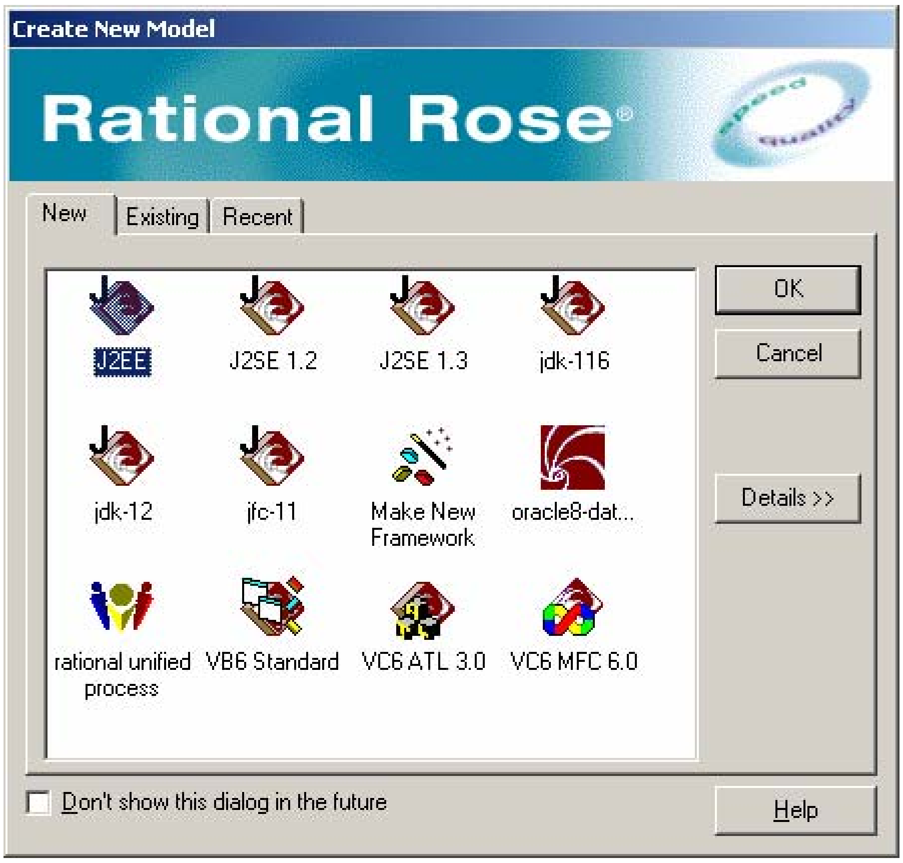
RATIONAL ROSE

1. Starting Rational Rose

Top of the Document Use Case Diagram ClassDiagram ActivityDiagram

When Rational Rose starts up, the following screen is displayed.

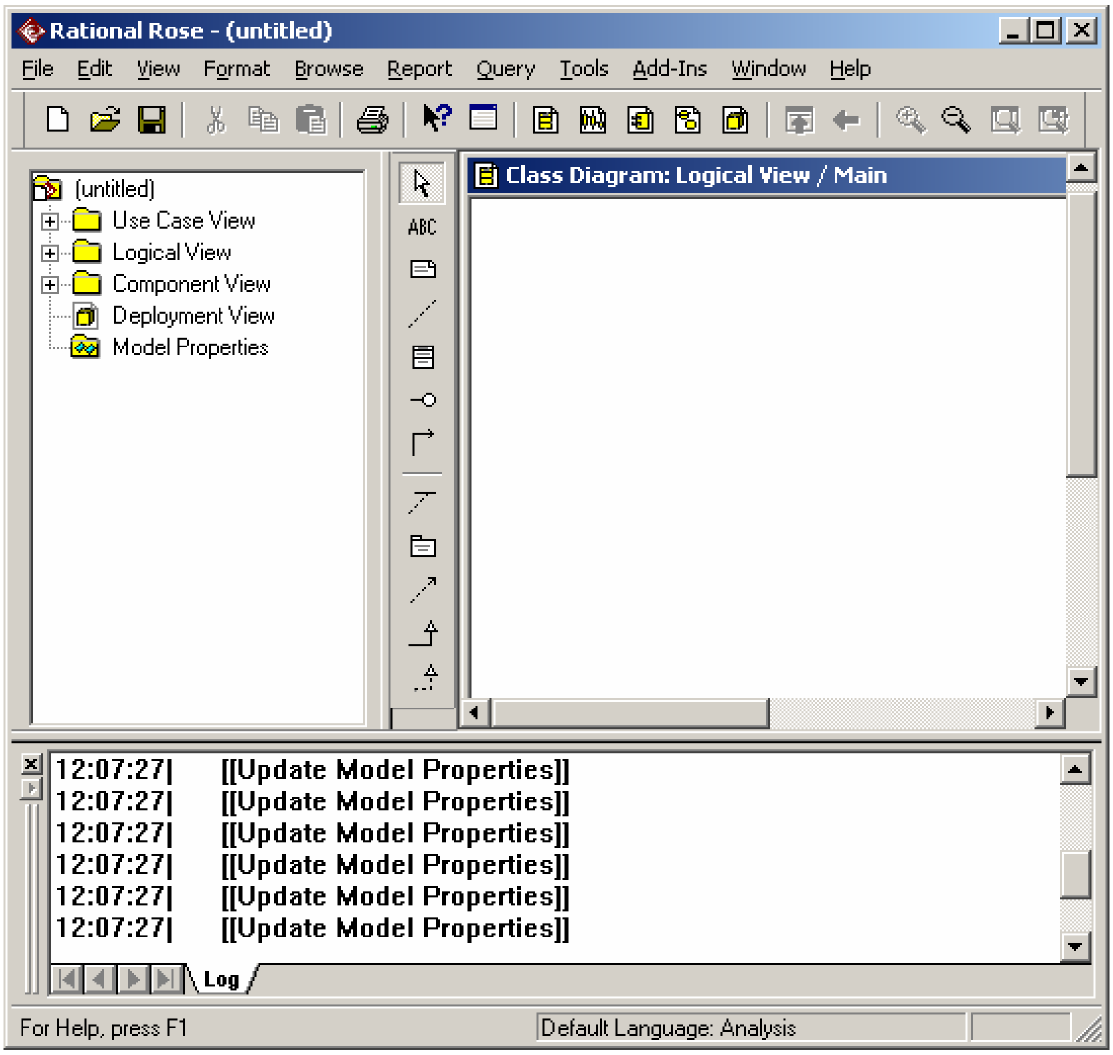


Create a new model, using Rational Unified Process.

2. Creating a project Model in Rational Rose

Top of the Document Use Case Diagram ClassDiagram ActivityDiagram

1. Start up Rational Rose Enterprise Edition.
2. Create a new model using the Rational Unified Process icon.
3. The window you will see will look something like this:

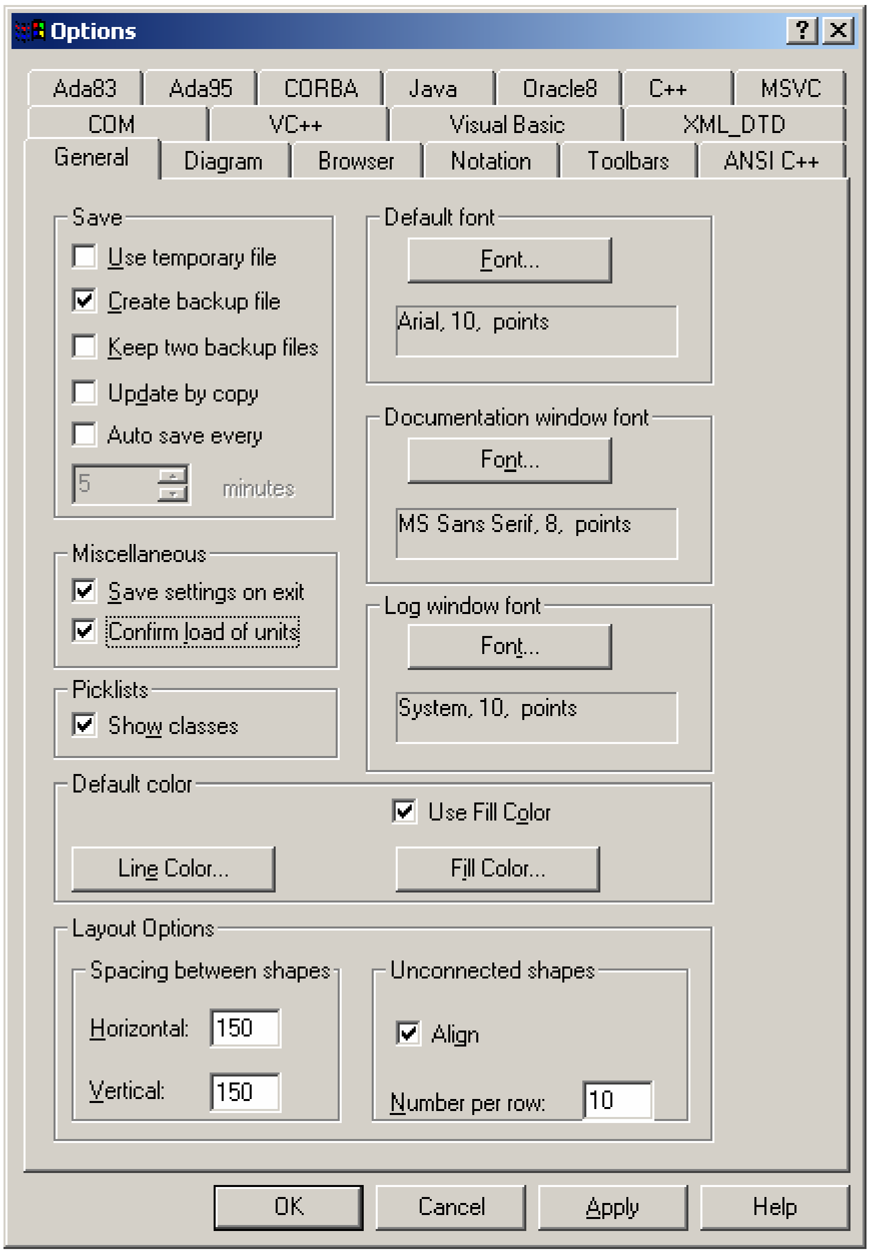

Rational Rose - (untitled)
File Edit View Format Browse Report Query Tools Add-Ins Window Help
(untitled)
Use Case View
Logical View
Component View
Deployment View
Model Properties
ABC
Class Diagram: Logical View / Main
Log
For Help, press F1
Default Language: Analysis

```
12:07:27|     [[Update Model Properties]]
12:07:27|     [[Update Model Properties]]
12:07:27|     [[Update Model Properties]]
12:07:27|     [[Update Model Properties]]
12:07:27|     [[Update Model Properties]]
12:07:27|     [[Update Model Properties]]
```

4. Using the View menu, turn off the log window. You will be left with the browser area on the left hand side, the application window, the standard toolbar and the diagram toolbox. This toolbox changes depending on which diagram you are drawing. The example shown is for a class diagram.

5. Configure the modelling tool, by double clicking Model Properties in the browser. Configure the tabs General, Diagram, Browser, Notation and Toolbars.

6. General, as follows



Save
- [ ] Use temporary file
- [x] Create backup file
- [ ] Keep two backup files
- [ ] Update by copy
- [ ] Auto save every 5 minutes

Miscellaneous
- [x] Save settings on exit
- [x] Confirm load of units

Picklists
- [x] Show classes

Default font: Font... Arial, 10, points

Documentation window font: Font... MS Sans Serif, 8, points

Log window font: Font... System, 10, points

Default color
- [x] Use Fill Color

Line Color... Fill Color...

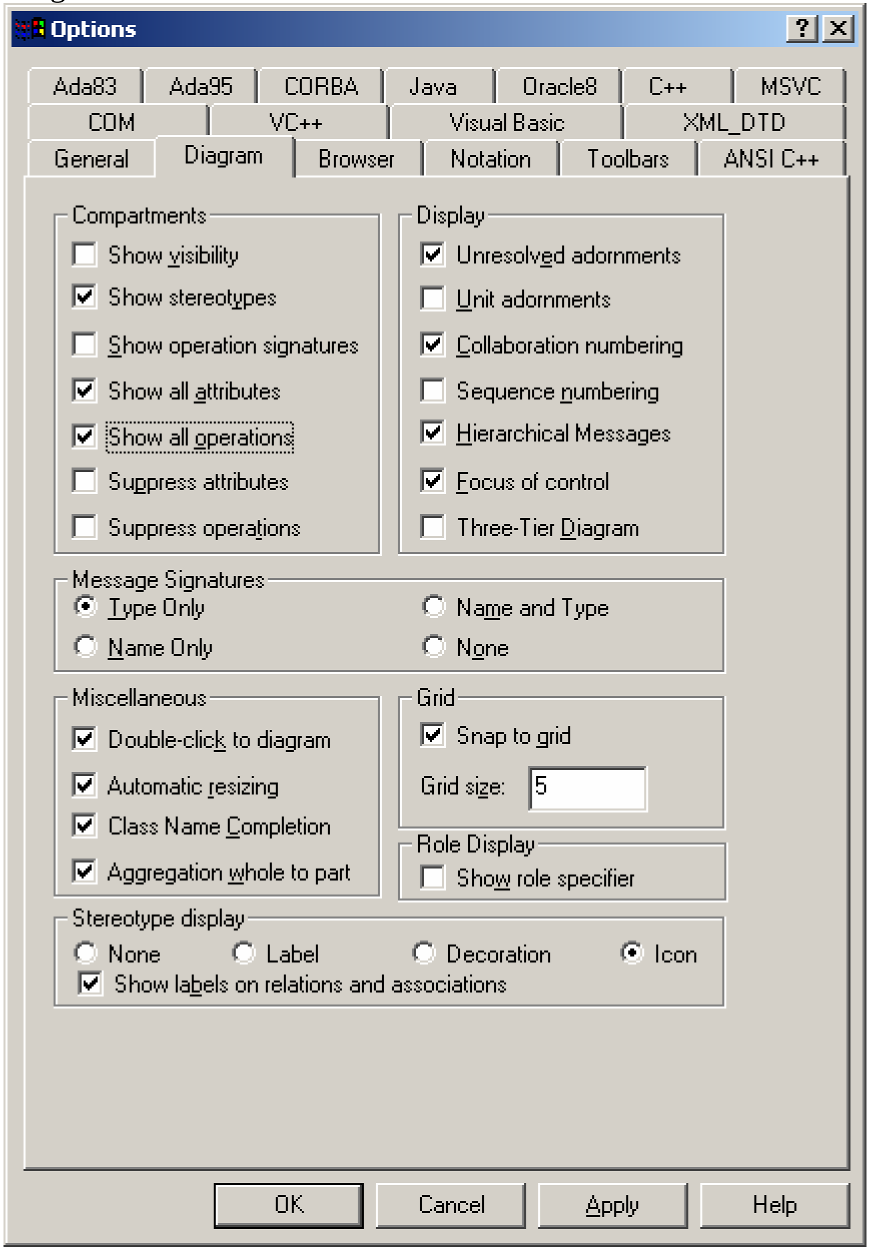
Layout Options

Spacing between shapes
Horizontal: 150
Vertical: 150

Unconnected shapes
- [x] Align

Number per row: 10

7. Diagram as follows:



Options

Ada83 | Ada95 | CORBA | Java | Oracle8 | C++ | MSVC
COM | VC++ | Visual Basic | XML_DTD
General | Diagram | Browser | Notation | Toolbars | ANSI C++

Compartments
- [ ] Show visibility
- [✓] Show stereotypes
- [ ] Show operation signatures
- [✓] Show all attributes
- [✓] Show all operations
- [ ] Suppress attributes
- [ ] Suppress operations

Display
- [✓] Unresolved adornments
- [ ] Unit adornments
- [✓] Collaboration numbering
- [ ] Sequence numbering
- [✓] Hierarchical Messages
- [✓] Focus of control
- [ ] Three-Tier Diagram

Message Signatures
[✓ Type Only] Name and Type Name Only None

Miscellaneous
- [✓] Double-click to diagram
- [✓] Automatic resizing
- [✓] Class Name Completion
- [✓] Aggregation whole to part

Grid
- [✓] Snap to grid
- Grid size: 5

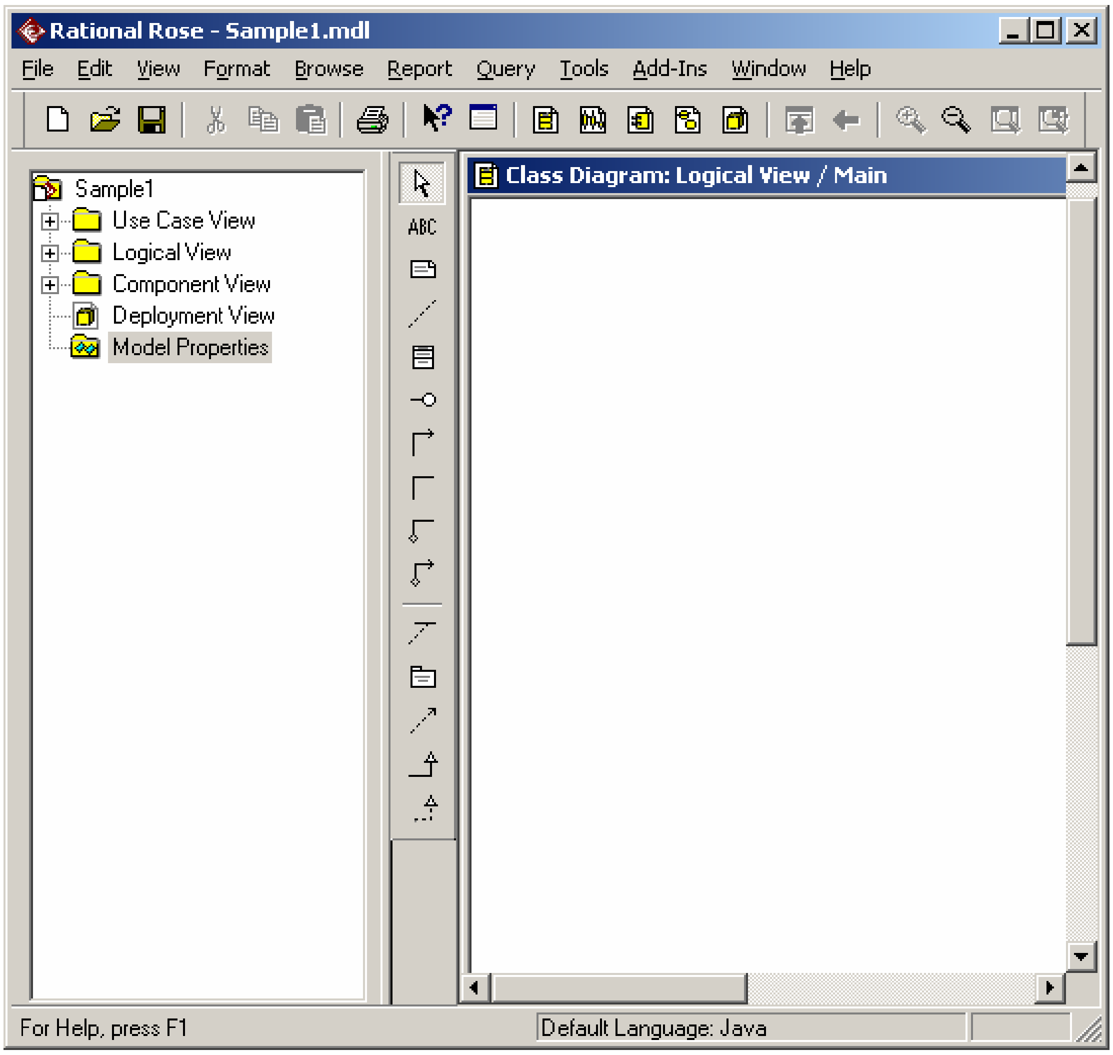
Role Display
- [ ] Show role specifier

Stereotype display
None Label Decoration [✓ Icon]
- [✓] Show labels on relations and associations

OK | Cancel | Apply | Help

8. Tick everything in the browser window and in the Notation window, use Unified notation, with a default language of Java. Leave other tick boxes blank.
9. In the toolbars tab, tick ‘Show standard toolbar’ and ‘Enable Docking’, ‘Show Diagram toolbar’ and ‘Enable docking’. Click the ‘…’ beside UML class diagram and add the following toolbar buttons to the current toolbar: ‘Creates an Association relationship’, ‘Creates an aggregation’ and ‘creates a unidirectional aggregation’. Click ‘close’ when all required buttons have been added.

10. Save your empty project model in a directory that is easily identifiable to you: e.g. F:/UML/Sample1. In future, this model will appear in the 'recent' tab when you go to open a model.


Rational Rose - Sample1.mdl
File Edit View Format Browse Report Query Tools Add-Ins Window Help
Sample1
Use Case View
Logical View
Component View
Deployment View
Model Properties
ABC
Class Diagram: Logical View / Main
For Help, press F1
Default Language: Java

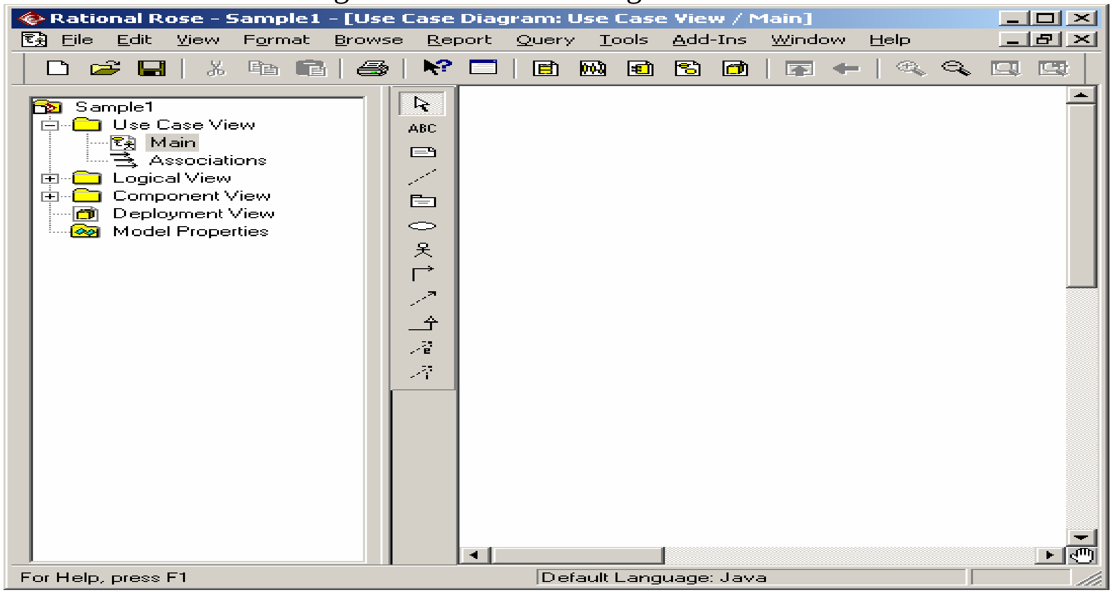
11. Close the model and the tool.

Top of the Document   Use Case Diagram ClassDiagram ActivityDiagram

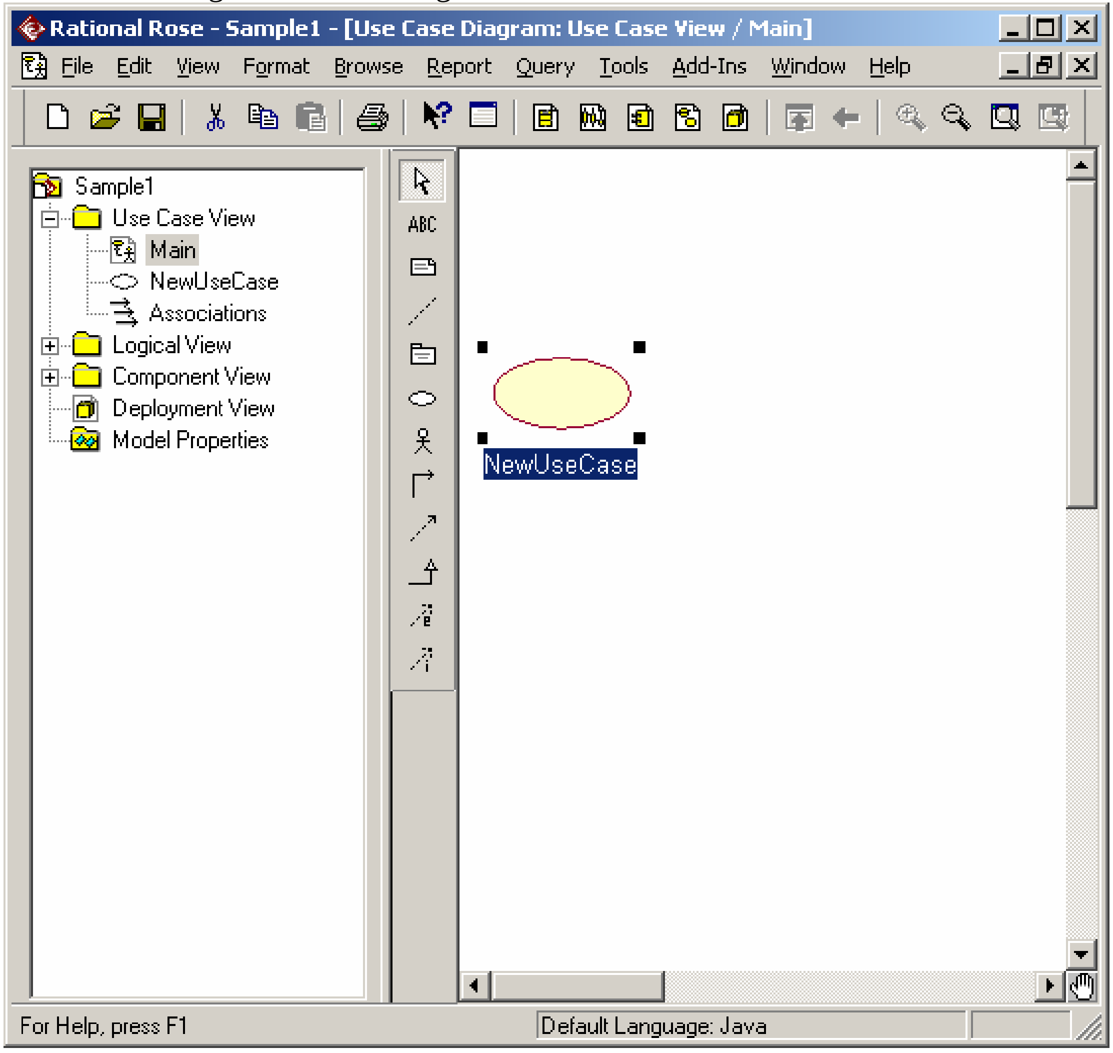
3. Use Case Diagram

1. Open the model previously created by starting up Rational Rose Enterprise Edition, choosing Create New Model and clicking the Existing tab and selecting the model by name. If a class diagram is open in the application window, close it.
2. In the Browser window, select Use Case View. Double click on Main. This opens the main Use Case. Note that the diagram toolbar has changed to reflect

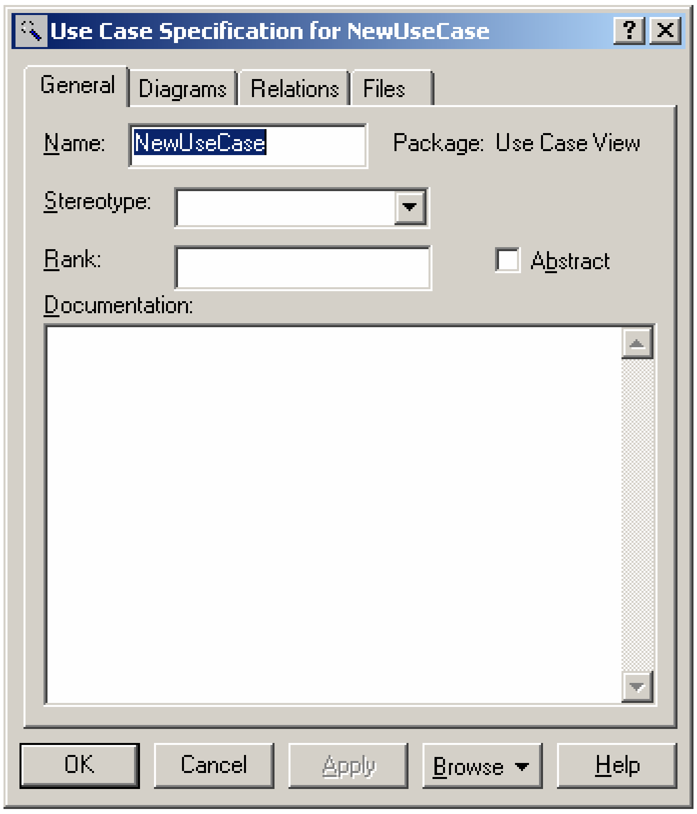
the fact that the active diagram is a Use Case diagram.



A Use Case can be added by clicking on the Use Case icon on the toolbar and then clicking in the active diagram.



3. The Use Case can be renamed while it is highlighted in blue. To rename it later, right click the use case and choose the Specification from the menu:



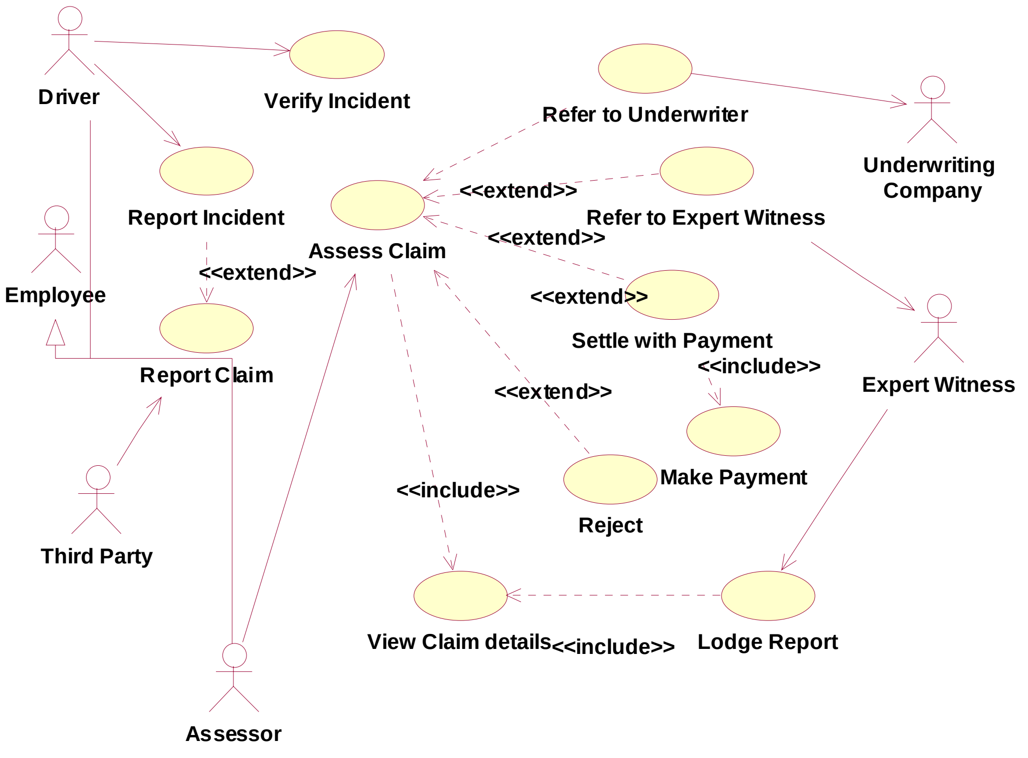
The use case can be renamed here.

4. Continue to add all necessary Use Cases to the diagram.
5. Add actors to the diagram by clicking on the Actor icon and clicking on the active diagram. As new actors are named, they appear in a list below any new actor that is represented on the diagram until such time as the new actor is named. This allows for the same actor to appear on a diagram twice. If the user decides to place the actor on the diagram twice, then (s)he will be prompted with the warning 'Newclass will be deleted from the model' Yes/No. WARNING: If you want to delete a duplicate icon from the model, be sure to use DEL – not Delete from Model. Delete from Model will remove all information on that actor / usecase/ class from the model.!!!

6. Use the 'Unidirectional Association' icon to draw the associations between actors and use cases.

7. Use the 'Extend Use Case' icon to extend a use case from another one**. If there is no button on the toolbar for this, right click on the toolbar and choose "customize…".** This will bring up the set of possible buttons. 'Extend Use Case' and 'Include Use Case' are about half way down the list. Add these to the toolbar and close the dialogue box.
8. Use the 'Include Use Case' to include the functionality of one use case in another.
9. Note that the items appear in the browser window when you insert them into the diagram.
10. If you wish to copy the diagram into Word, while the diagram is active, choose Edit from the standard toolbar and 'Copy Active Diagram'. This will put the diagram into the paste buffer.
11. An actor can be a generalisation of another actor or actors.
12. The Font of the diagram can be changed by changing the 'General' tab in the model properties, or by using the 'format' menu from the toolbar.

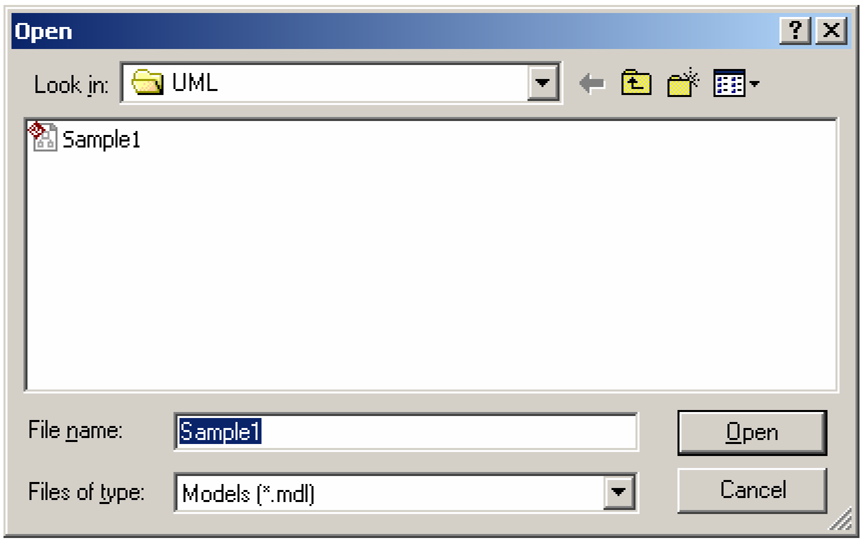
13. If you want to name a Use Case and you have left it, you can select it, right click and choose 'Open Specification'. Type over the current highlighted name.
14. When you 'save' it saves the entire model. If you want to add another diagram to this model, reopen the model.



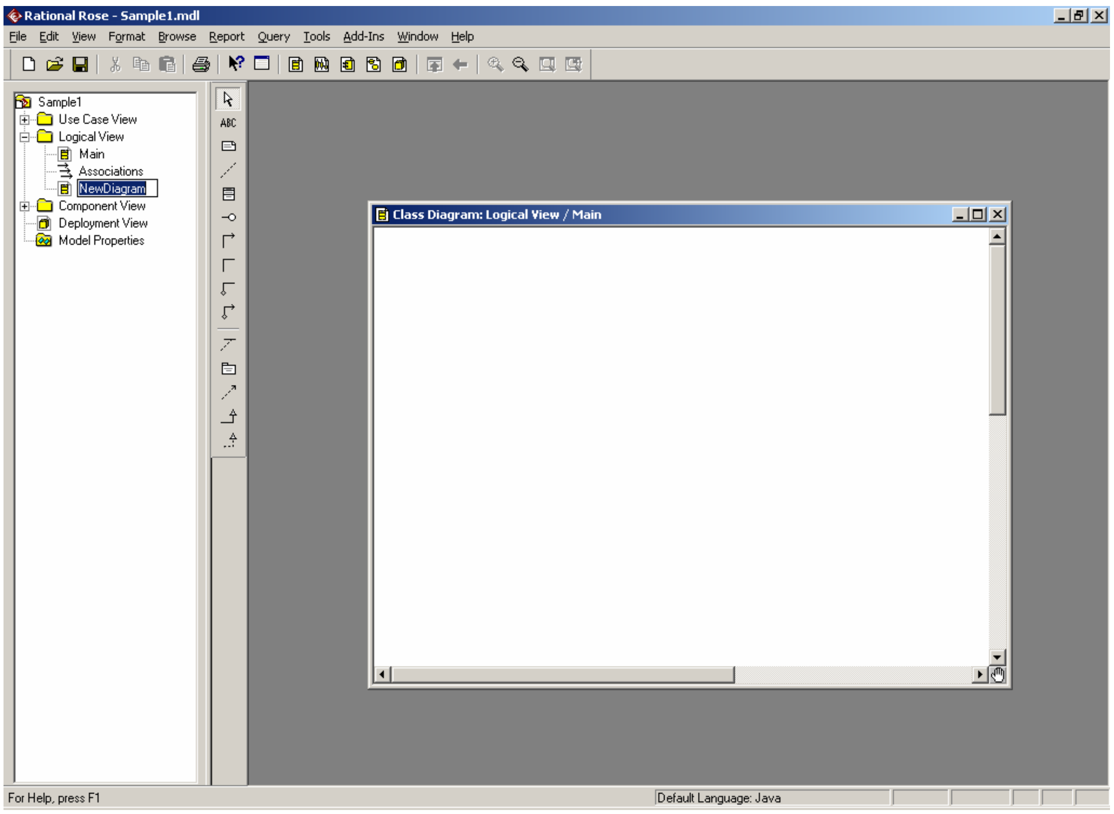
Top of the Document Use Case Diagram ActivityDiagram SequenceDiagram

4. Class Diagram

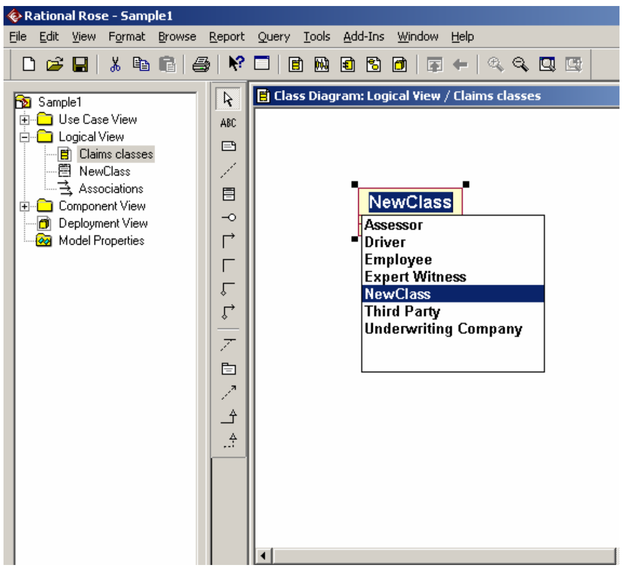
1. Start up Rational Rose using the Start Menu, Rational Suite Development Studio and Rational Rose Enterprise Edition (with red diamond beside it).
2. Using the Create New Model dialogue box, click on the Recent tab. Your project may be there. Alternately, click on the existing tab and browse to find your model.



This screen is automatically displayed:



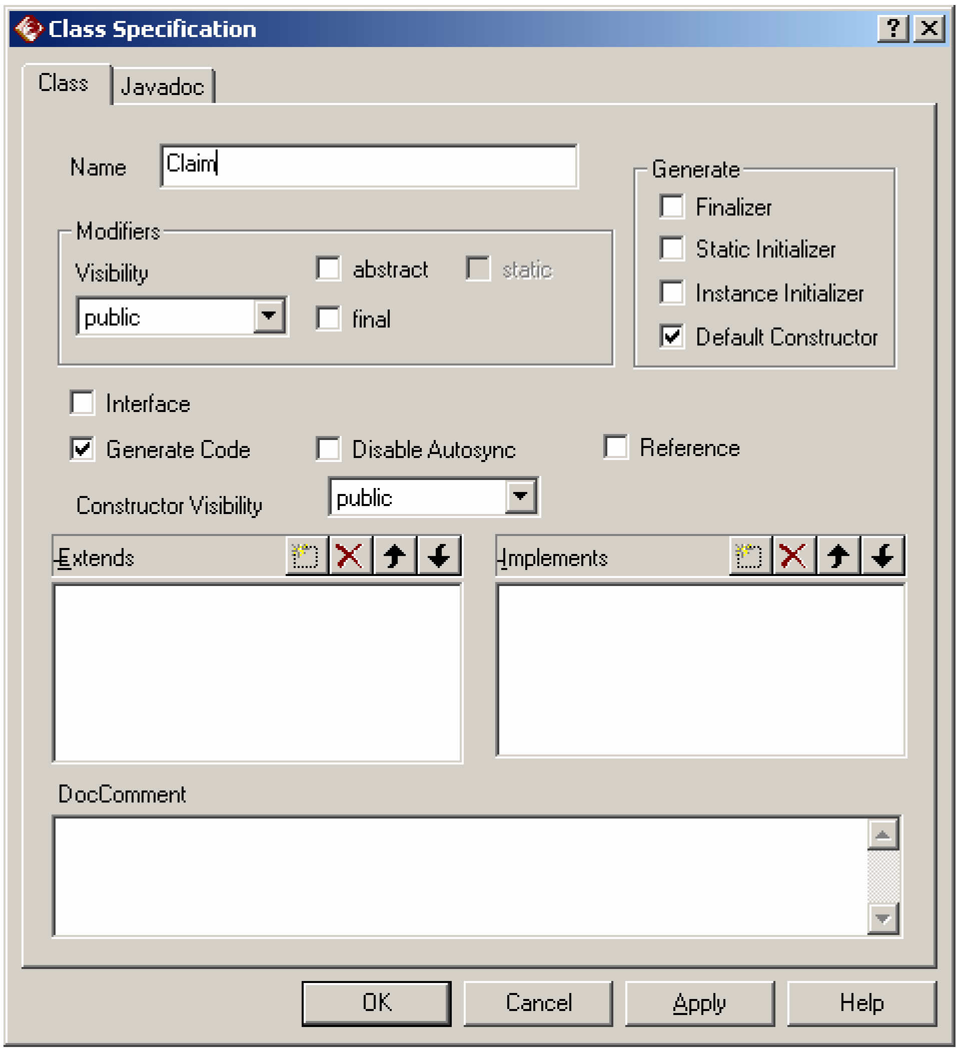
3. In the browser window, Click on the 'Logical View' folder, right click; click 'New' and 'Class Diagram' to create a new class diagram, naming it appropriately.



4. Open the new diagram by double-clicking its icon in the browser window.

5. Move the cursor over the icons on the Diagram Toolbox area to see what their functions are. Use the class icon to create and place a class in the diagram.

Note that the actors are included as possible labels for the class. Ignore these and replace the name NewClass with the name you want to put on the class. If you have omitted this step, you can rename it at any time by right-clicking on the class and choosing 'Open Specification' as follows:


Class Specification

Class | Javadoc

Name: Claim

Modifiers
Visibility: public
[ ] abstract [ ] static
[ ] final

Generate
[ ] Finalizer
[ ] Static Initializer
[ ] Instance Initializer
[✓] Default Constructor

[ ] Interface
[✓] Generate Code [ ] Disable Autosync [ ] Reference

Constructor Visibility: public

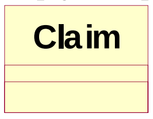
Extends

Implements

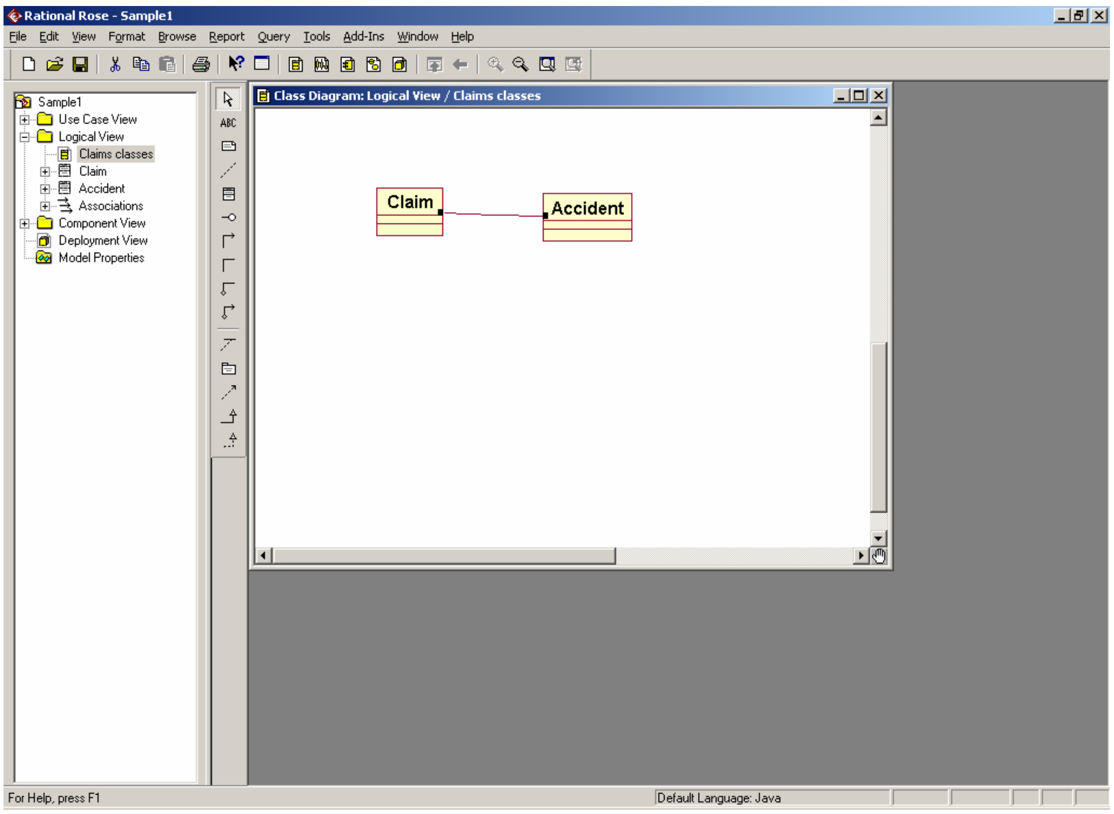
DocComment

OK | Cancel | Apply | Help

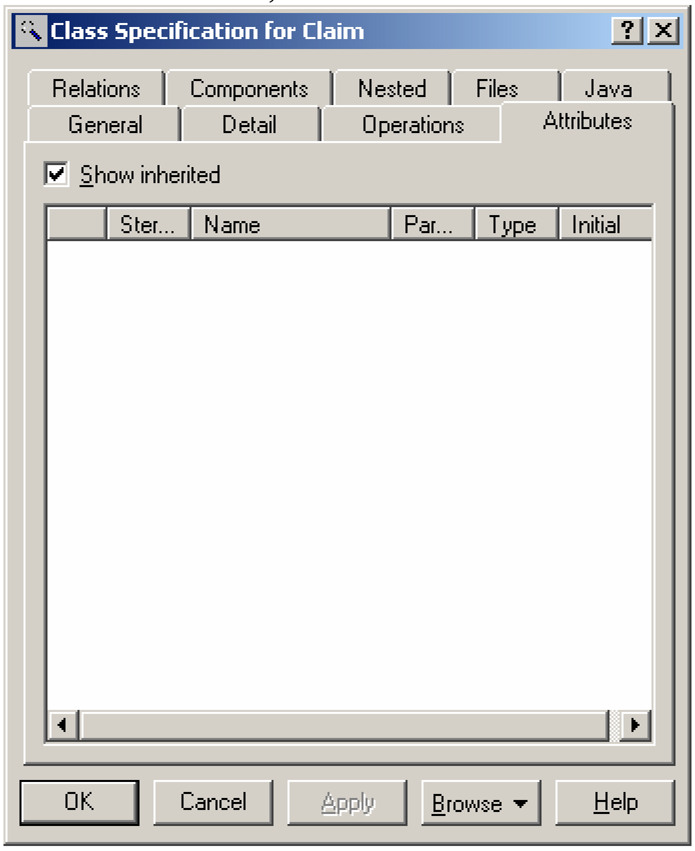
Leave all other boxes as they are for now. The class will be labelled, with two empty compartments below it:



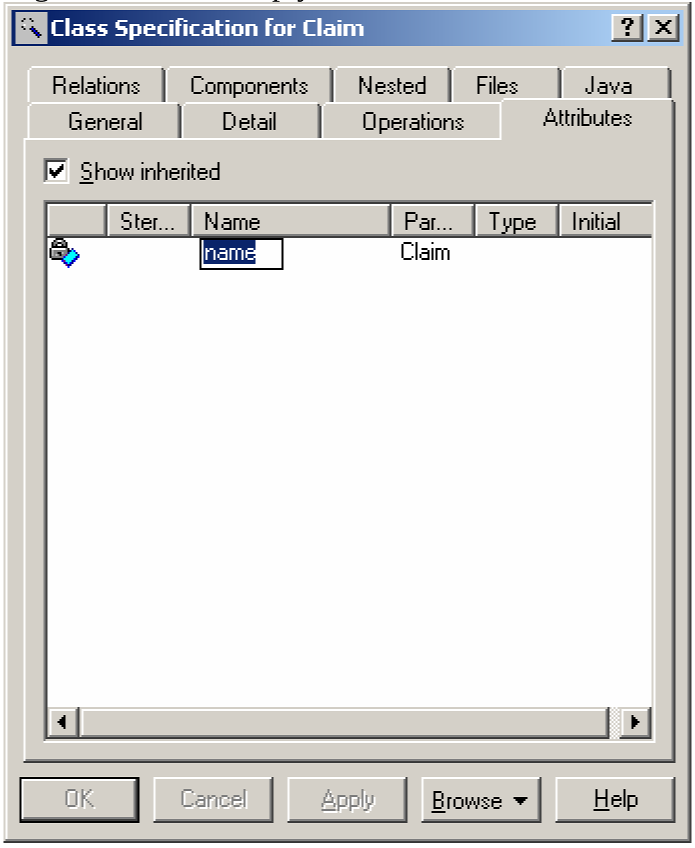
6. Create the rest of the classes from the diagram you have developed in tutorials.
7. You can modify the display format to display or suppress attributes or operations, or to show the signature of an operation.
8. When the classes are put in, you can add associations, using either the uni-directional association or the bi-directional association.



9. Right click on the association and ensure that ‘Stereotype label’, ‘Public’ and ‘Navigable’ are ticked. To specify the association further ‘Open standard specification’. You can add multiplicity and roles.
10. To add attributes, click the ‘attributes’ tab on the class specification.



Right click in the empty box and fill in the attribute name

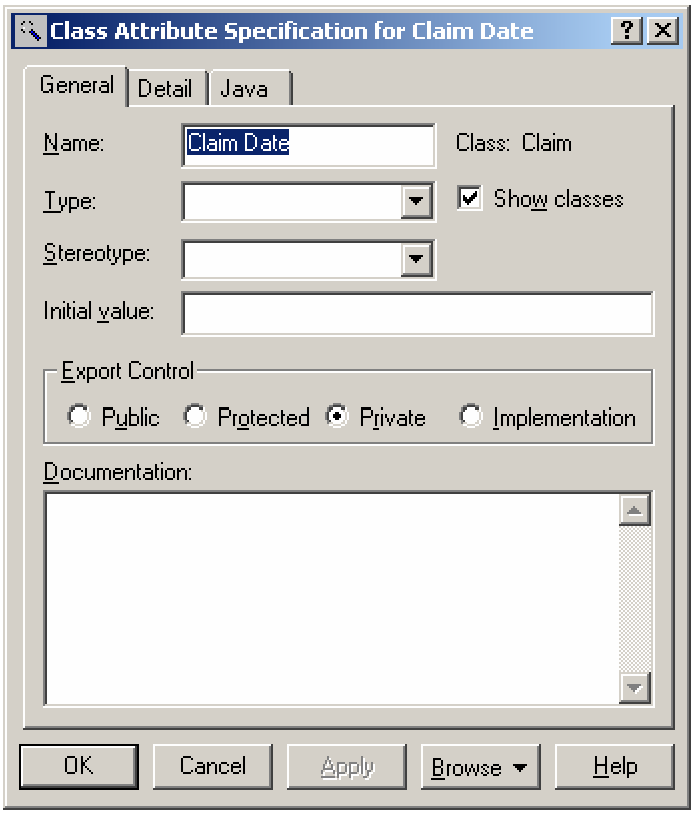

Class Specification for Claim

Relations | Components | Nested | Files | Java
General | Detail | Operations | Attributes

[✓] Show inherited

	Ster...	Name	Par...	Type	Initial
		name	Claim		

Double click on the attribute to add more detail:


Class Attribute Specification for Claim Date

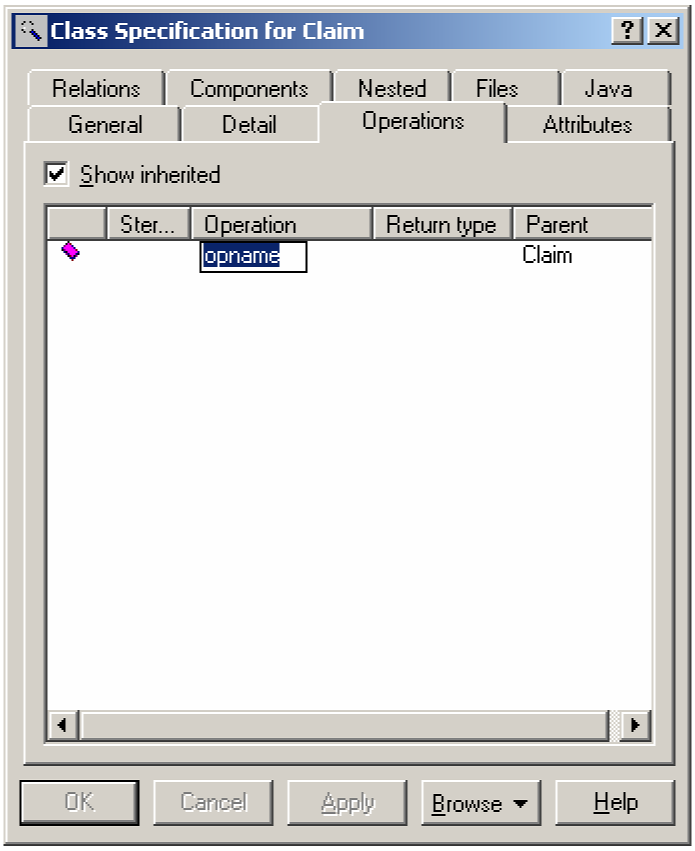
General | Detail | Java

Name: Claim Date
Class: Claim
Type:
[✓] Show classes
Stereotype:
Initial value:

Export Control
Public Protected [✓ Private] Implementation

Documentation:

Operations are added using the operations tab:


Class Specification for Claim

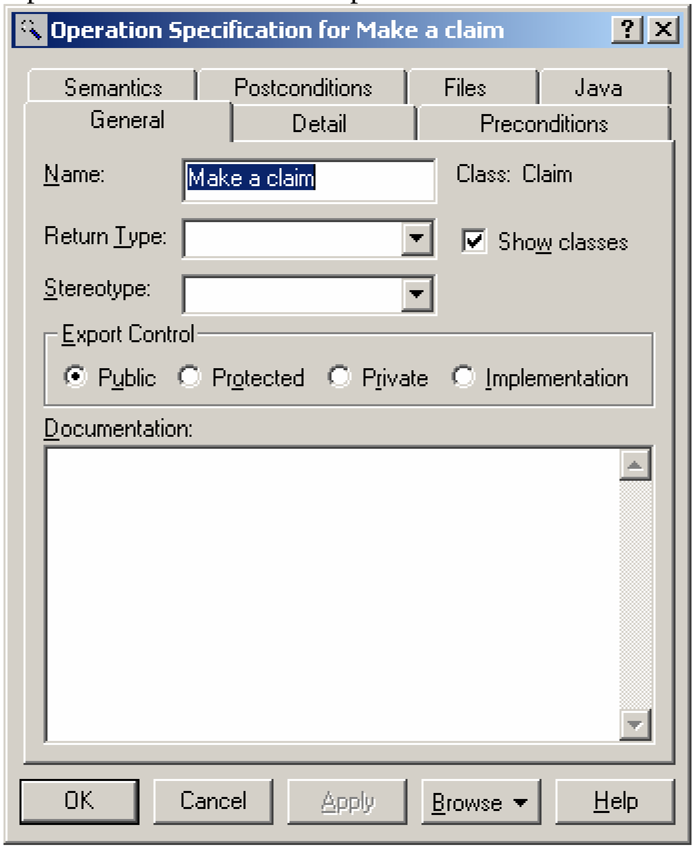
Relations | Components | Nested | Files | Java
General | Detail | Operations | Attributes

[✓] Show inherited

	Ster...	Operation	Return type	Parent
		opname		Claim

OK | Cancel | Apply | Browse | Help

Operations can be further specified:


Operation Specification for Make a claim

Semantics | Postconditions | Files | Java
General | Detail | Preconditions

Name: Make a claim

Class: Claim

Return Type:

[✓] Show classes

Stereotype:

Export Control: [✓ Public] Protected Private Implementation

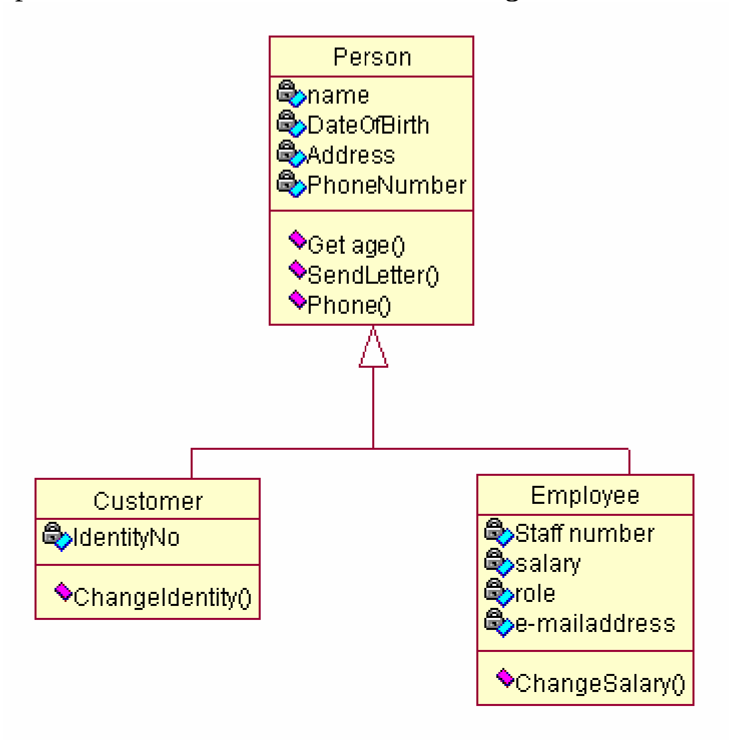
Documentation:

OK | Cancel | Apply | Browse | Help

Adding other associations

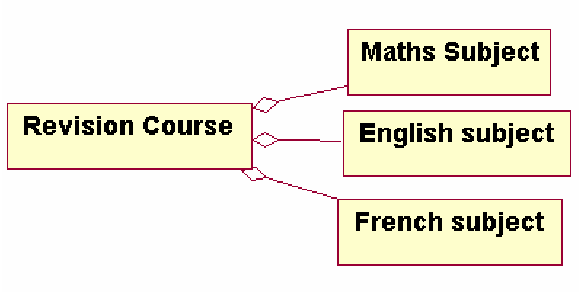
Generalisation

To add a generalisation, click on the Generalisation icon . Start from the specialised class and draw towards the general class.



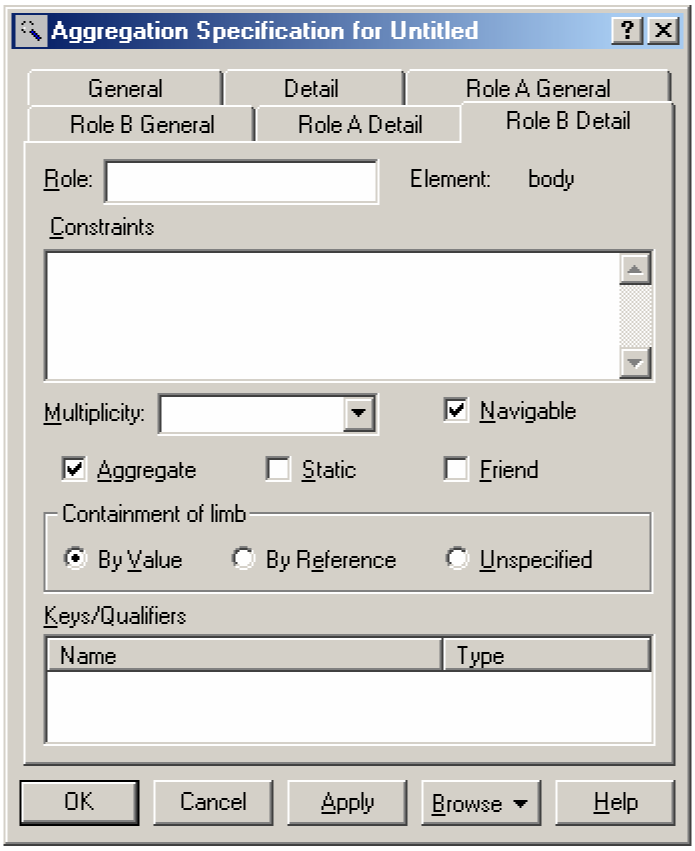
Aggregation

To create an aggregate relationship, either use the Aggregation icon or right click on an association and click the aggregate tick box. Start at class A, where A is the containing class. Draw to class B, where B is the part class.



Composition

To create a composition relationship, add an aggregation. Open the Specification of the aggregation and choose Role B detail (If the aggregate tick box is not ticked, then choose Role A detail instead). Tick the 'containment' radio button to fill in the aggregate diamond.


Aggregation Specification for Untitled

General | Detail | Role A General | Role B General | Role A Detail | Role B Detail

Role:
Element: body

Constraints

Multiplicity:

[✓] Navigable

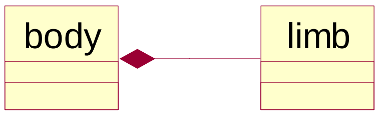
[✓] Aggregate [ ] Static [ ] Friend

Containment of limb

[● By Value] ○ By Reference ○ Unspecified

Keys/Qualifiers

Name	Type



11. When the model is saved, the diagram will be saved along with it.

Top of the Document Use Case Diagram ClassDiagram ActivityDiagram

5. Activity Diagram

If you are not already running Rational Rose, start it using the Start menu, Program Files, Rational Rose Enterprise Edition. Open the model from where you previously saved it. If you have not saved a model previously, read section 1.

Right click on the Use Case View and choose New and Activity diagram. The tools in the toolbar that you will need are:

This is the start state, which signifies the start of the workflow.

This is the end state, which signifies the end of the workflow.

This is the transition between activities, activity and state, activity and decision, decision and activity or state and activity.

This is the decision diamond.

This is the activity.

This is the state.

Use the lecture to explain how to draw the diagrams.

Top of the Document   Use Case Diagram ClassDiagram ActivityDiagram

6. Sequence Diagram

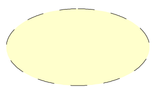
When drawing a Sequence diagram, it is assumed that you have previously:-

- Drawn a Use Case diagram in the Use Case View and populated it with actors and use cases.
- Set up at least one control class to control the Use Case.
- Set up (business) entity classes in the Logical View complete with attributes and operations.
- Set up boundary classes in the Logical View, complete with attributes and operations and any inherited classes.

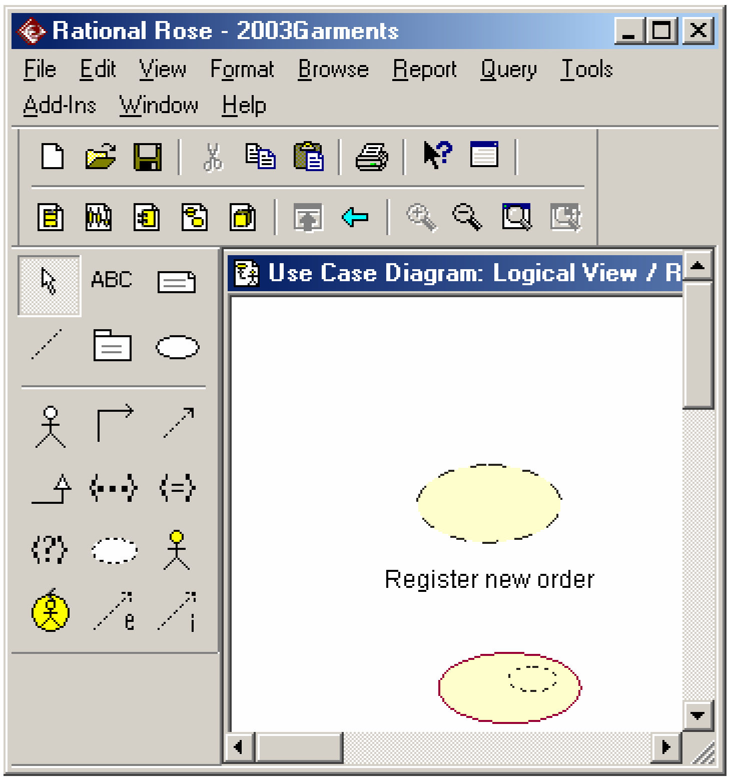
Before you draw a sequence diagram, you must have a realisation of the Use Case that you are representing. To create this, right click on Logical view in the browser and pick New Use Case Diagram

Call it ‘Use Case Realisation’

Populate the new Use Case diagram with one Use case for each one from the Use Case view. **USE THE ICON FROM THE TOOLBAR TO DO THIS**. If necessary, customise the toolbar to add this icon. It is called ‘Creates a Use Case realisation’. Open the specification on the Use case, give it the same name as the one in the Use Case view and give it a stereotype of ‘Use Case Realisation’. When this has been done, the realisations will come up in the diagram and in the logical view ***browser*** as dotted ovals.

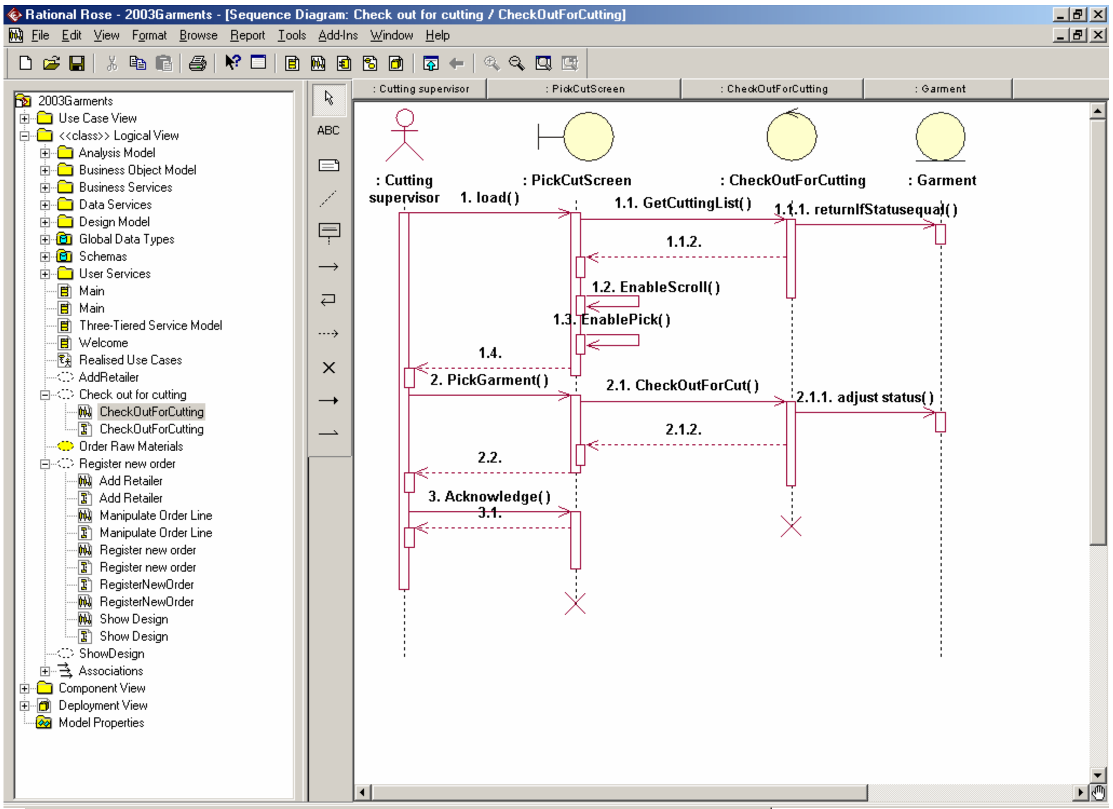

Order Raw Materials

Right click on the dotted oval ***in the browser*** and choose ‘New Sequence Diagram’. Give it the same name as the Use Case.


Rational Rose - 2003Garments
File Edit View Format Browse Report Query Tools
Add-Ins Window Help
ABC
Use Case Diagram: Logical View / R
Register new order

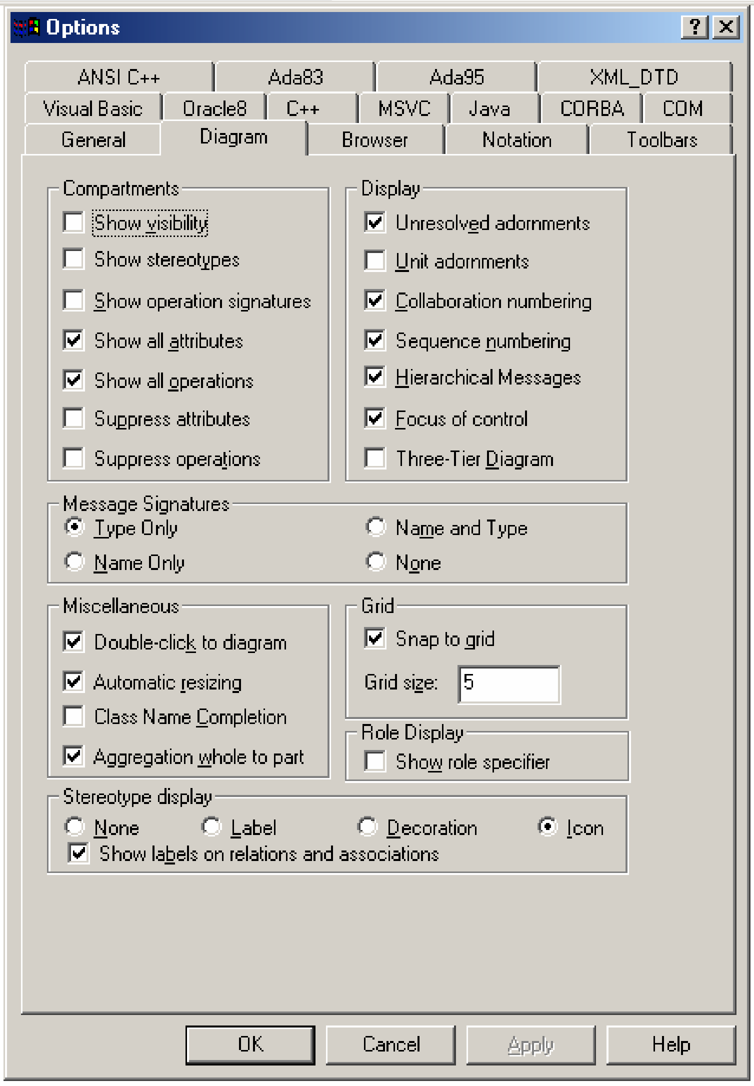
Pick required actors and classes from the browser and drag them onto the diagram. Depending on the options you have on the view, they may all be shown as rectangles, or may use the icon representing their stereotype.

To add an event / signal / message between classes; use the straight arrow icon → from the toolbar. Start at the originating class and drag it to the receiving class. Right click on the event and pick the operation from the list shown. If the operation is not in the class already, add it. The message can be edited and its specification altered to make it synchronous or to make sure there is a significant return value: To add a new sequence diagram (e.g. for another use case), develop the other sequence diagram (e.g. DiagramB) add a note to the original diagram (e.g. Diagram A). Drag DiagramB from the browser into the note on Diagram A. To end an object's lifeline in the Use Case, use the ✕ icon on its lifeline.


Rational Rose - 2003Garments - [Sequence Diagram: Check out for cutting / CheckOutForCutting]

File Edit View Format Browse Report Tools Add-Ins Window Help

- 2003Garments
  - Use Case View
  - <<class>> Logical View
    - Analysis Model
    - Business Object Model
    - Business Services
    - Data Services
    - Design Model
    - Global Data Types
    - Schemas
    - User Services
    - Main
    - Main
    - Three-Tiered Service Model
    - Welcome
    - Realised Use Cases
    - AddRetailer
    - Check out for cutting
      - CheckOutForCutting
      - CheckOutForCutting
    - Order Raw Materials
    - Register new order
      - Add Retailer
      - Add Retailer
      - Manipulate Order Line
      - Manipulate Order Line
      - Register new order
      - Register new order
      - RegisterNewOrder
      - RegisterNewOrder
      - Show Design
      - Show Design
    - ShowDesign
    - Associations
  - Component View
  - Deployment View
  - Model Properties


Options

ANSI C++ | Ada83 | Ada95 | XML_DTD
Visual Basic | Oracle8 | C++ | MSVC | Java | CORBA | COM
General | **Diagram** | Browser | Notation | Toolbars

Compartments
- ☐ Show visibility
- ☐ Show stereotypes
- ☐ Show operation signatures
- ☑ Show all attributes
- ☑ Show all operations
- ☐ Suppress attributes
- ☐ Suppress operations

Display
- ☑ Unresolved adornments
- ☐ Unit adornments
- ☑ Collaboration numbering
- ☑ Sequence numbering
- ☑ Hierarchical Messages
- ☑ Focus of control
- ☐ Three-Tier Diagram

Message Signatures
- [✓ Type Only] Name and Type Name Only None

Miscellaneous
- ☑ Double-click to diagram
- ☑ Automatic resizing
- ☐ Class Name Completion
- ☑ Aggregation whole to part

Grid
- ☑ Snap to grid
- Grid size: 5

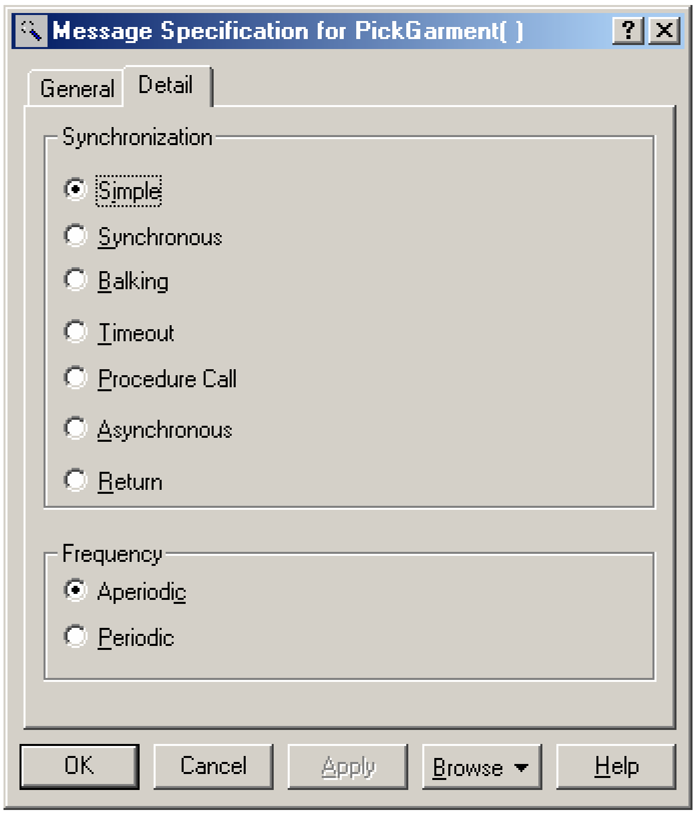
Role Display
- ☐ Show role specifier

Stereotype display
- None Label Decoration [✓ Icon]
- ☑ Show labels on relations and associations

OK | Cancel | Apply | Help

Note that in this diagram, the messages are numbered sequentially and hierarchically. To show this, edit the Model Properties (see browser pane). Choose the 'Diagram' tab and ensure that the three tick boxes called 'Collaboration numbering', 'Sequence numbering' and 'Hierarchical Messages' are ticked. The 'Icon' radio button should also be chosen.

First, populate the diagram with instances of the required classes. This can be done by dragging the class from the browser onto the screen. Try to keep the actor to the left side, followed by any boundary classes, control classes and entity classes, in that order. If you name the instance, the instance name will be separated from the class name by a colon. You may choose to use an anonymous instance. As soon as a class sends a message, it shows activation. The receiving class also shows activation. The message can specify which operation in the receiving class it invokes. The simple message can be named. The nature of the message can be altered, by choosing


Message Specification for PickGarment( )

General | Detail

Synchronization

- [✓ Simple]
- Synchronous
- Balking
- Timeout
- Procedure Call
- Asynchronous
- Return

Frequency

- [✓ Aperiodic]
- Periodic

'Open Specification' and choosing the detail tab. The nature of the message protocol can be specified.

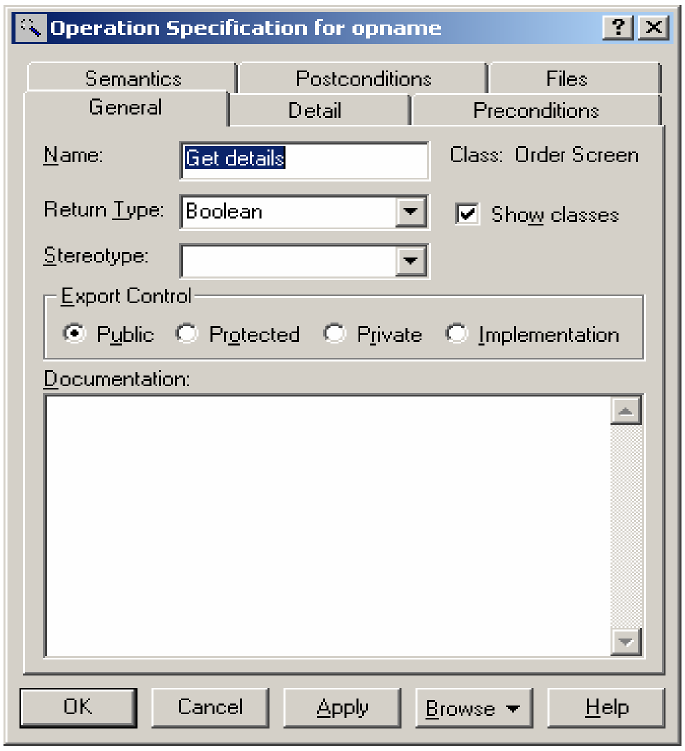
In procedural systems, most of the calls are either simple or synchronous. Synchronous suggests a dialogue between the calling and called objects. The simple message causes a thread to begin, which holds up all objects that it involves, until the object has received a 'Return' message to show that it has completed. For example, in the sequence diagram shown above, the actor is committed to a single thread of activity from message 1. Load until it receives back 1.4, a return. The actor is then free to activate another thread. This is the case for non-concurrent, single-thread systems.

Rational Rose only allows scenarios to be modelled. Selection and iteration cannot be modelled.

7. Collaboration Diagrams

Top of the Document   Use Case Diagram ClassDiagram ActivityDiagram

To create a collaboration diagram in Rational Rose, ensure first that the Use Case that you want to illustrate is present and that all classes have been set up, complete with



Operation Specification for opname

Tabs: Semantics | Postconditions | Files | General | Detail | Preconditions

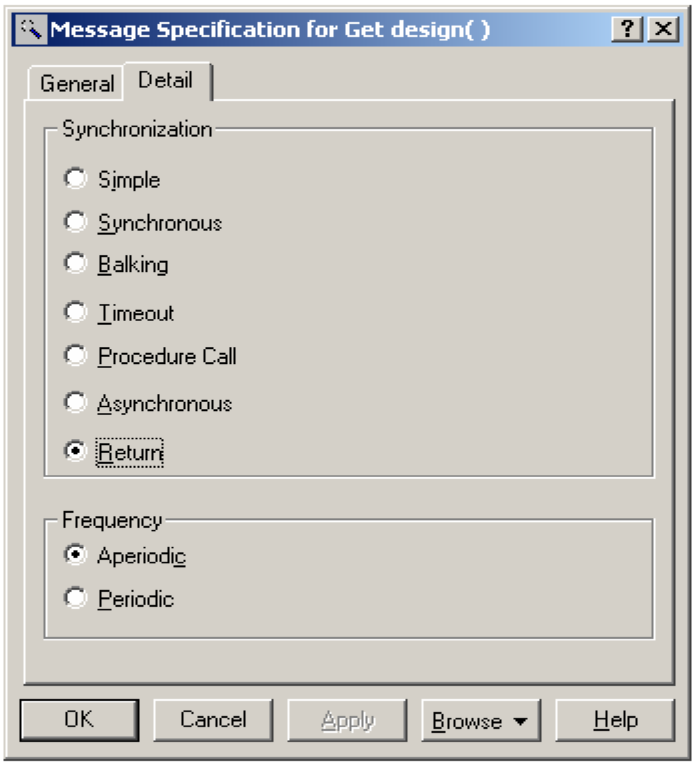
Name: Get details   Class: Order Screen

Return Type: Boolean   [✓] Show classes

Stereotype:

Export Control: [✓ Public] Protected Private Implementation

Documentation:



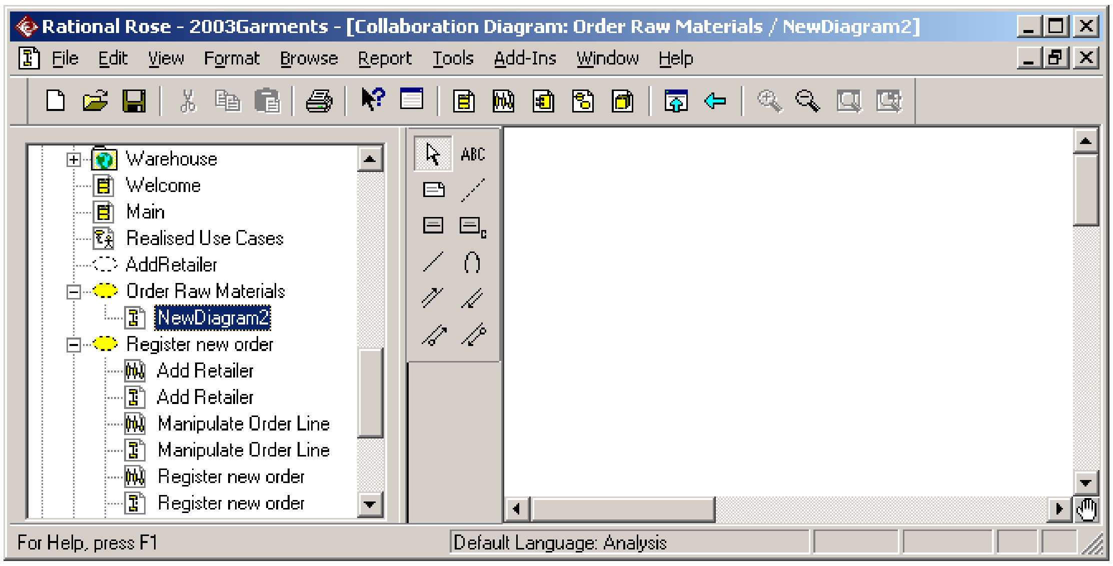
Message Specification for Get design( )

Tabs: General | Detail

Synchronization: Simple / Synchronous / Balking / Timeout / Procedure Call / Asynchronous / [✓ Return]

Frequency: [✓ Aperiodic] / Periodic

their stereotypes.



Rational Rose - 2003Garments - [Collaboration Diagram: Order Raw Materials / NewDiagram2]

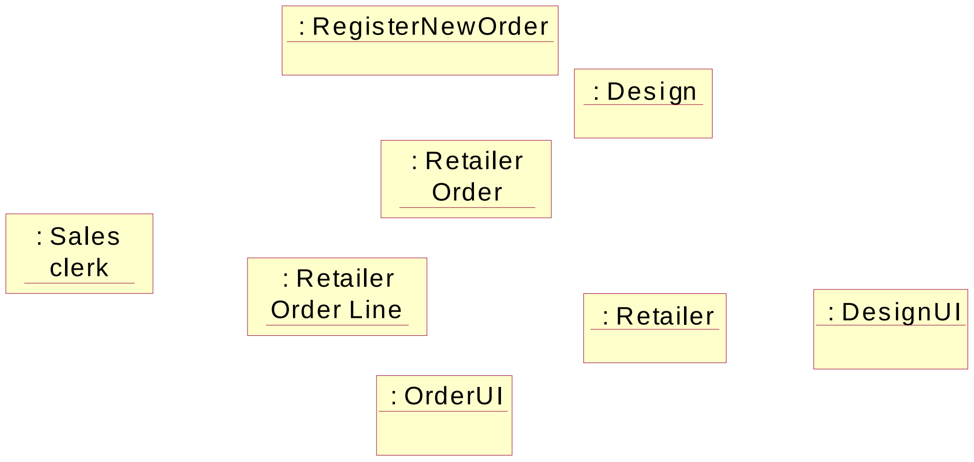
Warehouse
Welcome
Main
Realised Use Cases
AddRetailer
Order Raw Materials
NewDiagram2
Register new order
Add Retailer
Add Retailer
Manipulate Order Line
Manipulate Order Line
Register new order
Register new order

For Help, press F1   Default Language: Analysis

Right click on the Use Case in the browser and choose ‘New’ and ‘Collaboration Diagram’. Give the diagram the same name as the Use Case and double click on it to open it. Note the toolbar should look as above.

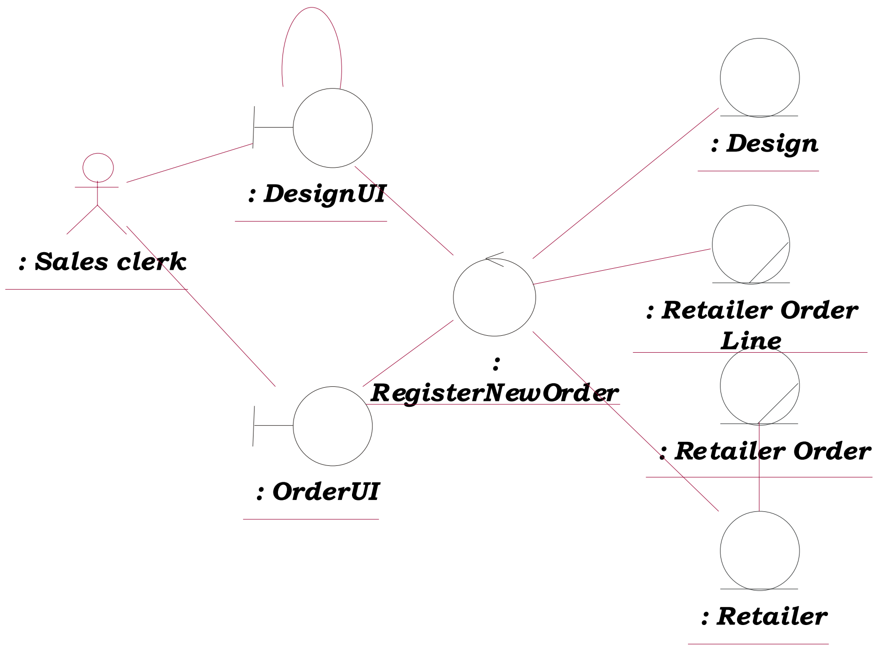
A Collaboration diagram has classes, links and messages. The links show how the classes communicate, while the messages travel on the links. Any two classes that communicate must be joined by links. Two classes may only be joined by one link ⁄, but there can be many messages passing between them. The messages are directional, so use either ⇗ or ⇙.

To draw the diagram, drag the objects from the browser.



When you do this, they may appear as boxes. When you have all of the objects on the diagram, do the following:

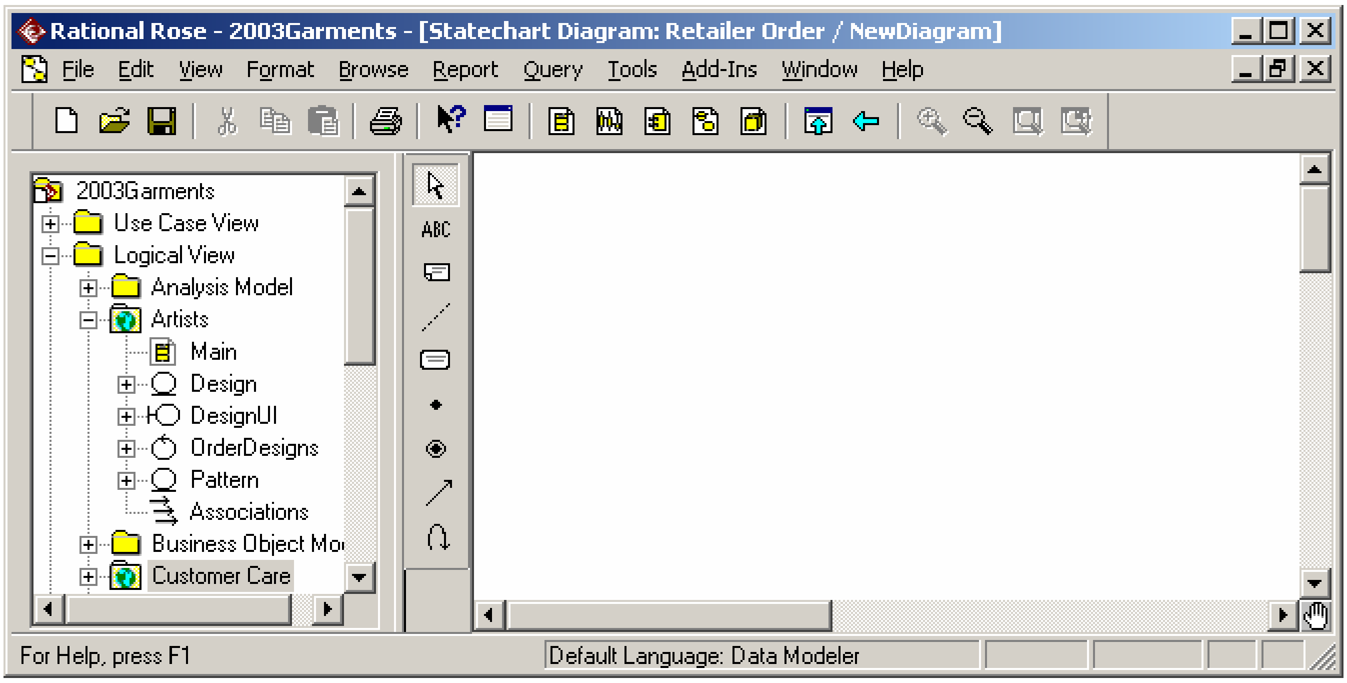
Select all from the edit menu. Go to Format menu and set stereotype to icon. At format menu, disable 'use fill colour' (if you wish!)At format menu, choose font and change the font to something that will distinguish the objects from the message names. (I chose bold italic 12 point bookman old style). Connect with links all objects that send messages to each other.



Go to top   Use Case Diagram Class Diagram Sequence Diagram Collaboration

8. Drawing a State Diagram

In the class diagram, select the persistent class for which you wish to draw a state diagram. Right click and choose Subdiagrams, 'New Statechart diagram'. Alternately, a statechart diagram can be started by right clicking on the Logical View and choosing 'New' and StateChart Diagram. However, this does not attach the diagram to an existing class.


Rational Rose - 2003Garments - [Statechart Diagram: Retailer Order / NewDiagram]
File Edit View Format Browse Report Query Tools Add-Ins Window Help
2003Garments
Use Case View
Logical View
Analysis Model
Artists
Main
Design
DesignUI
OrderDesigns
Pattern
Associations
Business Object Mo
Customer Care
ABC
For Help, press F1
Default Language: Data Modeler

Using the icons on offer, draw the diagram.

Go to top Use Case Diagram Class Diagram Sequence Diagram Collaboration

9. Generating tables from classes

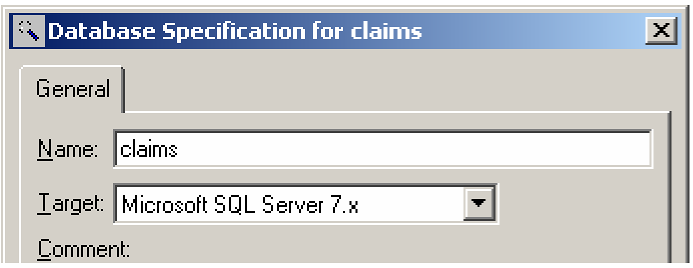
Converting classes to tables

Remember:

- You can only generate tables from PERSISTENT classes.
- To generate tables, there must be a defined schema.
- The schema must be associated with a database in the component view.

In the component view:

Set up a database (*Right click on logical view, choose data modeler, new, database).*


Database Specification for claims

General

Name: claims

Target: Microsoft SQL Server 7.x

Comment:

Open the specification for the database and associate it with whichever implementation route suits (e.g. SQL Server 7)

In the logical view:

(a) **Classify the classes**, using a 3-tier system. (To do this, t*ick the 'Three-tier diagram' box in the tools - options menu in Rational Rose.)*

Move all persistent classes into one package.

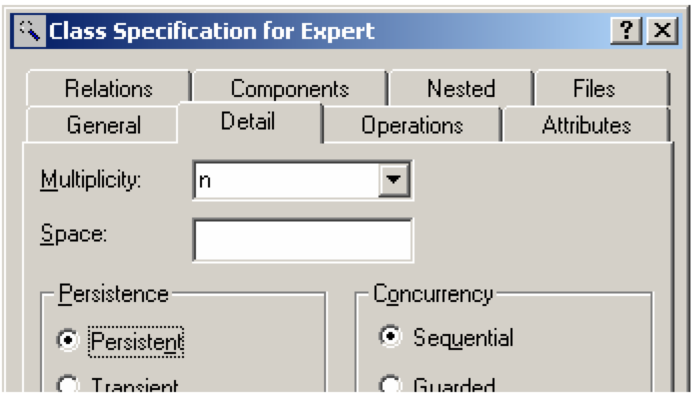
For each persistent class:

Open the standard specification

Select the 'detail' tab

Turn on the 'persistent' radio button.

If you have not already given the attributes data types, do so now.



Multiplicity: n

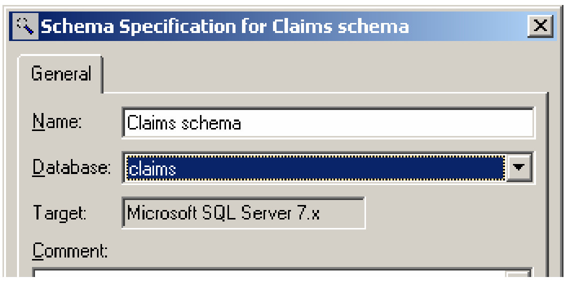
Space:

Persistence: [✓ Persistent] Transient

(b) **Set up a new schema** (Right click on logical view, choose data modeler, new, schema.)

Open the schema specification.

Associate it with the database you have created.



Name: Claims schema

Database: claims

Target: Microsoft SQL Server 7.x

Comment:

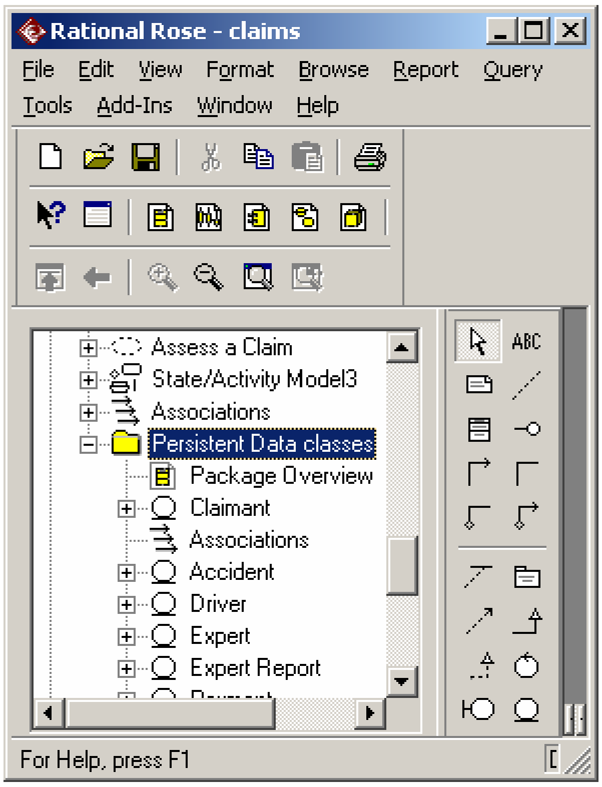
(c) **Transform the objects to data**. Right click on the package that holds the classes in the Logical view.

Choose data modeler

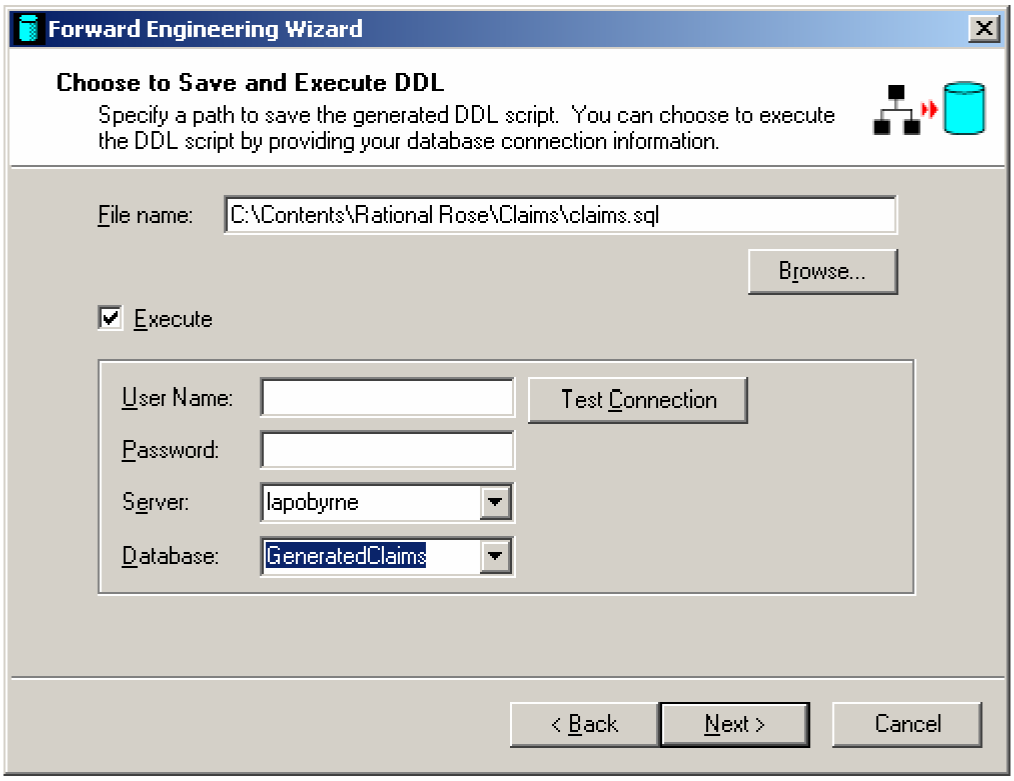
Choose 'Transform to data model'

Fill in destination schema and target database that you have set up. Execute the transformation. *(To see your tables, expand the schema. If they aren't there, maybe you didn't make them persistent?).*

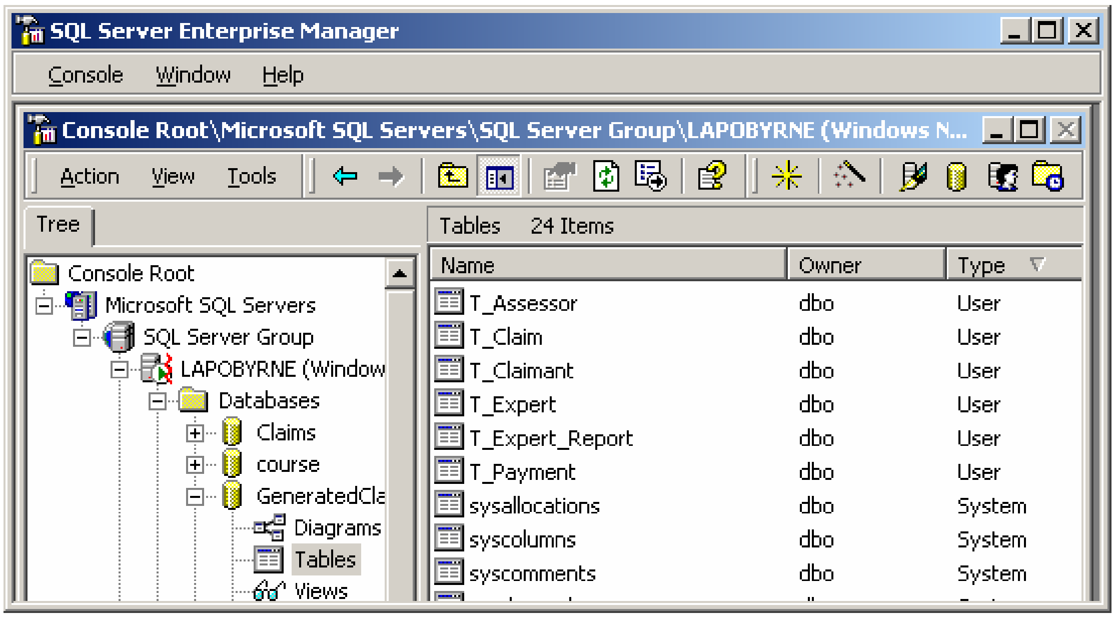
To see your data model, right click on the schema, choose data modeller, new, data model diagram)



(d) **Generate (and optionally execute) SQL.** Right click on the schema, choose data modeller, forward engineer…


Forward Engineering Wizard
Choose to Save and Execute DDL
Specify a path to save the generated DDL script. You can choose to execute the DDL script by providing your database connection information.
File name: C:\Contents\Rational Rose\Claims\claims.sql
Browse...
☑ Execute
User Name:
Test Connection
Password:
Server: lapobyrne
Database: GeneratedClaims
< Back
Next >
Cancel

This results in your tables being generated into your database.



Name	Owner	Type
T_Assessor	dbo	User
T_Claim	dbo	User
T_Claimant	dbo	User
T_Expert	dbo	User
T_Expert_Report	dbo	User
T_Payment	dbo	User
sysallocations	dbo	System
syscolumns	dbo	System
syscomments	dbo	System

Open a new data model diagram

Populate it with tables

Result

This creates a set of tables in the database.

You can view them by setting up a new data model diagram.

Right click on schema

Choose data modeler

Choose new

Choose data model diagram

Go to top Use Case Diagram Class Diagram Sequence Diagram Collaboration

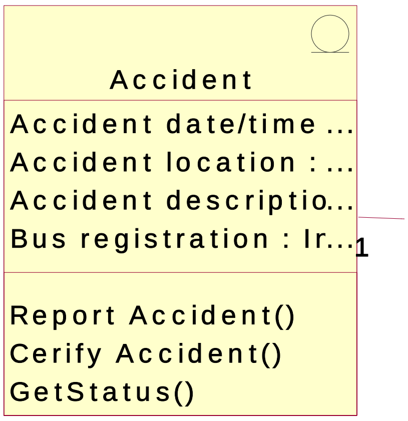
10. Generating SQL

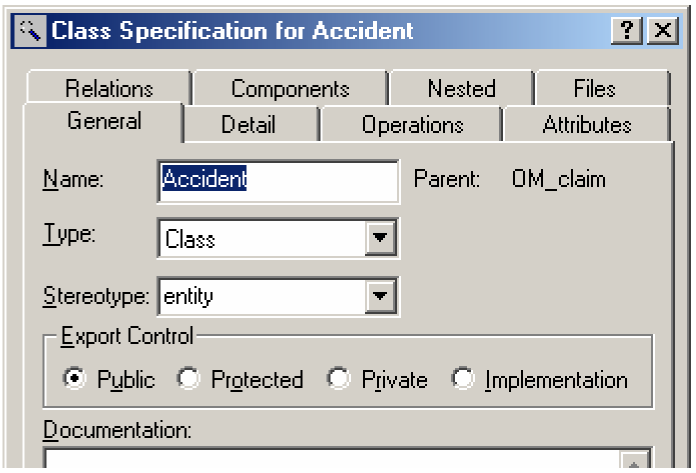
- Right click on the database
- Choose Data modeler
- Forward Engineer…

11. Generating Java Code

Before commencing code generation, it is advisable to have packaged your classes. The example shown below is from a class that is part of a package called ‘OM_Claim’.

In the class diagram window, choose a class and open its




Class Specification for Accident

Relations | Components | Nested | Files
General | Detail | Operations | Attributes

Name: Accident Parent: OM_claim

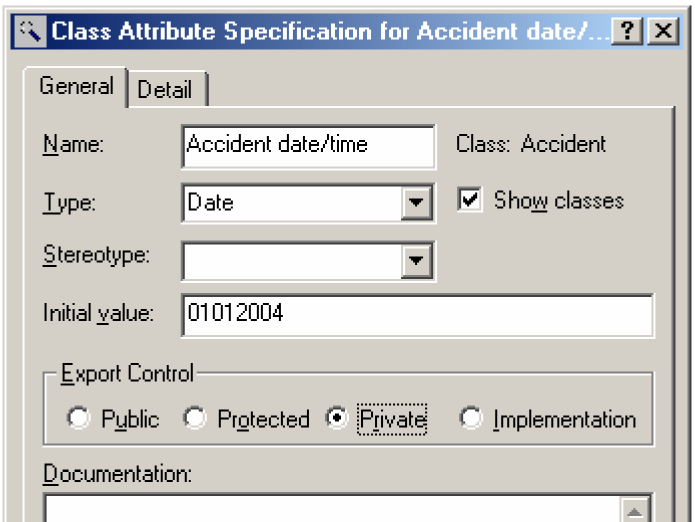
Type: Class

Stereotype: entity

Export Control: [✓ Public] Protected Private Implementation

Documentation:

specification.


Class Attribute Specification for Accident date/…

General | Detail

Name: Accident date/time Class: Accident

Type: Date [✓ Show classes]

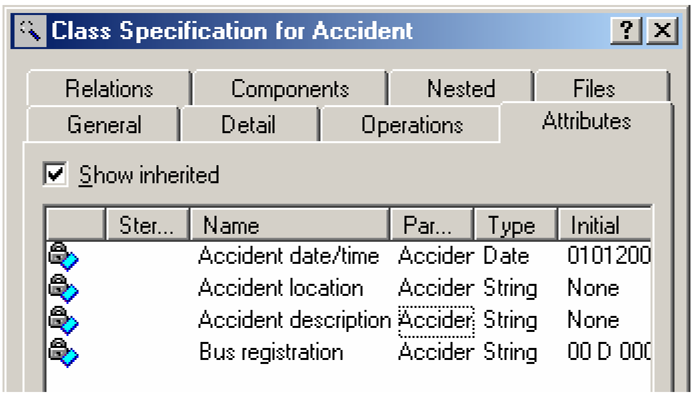
Stereotype:

Initial value: 01012004

Export Control: Public Protected [✓ Private] Implementation

Documentation:

Type in ‘This is my documentation for <*substitute your own class name*>’ Click on the attributes tab. Right click in the attributes table and choose ‘Specification’. Give the attributes the correct type, initial value and export control. Click OK to return to the class specification window. Repeat this for all attributes, ending on the class specification window.

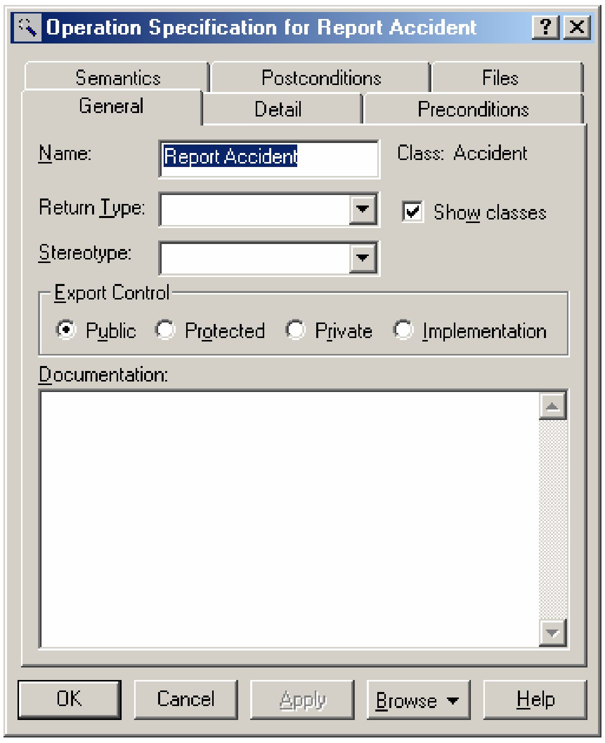

Class Specification for Accident

Relations | Components | Nested | Files
General | Detail | Operations | Attributes

[✓] Show inherited

	Ster...	Name	Par...	Type	Initial
		Accident date/time	Accider	Date	0101200
		Accident location	Accider	String	None
		Accident description	Accider	String	None
		Bus registration	Accider	String	00 D 000

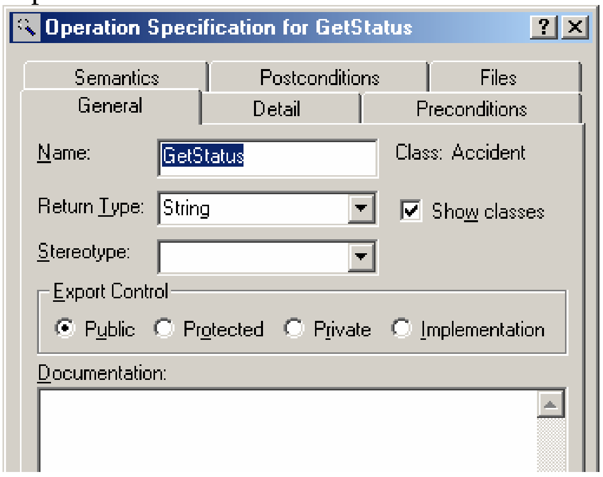
Click the Operations tab. Select one of the operations, right click and select ‘Specification’.


Operation Specification for Report Accident

Semantics | Postconditions | Files
General | Detail | Preconditions

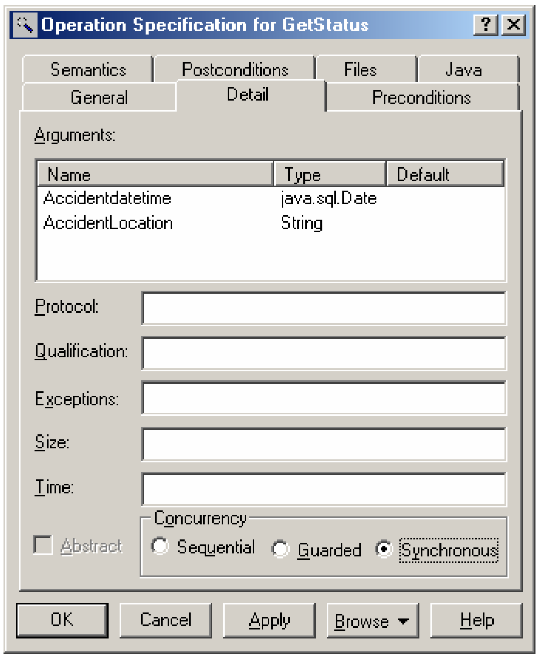
Name: Report Accident   Class: Accident
Return Type: [✓] Show classes
Stereotype:
Export Control: [● Public] ○ Protected ○ Private ○ Implementation
Documentation:

OK | Cancel | Apply | Browse | Help


Operation Specification for GetStatus

Semantics | Postconditions | Files
General | Detail | Preconditions

Name: GetStatus   Class: Accident
Return Type: String   [✓] Show classes
Stereotype:
Export Control: [● Public] ○ Protected ○ Private ○ Implementation
Documentation:


Operation Specification for GetStatus

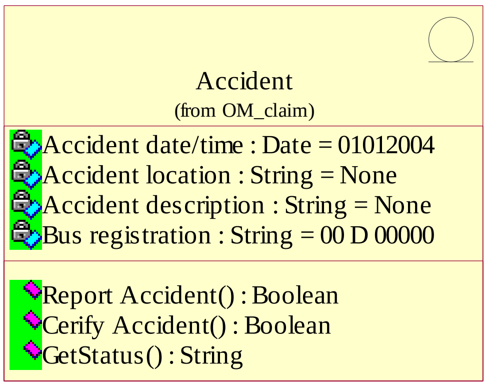
Semantics | Postconditions | Files | Java
General | Detail | Preconditions

Arguments:

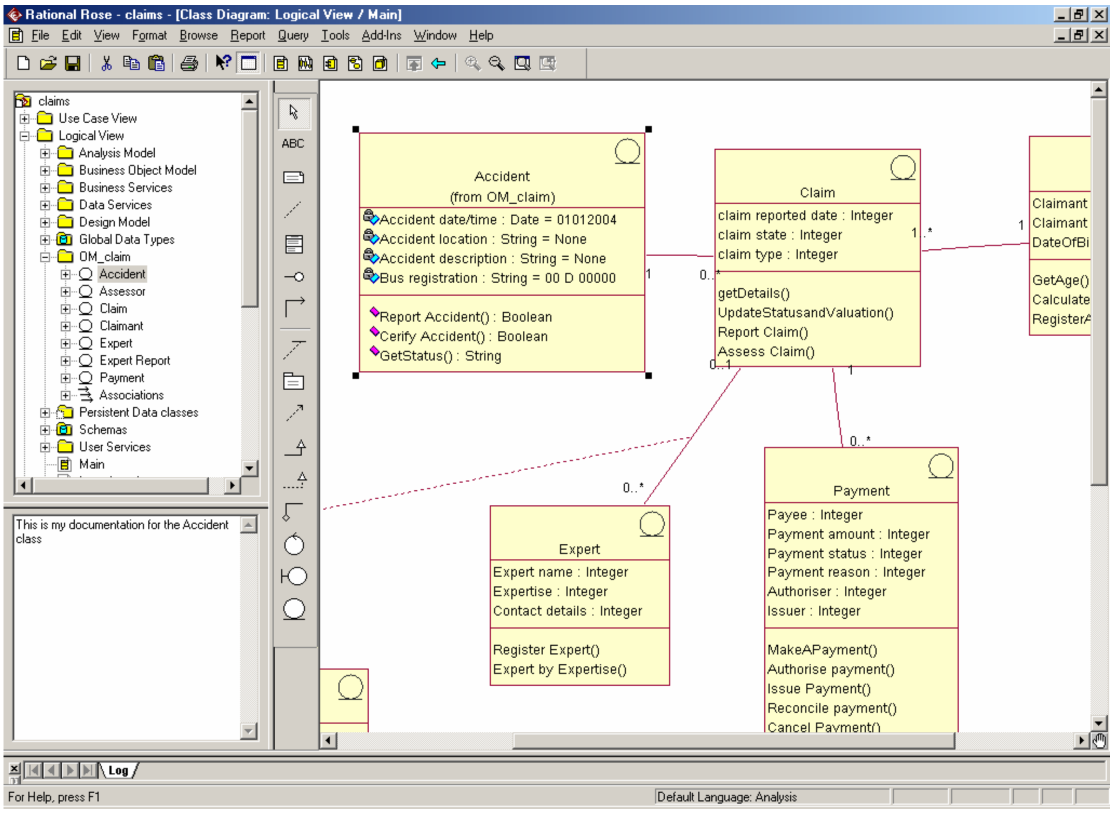
Name	Type	Default
Accidentdatetime	java.sql.Date	
AccidentLocation	String	

Protocol:
Qualification:
Exceptions:
Size:
Time:
[ ] Abstract
Concurrency: ○ Sequential ○ Guarded [● Synchronous]

OK | Cancel | Apply | Browse | Help


Accident
(from OM_claim)
Accident date/time : Date = 01012004
Accident location : String = None
Accident description : String = None
Bus registration : String = 00 D 00000
Report Accident() : Boolean
Cerify Accident() : Boolean
GetStatus() : String

You are now ready to generate the Java code for this class. Click the class. The documentation for this class is now displayed in the ‘Documentation’ panel.

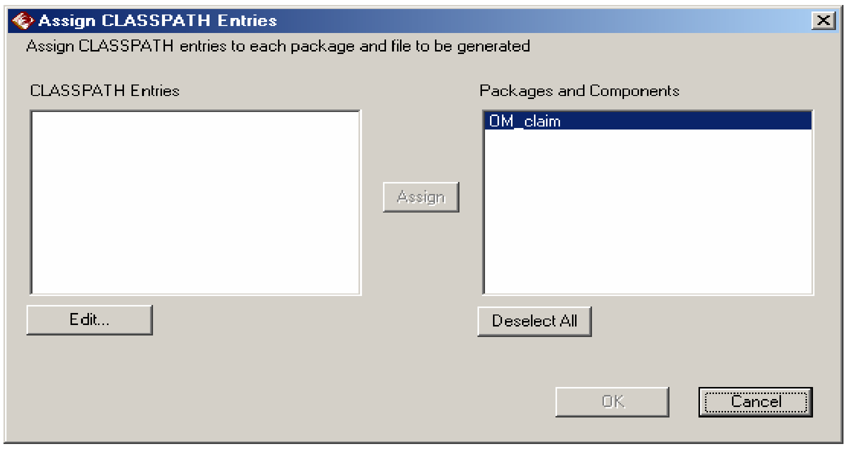


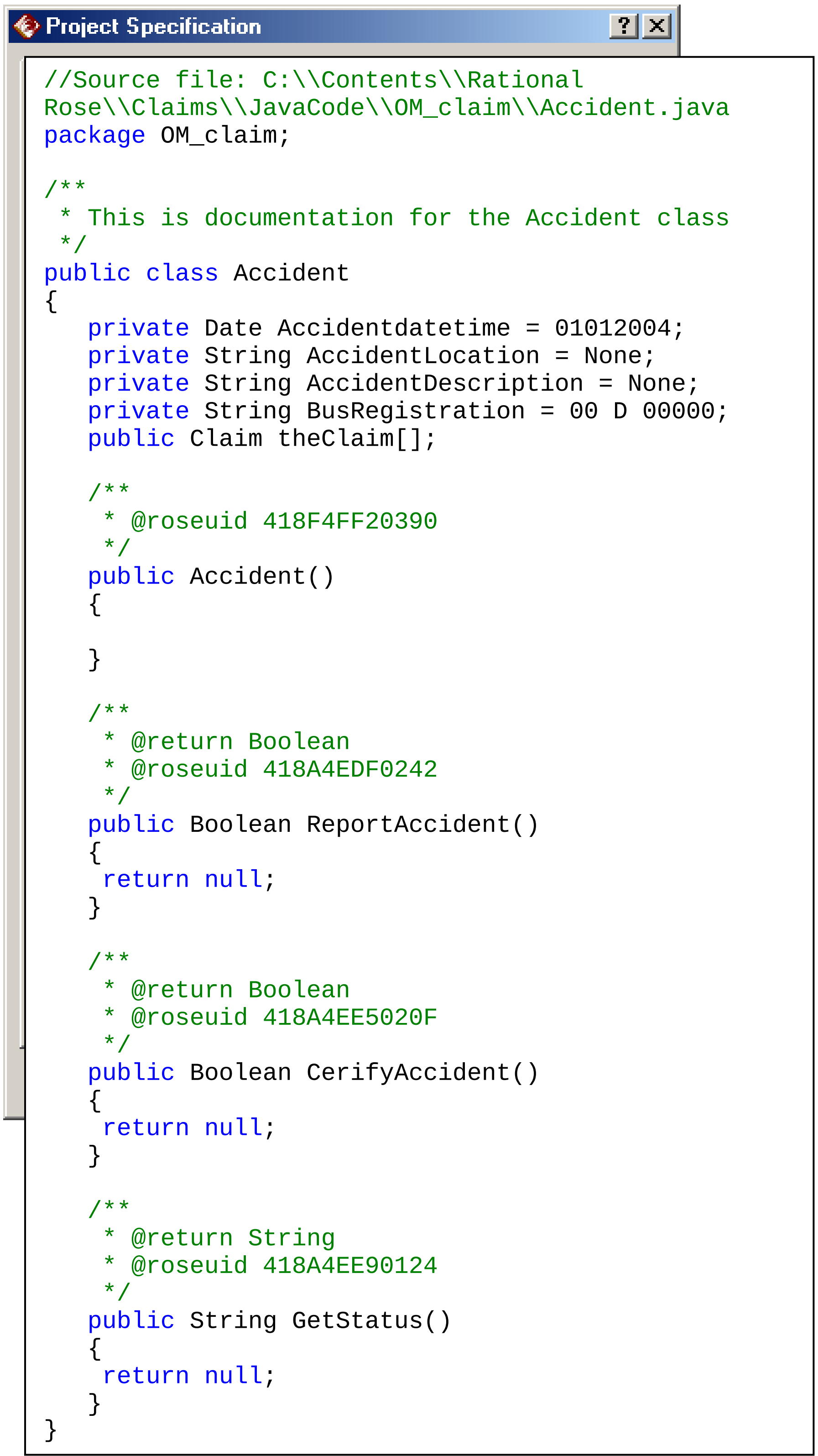
In the Tools menu, select Java / J2EE and Generate Code for the selected class. (You may get a warning that not all units are loaded. This is only significant for a completed project.).

```
10:44:57|  Starting Code Generation

10:44:57|  WARNING: Class Logical View::OM_claim::Accident - the name of attribute Accident
date/time is not a valid Java identifier.

10:44:57|  ERROR: Class Logical View::OM_claim::Accident - a name which is a valid Java
identifier cannot be constructed for attribute Accident date/time

10:45:16|  Starting Code Generation

10:45:16|  WARNING: Class Logical View::OM_claim::Accident - the name of attribute Accident
date/time is not a valid Java identifier.

10:45:16|  ERROR: Class Logical View::OM_claim::Accident - a name which is a valid Java
identifier cannot be constructed for attribute Accident date/time
```

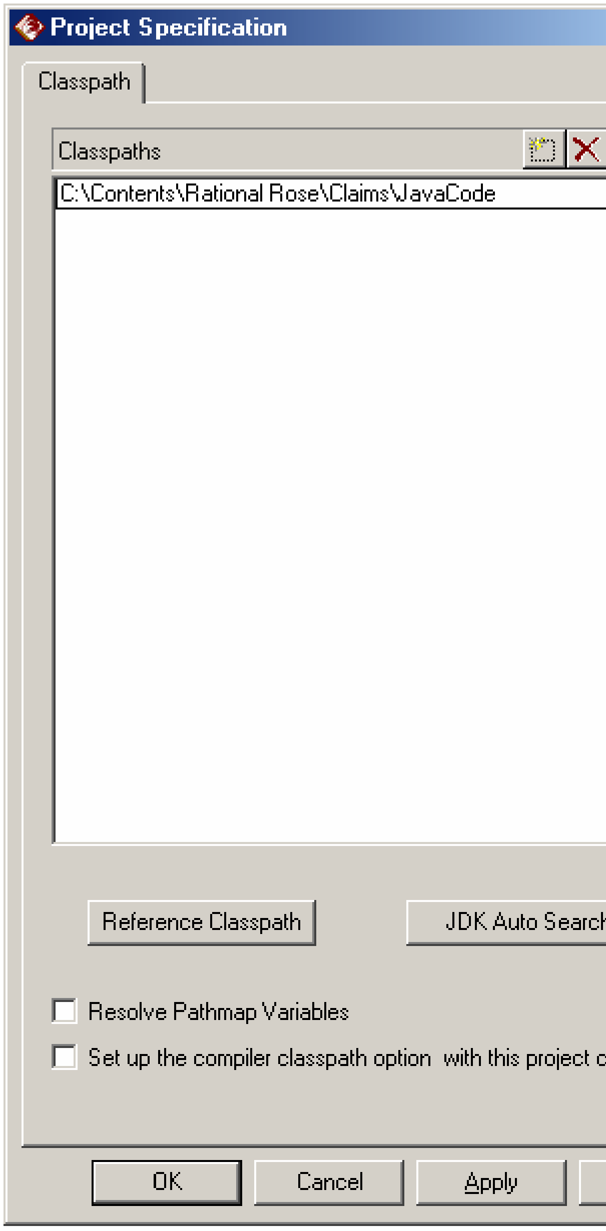
Code generation for the above yields the following errors:

Return to the class diagram and fix the problems. Generate again. The following dialogue is displayed:





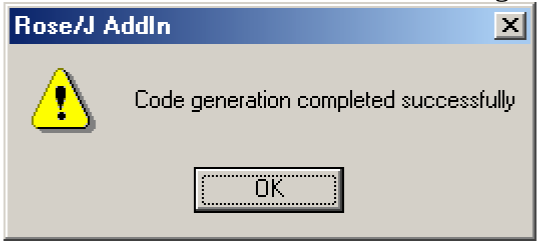
```
//Source file: C:\\Contents\\Rational
Rose\\Claims\\JavaCode\\OM_claim\\Accident.java
package OM_claim;

/**
 * This is documentation for the Accident class
 */
public class Accident
{
   private Date Accidentdatetime = 01012004;
   private String AccidentLocation = None;
   private String AccidentDescription = None;
   private String BusRegistration = 00 D 00000;
   public Claim theClaim[];

   /**
    * @roseuid 418F4FF20390
    */
   public Accident()
   {

   }

   /**
    * @return Boolean
    * @roseuid 418A4EDF0242
    */
   public Boolean ReportAccident()
   {
    return null;
   }

   /**
    * @return Boolean
    * @roseuid 418A4EE5020F
    */
   public Boolean CerifyAccident()
   {
    return null;
   }

   /**
    * @return String
    * @roseuid 418A4EE90124
    */
   public String GetStatus()
   {
    return null;
   }
}
```



C:\Contents\Rational Rose\Claims\JavaCode

Reference Classpath
JDK Auto Search
Resolve Pathmap Variables
Set up the compiler classpath option with this project c
OK
Cancel
Apply

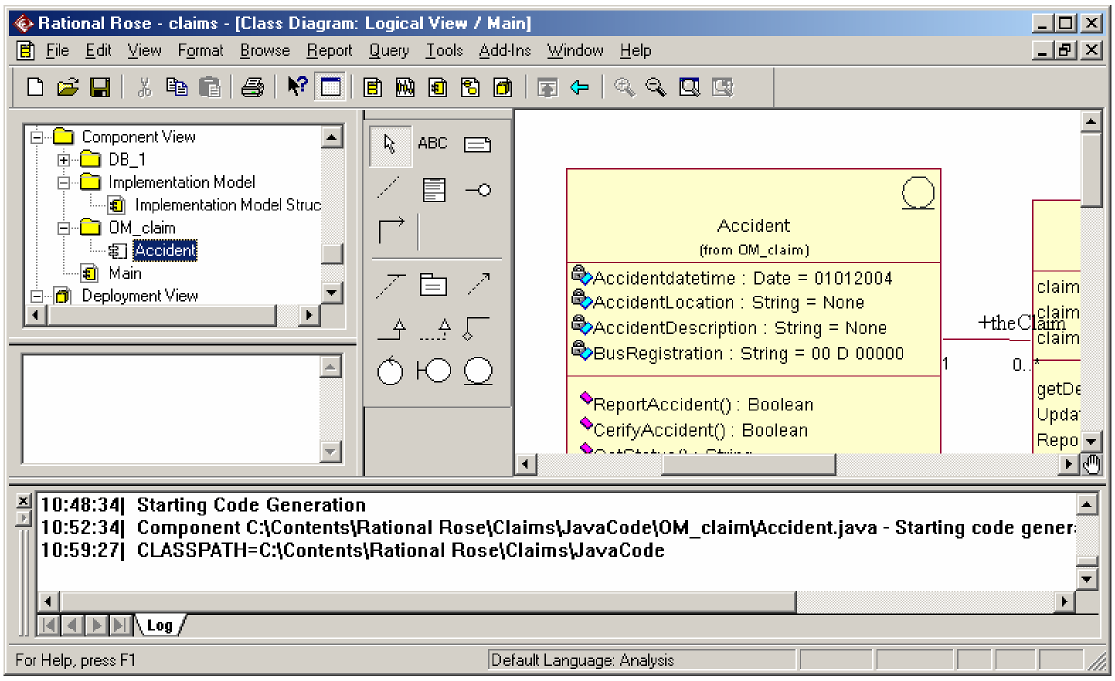
Returning to the Classpaths dialogue, click on the component and click 'Assign'. The component then disappears from the RHS of the screen. When the code generation is complete, hopefully you will get



Code generation completed successfully

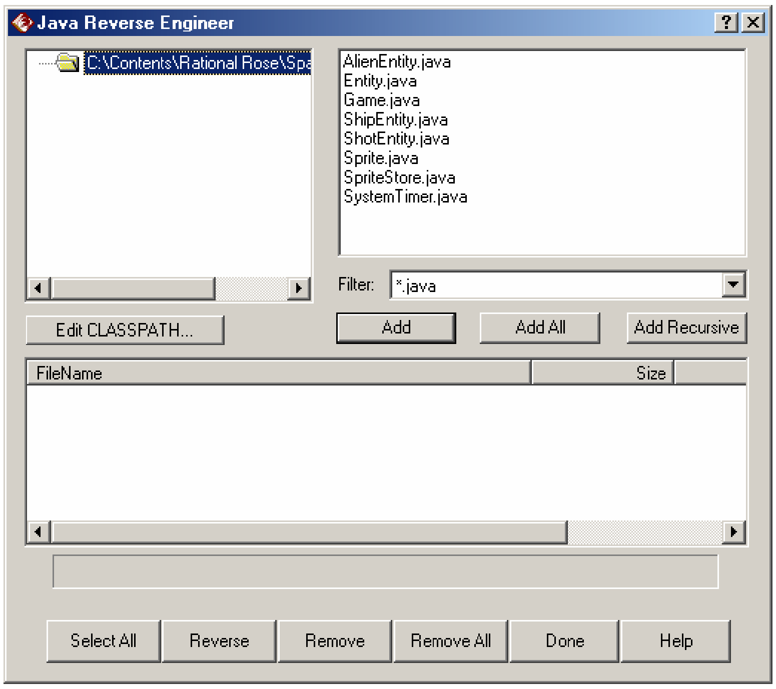
The following code has been generated into the folder: C:\Contents\Rational Rose\Claims\JavaCode\OM_claim

The Rational Rose model will produce the following log and new component:

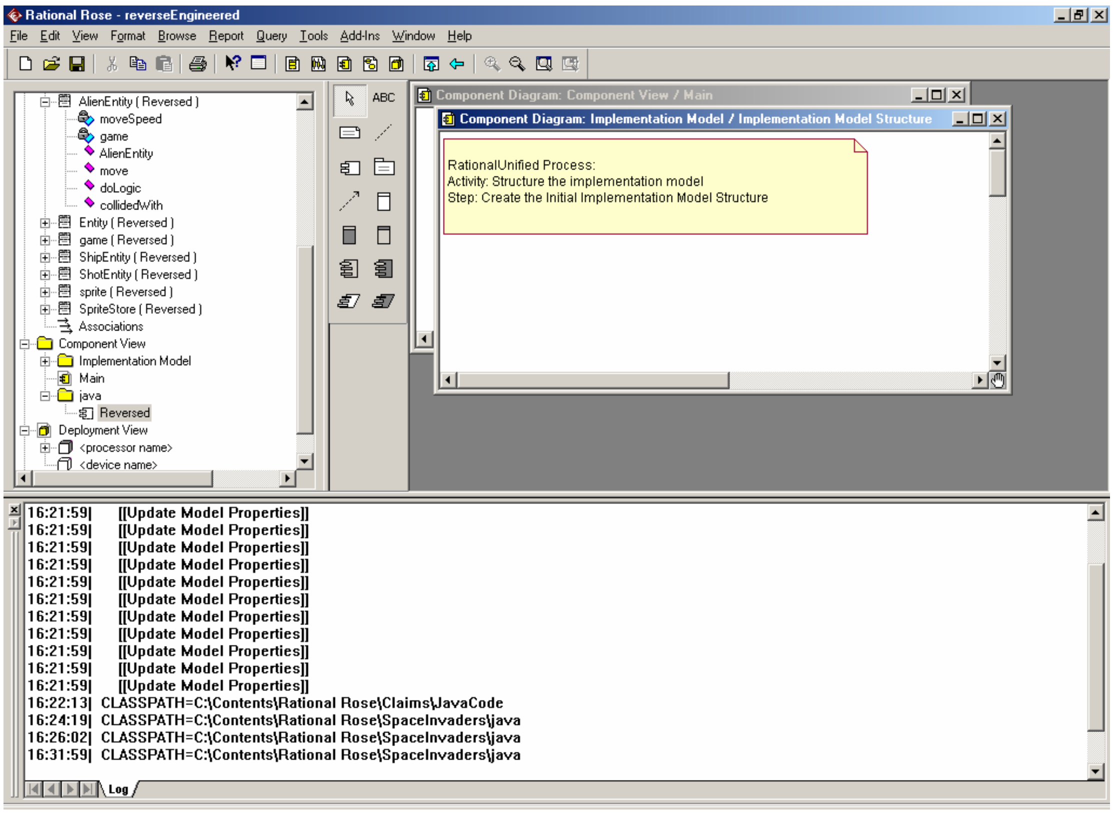

Rational Rose - claims - [Class Diagram: Logical View / Main]
File Edit View Format Browse Report Query Tools Add-Ins Window Help
Component View
DB_1
Implementation Model
Implementation Model Struc
OM_claim
Accident
Main
Deployment View
ABC
Accident
(from OM_claim)
Accidentdatetime : Date = 01012004
AccidentLocation : String = None
AccidentDescription : String = None
BusRegistration : String = 00 D 00000
ReportAccident() : Boolean
CerifyAccident() : Boolean
+theClaim
1
0..*
claim
claim
claim
getDe
Upda
Repo
Log
For Help, press F1
Default Language: Analysis

```
10:48:34|  Starting Code Generation
10:52:34|  Component C:\Contents\Rational Rose\Claims\JavaCode\OM_claim\Accident.java - Starting code gener
10:59:27|  CLASSPATH=C:\Contents\Rational Rose\Claims\JavaCode
```

12. Reverse Engineering a project



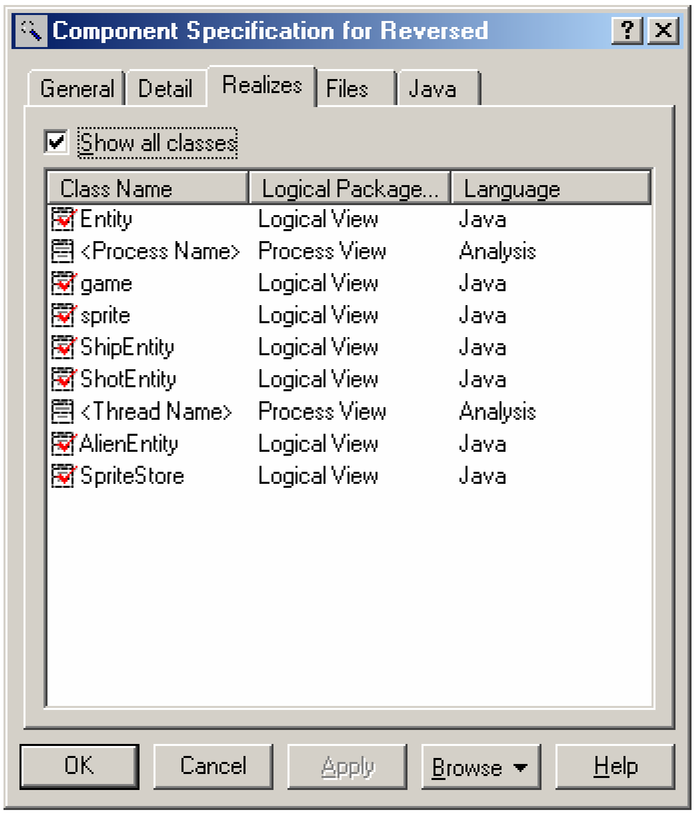
To reverse engineer the project, put all .java sources into the one directory. Create a component in the Component view. I have called the component ‘Reversed’ here. Right-click on the component and choose Java J2EE and ‘Reverse Engineer’. Specify the path where the source you want to reverse engineer is. Select each of the .java sources individually and add them. They are not added until they come up in the window. When you have finished, click ‘Done’. This will return to the project and give a report in the log window.



RationalUnified Process:
Activity: Structure the implementation model
Step: Create the Initial Implementation Model Structure

```
16:21:59|     [[Update Model Properties]]
16:21:59|     [[Update Model Properties]]
16:21:59|     [[Update Model Properties]]
16:21:59|     [[Update Model Properties]]
16:21:59|     [[Update Model Properties]]
16:21:59|     [[Update Model Properties]]
16:21:59|     [[Update Model Properties]]
16:21:59|     [[Update Model Properties]]
16:21:59|     [[Update Model Properties]]
16:21:59|     [[Update Model Properties]]
16:21:59|     [[Update Model Properties]]
16:22:13|  CLASSPATH=C:\Contents\Rational Rose\Claims\JavaCode
16:24:19|  CLASSPATH=C:\Contents\Rational Rose\SpaceInvaders\java
16:26:02|  CLASSPATH=C:\Contents\Rational Rose\SpaceInvaders\java
16:31:59|  CLASSPATH=C:\Contents\Rational Rose\SpaceInvaders\java
```

When this has been completed, open the specification for the new component (‘Reversed’ in the illustration). The following is shown. Right click on each class and choose ‘Assign’.


Component Specification for Reversed

☑ Show all classes

Class Name	Logical Package...	Language
Entity	Logical View	Java
<Process Name>	Process View	Analysis
game	Logical View	Java
sprite	Logical View	Java
ShipEntity	Logical View	Java
ShotEntity	Logical View	Java
<Thread Name>	Process View	Analysis
AlienEntity	Logical View	Java
SpriteStore	Logical View	Java

13. Customising your model

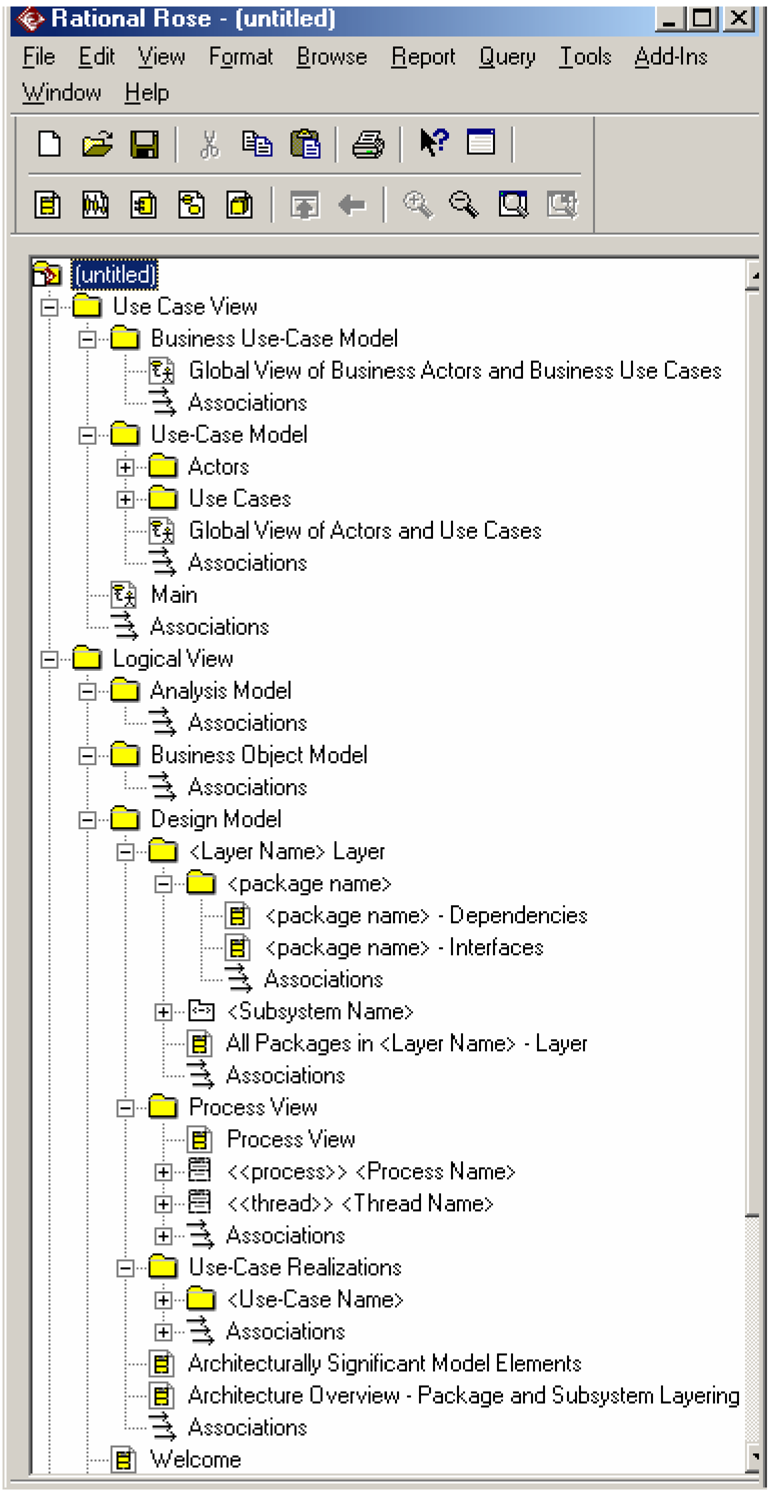
When you start up a new model in Rational Rose, using the Rational Unified Process template, there are many packages and folders that have been set up for your use. These can be extremely confusing.

When starting a model for a simple system, it may not be necessary to model the Business domain. If not, you may delete the Business Use-Case Model from the Use Case View, leaving only the Use-Case model.

Be aware that the tool will allow you to insert almost any icon in any folder. Try to keep to the guidelines – i.e. put actors into the Use Case View, Use Case Model, Actors folder.

The number of folders in the Logical view will depend on the options that you have taken in the Diagram tab in the Options menu. If you have selected the 'Three-tier diagram', Rational Rose provides folders to allow you to manage the different layers and also to allow you to evolve the model from the business domain through to implementation. Once again, for a simple system, all of these folders may not be necessary. Take a note of the folders that are offered and try cutting them down, unless you know the meaning of the folders. Always use appropriate folder names to denote and classify classes, objects and diagrams.

When it comes to generation, the folder denotes the package to which the class belongs. Ensure that it is appropriately named and that each package folder's contents are appropriate to the task that will be done on the folder.


Rational Rose - (untitled)
File Edit View Format Browse Report Query Tools Add-Ins
Window Help

```
(untitled)
  Use Case View
    Business Use-Case Model
      Global View of Business Actors and Business Use Cases
      Associations
    Use-Case Model
      Actors
      Use Cases
      Global View of Actors and Use Cases
      Associations
    Main
    Associations
  Logical View
    Analysis Model
      Associations
    Business Object Model
      Associations
    Design Model
      <Layer Name> Layer
        <package name>
          <package name> - Dependencies
          <package name> - Interfaces
          Associations
        <Subsystem Name>
        All Packages in <Layer Name> - Layer
        Associations
      Process View
        Process View
        <<process>> <Process Name>
        <<thread>> <Thread Name>
        Associations
      Use-Case Realizations
        <Use-Case Name>
        Associations
      Architecturally Significant Model Elements
      Architecture Overview - Package and Subsystem Layering
      Associations
    Welcome
```

To delete a folder, select it and use CTRL-D.

14. Troubleshooting

1) I add a business class that has the same name as an actor and the drawing tool shows me the actor:

   Before naming the class, open its specification. Change the stereotype to business entity, and then rename it to the actor name. You will be warned that the name appears in multiple domains, but that is OK.

2) When I add my class, it doesn't look like the one in the sample.

When you add a class, the stereotype may default to none. This will give the normal 3-compartment box display. When you change the stereotype, this can change the way in which the class is displayed. To change the display, right click on the class and choose stereotype. If you want the 3-box compartment, choose 'none'.