- ImageBuffer : Image = null
- CopyImage : Image = null
- cig : Graphics
- ButtonOffset : int = 0
- og : Graphics
- ImageSize : Dimension
# AnalyzerFont : Font
# AnalyzerFontMetrics : FontMetrics
- probes : Vector = null
- ImageButtonsDisabled : boolean = false
- PressedProbe : Probe = null
- DragProbe : Probe = null
- CurrentCol : int = 0
- ImageUpdated : boolean = false
- frame : Frame
+ AnalyzerPanel (DigSim app, Frame f) :
+ AdjustScrollbar ( ) : void
+ CheckOffScreenImage ( ) : void
+ PrepareOffScreenImage ( ) : void
+ SelectButton (int ButtonType) : void
+ ButtonPressed (int b) : void
+ mouseDown (Event event, int x, int y) : boolean
+ mouseUp (Event event, int x, int y) : boolean
+ mouseDrag (Event evt, int x, int y) : boolean
+ DrawProbeHistory (Graphics g, Probe ActProbe) : void
+ DrawNoProbes (Graphics g) : void
+ DrawEmptyProbes (Graphics g) : void
+ paint (Graphics g) : void
+ update (Graphics g) : void
+ handleEvent (Event ev) : boolean
+ action (Event ev, Object arg) : boolean
+ update (Probe ActProbe) : void
+ drawLevel (Probe ActProbe) : void
+ update ( ) : void
+ LoadButtonsImage ( ) : void

Generating a Web Viewer Report

1. Choose TOOLS->REPORTS->WEB VIEWER to open the wizard.

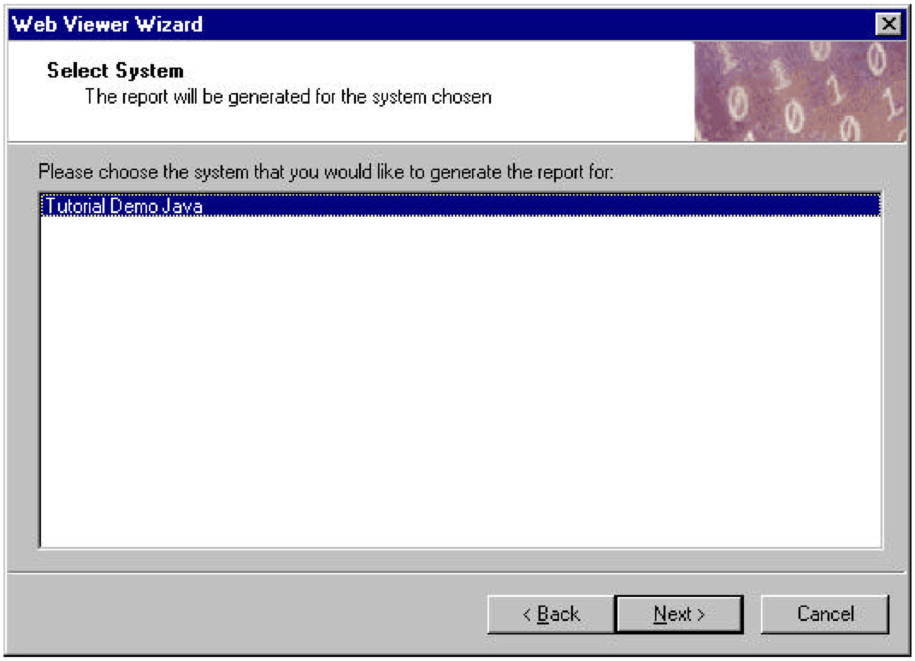


Welcome to the Web Viewer Wizard!

This Wizard will guide you through the steps to create a web report.

Press the Next button to continue.

2. Click the Next button and the Select System screen opens.



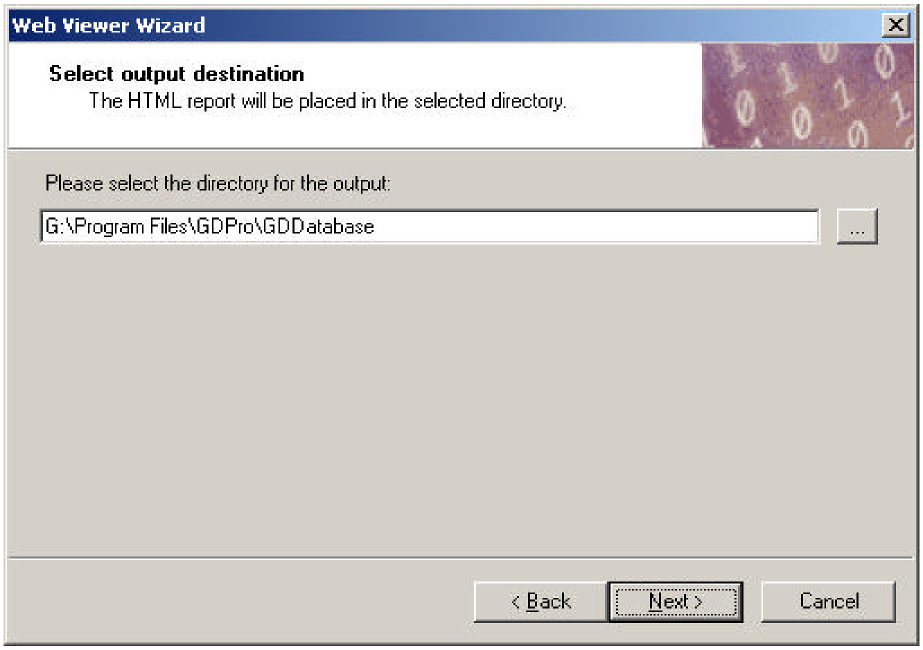
Select System

The report will be generated for the system chosen

Please choose the system that you would like to generate the report for:

Tutorial Demo Java

3. Select the system "Tutorial Demo Java" from the list box and click the Next button. The third wizard window opens.



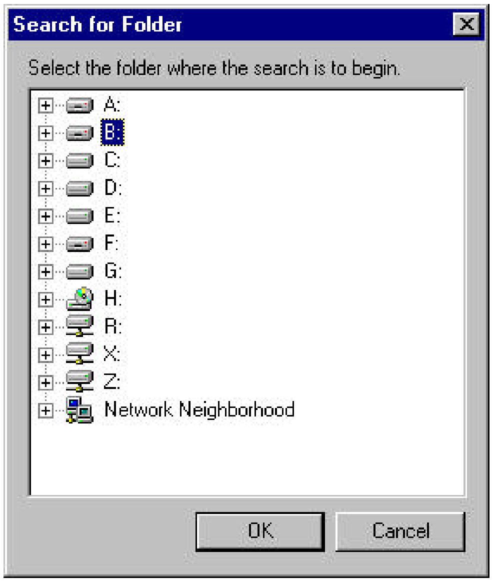
Web Viewer Wizard

Select output destination
The HTML report will be placed in the selected directory.

Please select the directory for the output:

G:\Program Files\GDPro\GDDatabase

3. Click the ... located to the right of the text box and the Search for Folder dialog box opens.

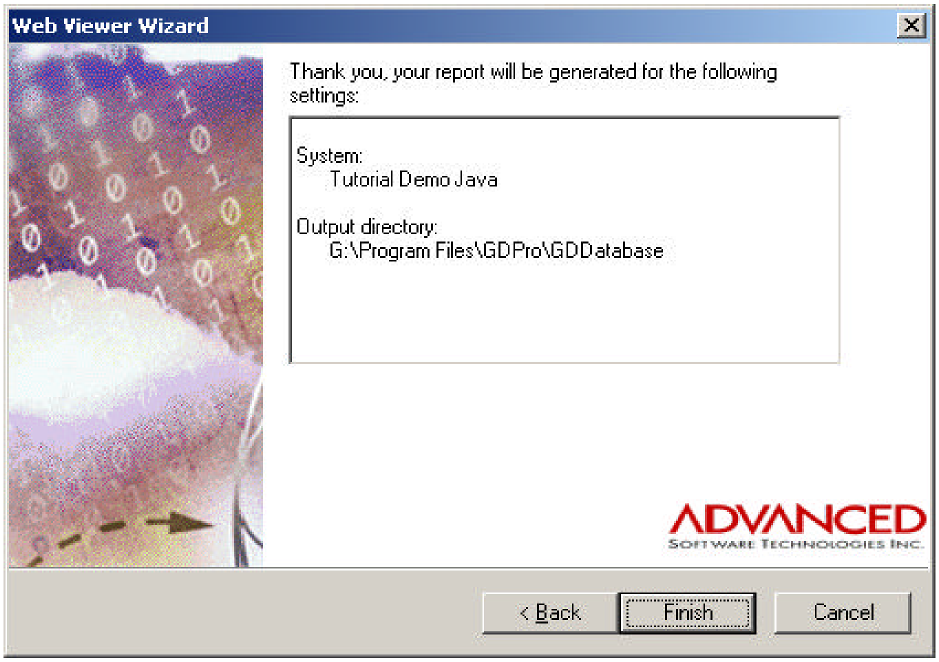


Search for Folder

Select the folder where the search is to begin.

4. Select the directory *x:/path/GDPro/GDDatabase* for the generated diagram. Click OK, the dialog box closes and the selected directory appears in the path text box.

5. Click the Next button an information dialog box opens showing your selection.



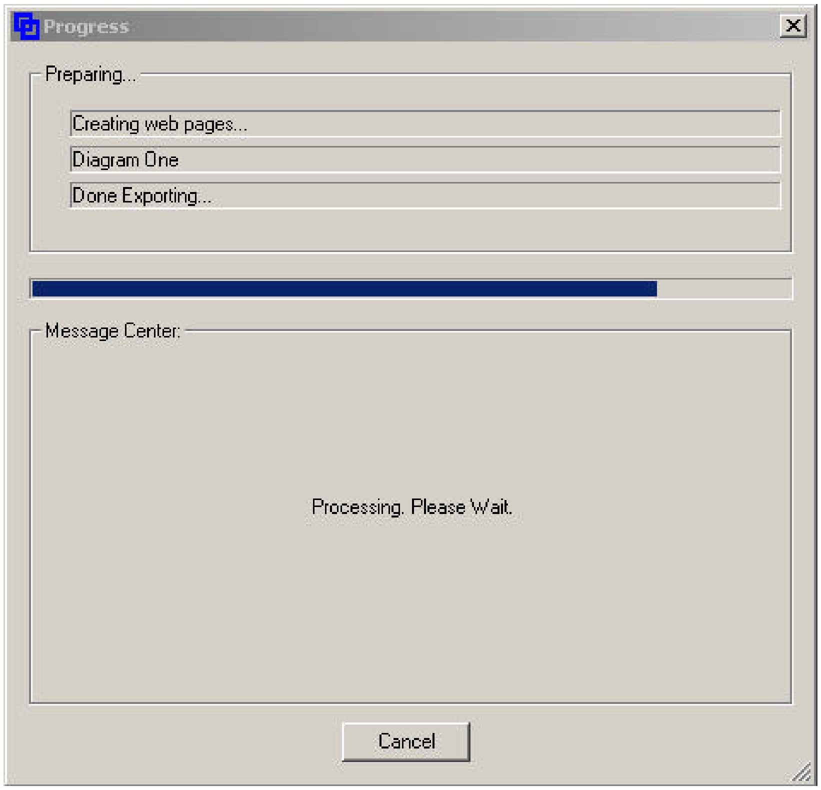
Web Viewer Wizard

Thank you, your report will be generated for the following settings:

System:
Tutorial Demo Java

Output directory:
G:\Program Files\GDPro\GDDatabase

5. Click Finish and a Progress dialog box opens, tracking the progress of the view generation.



Progress

Preparing...

Creating web pages...

Diagram One

Done Exporting...

Message Center:

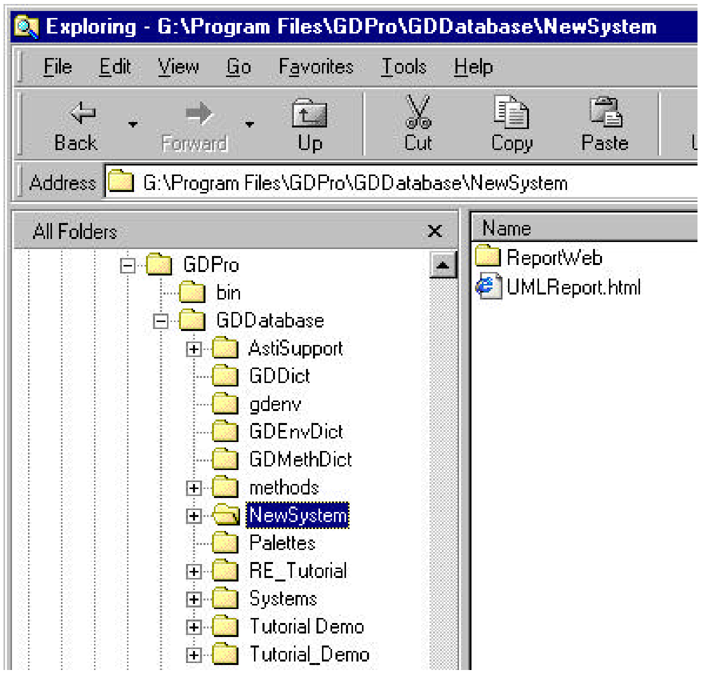
Processing. Please Wait.

As the HTML reports are created, the associated diagrams are opened and then closed in the view area of GDPro. Any diagrams opened when you start the process remain open after the report is created.

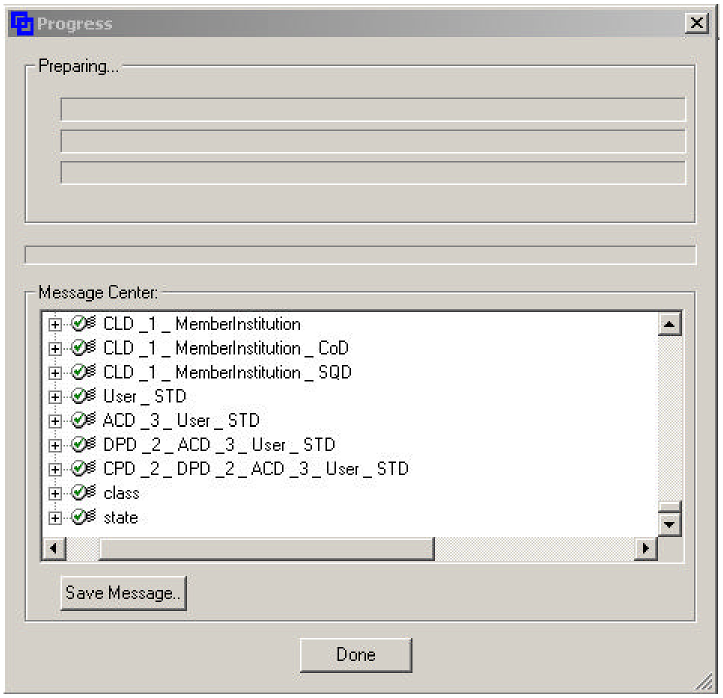
Process of Creating a Web Viewer Report

1. During the creation process a directory is created under the output directory you selected. The directory is the same as the system name. In the following graphic, you can see that the name of the system was Tutorial_Demo Java. Once the report generation is complete, the HTML file is placed in the directory as UMLReport.html. A ReportWeb directory is also created.

   The ReportWeb directory stores the web page structure. Each symbol is numbered and represented by a directory. The symbols are numbered in case two symbols have the same name. The number prevents duplicate files or overwriting of files.



2. The Progress dialog box also remains open.



CLD_1_MemberInstitution
CLD_1_MemberInstitution_CoD
CLD_1_MemberInstitution_SQD
User_STD
ACD_3_User_STD
DPD_2_ACD_3_User_STD
CPD_2_DPD_2_ACD_3_User_STD
class
state

3. To save the messages generated by the creation process, click the Save Message button. The Save As dialog box opens.
4. Select the directory where you the messages to be saved. The default name is GDResults.txt and the file is saved as a text file.
5. Click the Done button to close the Progress dialog box. The box closes and the report you just created opens.

Note: Click here for additional information on this report.

6. Once you have reviewed the report close the HTML Report Viewer.

Conclusion

This concludes our brief tour of GD*Pro's* Java reverse-engineering application.

We encourage you to give us feedback as to what you like and dislike about the product, as well as any input for additional features. You can use the e-mail feedback box located under HELP->COMMENTS.

We can be contacted at:

WWW:	http://www.advancedsw.com
Support:	support@advancedsw.com
e-mail:	info@advancedsw.com
Phone:	(303) 730-7981
FAX:	(303) 730-7983