### **RATIONAL ROSE**

1.	STARTING RATIONAL ROSE	.2
2.	CREATING A PROJECT MODEL IN RATIONAL ROSE	.2
3.	USE CASE DIAGRAM	.6
4.	CLASS DIAGRAM	.9
5.	ACTIVITY DIAGRAM1	6
6.	SEQUENCE DIAGRAM1	17
7.	COLLABORATION DIAGRAMS	21
8.	DRAWING A STATE DIAGRAM2	23
9.	GENERATING TABLES FROM CLASSES	23
10.	GENERATING SQL	26
11.	GENERATING JAVA CODE2	26
12.	REVERSE ENGINEERING A PROJECT	30
13.	CUSTOMISING YOUR MODEL	33
14.	TROUBLESHOOTING	34

# 1. Starting Rational Rose

<u>Top of the Document</u> <u>Use Case Diagram</u> <u>ClassDiagram</u> <u>ActivityDiagram</u> 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)		
<u>File E</u> dit <u>V</u> iew F <u>o</u> rmat <u>B</u> rowse !	<u>R</u> eport Query <u>T</u> ools <u>A</u> dd-Ins <u>W</u> indow <u>H</u> elp	
🗅 🖻 🖬 🛛 X 🖻 🛍 🎒	🕅 🗖   🖻 🐘 🗃 🖏 🖬   🗊 🖛   🍕 🔍 🛄 🖻	<b>1</b>
Image: Second state of the second	ABC     Image: Class Diagram: Logical View / Main     ABC     Image: Class Diagram: Logical View / Main     Image: Class Diagram: Logical View / Main     Image: Class Diagram: Logical View / Main     Image: Class Diagram: Logical View / Main	
12:07:27  [[Update Mod 12:07:27] [[Update Mod 12:07:27] [[Update Mod 12:07:27] [[Update Mod 12:07:27] [[Update Mod 12:07:27] [[Update Mod 12:07:27] [[Update Mod	el Properties]] el Properties]] el Properties]] el Properties]] el Properties]] el Properties]]	

4. Using the <u>V</u>iew 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

<b>WR</b> Options	? ×
Ada83     Ada95     CORBA       COM     VC++       General     Diagram     Brow       Save       □     Use temporary file       ☑     Create backup file	Java Oracle8 C++ MSVC Visual Basic XML_DTD wser Notation Toolbars ANSI C++ Default font
<u>Keep two backup files</u> Update by copy     Auto save every	Arial, 10, points Documentation window font Font
Miscellaneous           Image: Save settings on exit           Image: Save setting settings on exit <tr< td=""><td>MS Sans Serif, 8, points Log window font Fon<u>t</u></td></tr<>	MS Sans Serif, 8, points Log window font Fon <u>t</u>
Default color	System, 10, points
Layout Options Spacing between shapes	Unconnected shapes
Horizontal: 150	✓ Align <u>N</u> umber per row: 10
OK	Cancel <u>A</u> pply Help

#### 7. Diagram as follows:

Coptions	? ×
Ada83 Ada95 CORBA COM VC++	Java Oracle8 C++ MSVC Visual Basic XML_DTD
General     Diagram     Brows       Compartments     □     Show visibility       Image: Show stereotypes     □     Show stereotypes       Image: Show operation signatures     Image: Show all attributes     Image: Show all operations       Image: Show all operations     Image: Show all operations     Image: Suppress attributes       Image: Suppress operations     Image: Suppress operations       Image: Message Signatures     Image: Suppress operations <t< td=""><td>er Notation Toolbars ANSI C++ Display Unresolved adornments Unit adornments Collaboration numbering Sequence numbering Ecollaboration Messages Feocus of control Three-Tier Diagram O Name and Type O None Grid</td></t<>	er Notation Toolbars ANSI C++ Display Unresolved adornments Unit adornments Collaboration numbering Sequence numbering Ecollaboration Messages Feocus of control Three-Tier Diagram O Name and Type O None Grid
<ul> <li>✓ Double-click to diagram</li> <li>✓ Automatic resizing</li> <li>✓ Class Name Completion</li> <li>✓ Aggregation whole to part</li> <li>Stereotype display</li> <li>○ None</li> <li>○ Label</li> <li>✓ Show labels on relations and</li> </ul>	Image: Signaputo grid         Grid size: 5         Role Display         Image: Show role specifier         O Decoration         Image: Bassociations
ОК	Cancel <u>A</u> pply 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								_ 🗆	×
<u>File E</u> dit <u>V</u> iew Format Browse	<u>R</u> eport	Query	<u>T</u> ools	<u>A</u> dd-Ins	<u>W</u> indow	<u>H</u> elp			
🗋 🗅 😅 🖬 🛛 X 🖻 💼 🖨	<b>N?</b>		) 🕅	1	0   🗐	+	ର୍ ବ୍		
Sample1 Use Case View Logical View Deployment View Model Properties			s Diag	ram: Log	ical ¥iew	/ Main			
For Help, press F1			efault l	.anguage:	Java			<u>•</u>	

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.

			ne fuet that the delive diagram is a ese cuse diagram.							
🚸 Rational Rose - Sample1 - [Use C	ase Diag	ram: Use	Case	: View /	Main]					
🔁 File Edit View Format Browse	<u>R</u> eport	Query ]	ools	<u>A</u> dd-Ins	; <u>W</u> indow	Help			2 ×	
	N? 🗖	E1 (M)		5 0	)   📻 🔸	-   🕾	9		<b>E</b>	
Sample1 Use Case View Aain Component View Deployment View Model Properties	k m = / = 0 + t / 2 - 2 - 2 - 2 - 2 - 2									
For Help, press F1 Default Language: Java										

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

🗞 Rational Rose - Sample1 - [Use Case Diagram: Use Case View / Main]							
🔁 File Edit View Format Brows	e <u>R</u> eport	: <u>Q</u> uery	<u>T</u> ools	<u>A</u> dd-Ins	<u>W</u> indow	Help	_ 8 ×
0 🖻 🖬 🕹 🛍 🖨	№ □	]   🖪	•	8 🖻	☶ +		. 🖪 🖻
Sample1 Use Case View Main NewUseCase Logical View Component View Deployment View Model Properties		NewUse	Case				
For Help, press F1		Def	ault Lang	juage: Jav	a		

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:

Use Case Specification for NewUseCase
General Diagrams Relations Files
Name: NewUseCase Package: Use Case View
Stereotype:
Bank:
Documentation:
<u> </u>
OK Cancel <u>Apply</u> <u>B</u> rowse ▼ <u>H</u> elp

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' ito 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

Open	?×	
Look jn: 🔁 UML 💽 🖛 🛍 🗰 🕇		ħ
Sample1	aı di	ui is
File <u>n</u> ame: Sample1		
Files of type: Models (*.mdl)		

your model.

This screen is automatically displayed:



3. In the browser window, Click on the 'Logical View' folder, right click; click



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       Generate         Visibility       abstract       static         public       final       Instance Initialize         Interface       Officients       Default Construct         Generate Code       Disable Autosync       Reference	or
Constructor Visibility public	
Extends X + +	<u>↓</u>
DocComment	
	A Y
OK Cancel Apply H	elp

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-

🚸 Rational Rose - Sample1		_ @ ×
Eile Edit View Format Browse Report	Query Iools Add-Ins Window Help	
🗋 🗅 🚅 🔛 👗 🖻 💼 🎒 🧩		
Sample1 Use Case View Use Case View Use Case View Conditions Component View Conditions Co	Claim Accident	
For Help, press F1	Default Language: Java	

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.

🥰 Class Specifi	cation for Clair	n		<u>? ×</u>
Relations General	Components   Detail	Nested   Operations	Files A	Java   ttributes
☑ <u>S</u> how inher	ited			
Ster	Name	Par	Туре	Initial
•				
ОК	Cancel App	ply <u>B</u> rov	vse 🔻	<u>H</u> elp

Relations Components Nested Files Java
Constal Datai Occasione Attributes
General Detail Operations Attributes
Show inherited
Ster Name Par Type Initial
OK Cancel <u>Apply</u> <u>Browse</u> <u>Help</u>
Oouble click on the attribute to add more detail:
Class Attribute Specification for Claim Date
General Detail Java
Name: Claim Date Class: Claim
Iype:
Stereotype:
Initial value:
- Funct Control
C Public C Protected  Private C Implementation
Locumentation.
<b>v</b>

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

Operations are added using the operations tab:

🔍 Class Spec	ification for C	laim	<u>? ×</u>
Relations General	Components Detail	Nested File	es Java   Attributes
💌 <u>S</u> how inh	erited		
Ster.	Operation	Return type	Parent
ľ	opname		Cialiti
OK	Cancel		· ▼ <u>H</u> elp

Operations can be further specified:

🔦 Operation S	pecification for Make	a claim	? ×
Semantics General	Postconditions Detail	Files Preco	Java nditions
<u>N</u> ame:	Make a claim	Class: C	laim
Return <u>T</u> ype:	•	🛛 🔽 Sho	<u>w</u> classes
Stereotype:	•	·]	
Export Contro	bl	-	
• P <u>u</u> blic •	Protected O Private	e O <u>I</u> mple	mentation
Documentation	1:		
			-
OK	Cancel Apply	<u>B</u> rowse ▼	<u>H</u> elp

### Adding other associations

### Generalisation

To add a generalisation, click on the Generalisation icon  $\square$ . 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	boc	ly	•	limb
General Detail Role A General Role B General Role A Detail			•	
Bole:  Element: body    Constraints				
Multiplicity: 💽 🔽 Navigable				
✓ Aggregate     Static     Friend       Containment of limb       ● By Yalue     ● By Reference     Unspecified				
Keys/Qualifiers				
Name Type				
OK Cancel Apply Browse - Help				

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 <u>section 1</u>.

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.

 $\diamond$  This is the decision diamond.

 $\Box$  This is the activity.

 $\Box$  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

Corder 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.



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  $\rightarrow$  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 rightarrow 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  $\times$  icon on its lifeline.



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

Kessage Specification for PickGarment()	? X
General Detail	
Synchronization	
© Simple	
© Synchronous	
◯ <u>B</u> alking	
© <u>⊺</u> imeout	
C Procedure Call	
C Asynchronous	
⊙ <u>B</u> eturn	
Frequency Aperiodi <u>c</u> <u>P</u> eriodic	
OK Cancel Apply Browse -	<u>H</u> elp

'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

Coperation Specification for opname	Ressage Specification for Get design( )	<u>?</u> ×
Semantics       Postconditions       Files         General       Detail       Preconditions         Name:       Get details       Class: Order Screen         Return Iype:       Boolean       ✓         Stereotype:       ✓       Show classes         Export Control       ✓       Public       Protected         Ocumentation:       ✓       Implementation	General       Detail         Synchronization         Synchronous         galking         Imeout         Procedure Call         Asynchronous         Return	
UK Cancel <u>Apply</u> <u>B</u> rowse ▼ <u>H</u> elp	OK Cancel <u>Apply</u> <u>B</u> rowse ▼ <u>H</u>	elp

their stereotypes.

🚸 Rational Rose - 2003Garments - [Coll	aboration Diagram: Order Raw Materials / NewDiagram2]	
Eile Edit View Format Browse Rep	ort <u>T</u> ools <u>A</u> dd-Ins <u>W</u> indow <u>H</u> elp	-8×
🔄 🗅 🚅 🔚 🕺 🖻 💼 🎒 🦃		
Warehouse     Welcome     Main     Realised Use Cases     AddRetailer     Order Raw Materials     NewDiagram2     Register new order     MAdd Retailer     M Add Retailer     M Add Retailer     M Add Retailer     M Register new order     M Register new order     M 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

 $\checkmark$ , but there can be many messages passing between them. The messages are directional, so use either  $\checkmark$  or  $\checkmark$ .

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.



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).

Open the specification for the database and associate it with

Catabase Specification for claims	X
General	
Name: claims	
Target: Microsoft SQL Server 7.x	
Comment:	

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, tick 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.

(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.

Class Specifi	cation for Ex	pert		?
Relations	Componer	nts	Nested	Files
General	Detail	Ope	rations	Attributes
<u>M</u> ultiplicity:	n	•	]	
<u>S</u> pace:				
Persistence -			ncurrency-	
• Persiste <u>n</u> t		•	Sequentia	I
C Transient		C	Guarded	
On Schema	• Specification	for Clain	is schema	×
General				

General	
<u>N</u> ame:	Claims schema
<u>D</u> atabase	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 Engineer	ing Wizard
<b>Choose to Save</b> Specify a path t the DDL script t	and Execute DDL to save the generated DDL script. You can choose to execute by providing your database connection information.
<u>F</u> ile name:	:\Contents\Rational Rose\Claims\claims.sql
Evecute	Browse
	Test Connection
<u>D</u> ser Name. Password:	
S <u>e</u> rver:	
<u>D</u> atabase:	
	< <u>B</u> ack <u>N</u> ext > Cancel

This results in your tables being generated into your database.

🚡 SQL Server Enterprise Manager			
<u>C</u> onsole <u>W</u> indow <u>H</u> elp			
🚡 Console Root\Microsoft SQL Serv	vers\SQL Server Group\LAPOBYR	NE (Windows N	
Action ⊻iew Tools ← → _	🗈 🔳 💣 🖗 🗟 🛛 🔆	k   🎊   👂	0 🖸 🔁
Tree	Tables 24 Items		
Console Root	Name	Owner	Туре 🗸
📄 🗐 Microsoft SQL Servers	T_Assessor	dbo	User
🗐 🗐 SQL Server Group	T_Claim	dbo	User
🗐 🕂 🔂 LAPOBYRNE (Window	T_Claimant	dbo	User
📄 🧰 Databases	T_Expert	dbo	User
📔 🗄 🗄 🔰 Claims	T_Expert_Report	dbo	User
🗄 🕀 🛄 course	T_Payment	dbo	User
GeneratedCla	📰 sysallocations	dbo	System
Diagrams	📰 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'.

		C	choose a clas	s and open its	5	
	$\bigcirc$	9	Class Specifi	cation for Accid	lent	<u>?</u> ×
Accid	ent		Relations	Components	Nested	Files
<mark>Accident da</mark>	te/time		General	Detail 0	perations	Attributes
Accident loc	ation :		<u>N</u> ame: Acc	ident	Parent: 0	)M_claim
Accident de Bus registra	tion : In		 	s	<b>-</b>	
			Starashunar Janti			
Report Accie	dent()		Export Control	y	<u> </u>	
Cerify Accid	ent()		• Public O	Protected C P	rivate O Imp	lementation
			Documentation:			
ecification.						
Class Attribute Specifi	cation for Acci	dent date/ <b>?</b> 🗙	Type in	This is my	1	
General Detail			vour owi	itation for <i><si< i=""> 1 class name&gt;</si<></i>	·' Click	
Name: Accident dat	e/time Cl	lass: Accident	on the at	tributes tab. 1	Right	
Tupe: Date			click in t	he attributes t	able and	
Type.  Date		- 5110 <u>-1</u> 6103363	the attrib	specification outes the corre	ct type.	
	<b>•</b>		initial va	lue and export	t	
Initial value: 01012004						
Export Control						
O P <u>u</u> blic O Pr <u>o</u> tected	I 🖲 <u>Private</u> C	D Implementation				
Documentation:						

-

In the class diagram window,

control. Click OK to return to the class specification window. Repeat this for all attributes, ending on the class specification window.

🔦 Class	: Specil	ication for A	ccide	nt		? ×
Rela	itions	Componer	nts	Neste	ed	Files
Gen	eral 🗍	Detail	Op	erations	A	ttributes
🔽 Sh	ow inheri	ted				
	Ster	Name		Par	Туре	Initial
	Ster	Name Accident date	e/time	Par Accider	Type Date	Initial 0101200
	Ster	Name Accident date Accident loca	e/time ition	Par Accider Accider	Type Date String	Initial 0101200 None
4242	Ster	Name Accident date Accident loca Accident dese	e/time ition	Par Accider Accider Accider	Type Date String String	Initial 0101200 None None

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

Semantics Postconditions Files   General Detail Preconditions   Name: Peport Accident Class: Accident   Return Lype: Image: Im	Operation Specification for Benort Accident	2 X	Operation Speci	fication for Gets	itatus
General       Detail       Preconditions         Mame:       Report Accident       Class: Accident         Return Lype:       ▼ Shogy classes       Stereotype:       ▼         Export Control       ♥ Lybic       ♥ Pigtected       ♥ Pigtectectectectectectectectectectectectect	Computing Destroyability File		Semantics	Postconditio	ins Files
Name:       Declar       Class: Accident         Return Type:       Image: String       Image: String       Image: String         Stereotype:       Image: String       Image: String       Image: String         Export Control       Pyblic       Protected C       Private       Implementation         Documentation:       Image: String       Image: String       Image: String       Image: String       Image: String         DK       Cancel       Apply       Browse ▼       Help       Image: String       Image: String </td <td>General Detail Precondition</td> <td>S  </td> <td>General</td> <td>Detail</td> <td>Preconditions</td>	General Detail Precondition	S	General	Detail	Preconditions
Return Lype: Image: String   Stereotype: Image: String   Export Control Pyblic C Protected C Private C Implementation   Documentation: Image: String   DK Cancel Apply Browse Image: Help   Accident (from OM_claim) Accident date/time : Date = 01012004   Accident description : String = None   Bus registration : String = 00 D 00000   Report Accident() : Boolean   Certify Accident() : Boolean	Name: Report Accident Class: Accide	ent	Name: GetS	tatus	Class: Accident
Stereotype:     Export Control     Public     Public <th>Return <u>T</u>ype:</th> <th>isses</th> <th>Return <u>T</u>ype: String</th> <th></th> <th>✓ Show classe</th>	Return <u>T</u> ype:	isses	Return <u>T</u> ype: String		✓ Show classe
Export Control  Public Protected Private Implementation  Documentation:  Documentation:  Concel Apply Browse  Help  Accident (fiom OM_claim)  Accident tate = 01012004 Accident tate = 01012004 Accident date/time : Data Concel Detail  Protocol  Pro	Stereotype:				<u>-</u>
Public © Protected © Private © Implementation          Documentation:         Discrete       Operation Specification for GetStatus         Provide C Protected © Private © Implementation         Discrete       Operation Specification for GetStatus         Provide C Protected © Private © Implementation         Discrete       Operation Specification for GetStatus         Protected © Protected			- Export Control		
Documentation:         □	Public C Protected C Private C Implement	ation		otected (O Prival	te 🔍 Implementatio
OK Cancel   Apply Browse   Help     Accident   (from OM_claim)     Accident date/time : Date = 01012004   Accident location : String = None   Accident location : String = None   Accident description : String = None   Bus registration : String = 00 D 00000     Report Accident() : Boolean   Certify Accident() : Boolean   Certify Accident() : Boolean   Certify Accident() : Boolean	Documentation:		Documentation:		
OK Cancel Apply Browse Help     Name Type Default     Accident java.sql.Date     Accident discription: String     Protocol:     Qualification:     Qualification:     String = None     Accident description:     String = None     Accident description:     String = None     Bus registration:     String = 00 D 00000     Image: Concurrency     Cerify Accident():   Boolean   Cerify Accident():   Boolean   Cerify Accident():   Boolean   Cerify Accident():   Boolean   Cerify Accident():     String		V	Contraction Sp Semantics General	Postconditions	tStatus Java   Files Java   Preconditions
OK       Lancer       Apply       Browse < Help         Image: Accident Location       java sql.Date         Accident Location       String         Accident date/time : Date = 01012004         Accident location : String = None         Accident description : String = None         Accident description : String = None         Bus registration : String = 00 D 00000         Report Accident() : Boolean         Cerify Accident() : Boolean         Cerify Accident() : Boolean         Cerify Accident() : String			Name	Tur	Default
Accident (from OM_claim)  Accident date/time : Date = 01012004  Accident location : String = None  Accident location : String = None  Accident description : String = None Bus registration : String = 00 D 00000  Image: Concurrency Con	UK Cancel Apply Browse •		Accidentdatetir	ne java	a.sql.Date
Accident (from OM_claim)  Accident date/time : Date = 01012004 Accident location : String = None Accident location : String = None Bus registration : String = 00 D 00000  Report Accident() : Boolean Cerify Accident() : Boolean Cerify Accident() : Boolean Cerify Accident() : String			AccidentLocati	on Strin	ng
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   Cerify Accident() : Boolean   Cerify Accident() : String					
(from OM_claim)       Qualification:         Accident date/time : Date = 01012004       Exceptions:         Accident location : String = None       Size:         Bus registration : String = 00 D 00000       Iime:         Report Accident() : Boolean       Concurrency         Cerify Accident() : Boolean       Sequential © Guarded © Synchronou         GetStatus() : String       String	Accident		Protocol:		
Accident date/time : Date = 01012004   Accident location : String = None   Accident description : String = None   Bus registration : String = 00 D 00000     Ime:   Cerify Accident() : Boolean   Cerify Accident() : Boolean   Certify Accident() : Boolean   CetStatus() : String	(from OM_claim)		Qualification:		
Accident description : String = None         Bus registration : String = 00 D 00000         Report Accident() : Boolean         Cerify Accident() : Boolean         GetStatus() : String	$\mathbf{P}_{\mathbf{A}}$ Accident date/time : Date = 01012004		Exceptions:		
Bus registration : String = 00 D 00000 Report Accident() : Boolean Cerify Accident() : Boolean CetStatus() : String	Accident description : String = None		Size:		
Report Accident(): Boolean       Concurrency         Cerify Accident(): Boolean       Abstract         CetStatus(): String       Sequential © Guarded © Synchronou	Bus registration : String = $00 \text{ D} 00000$		Time		
Cerify Accident(): Boolean Cerify Accident(): Boolean CetStatus(): String	Panatt Assident() : Paslaan		Lime:	Concurrency	
GetStatus(): String	Cerify Accident(): Boolean		□ <u>A</u> bstract (	O Sequential O G	uarded 💿 Synchrono
	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:

iu nie to be ge	nerated	
	Packages and Components	
	OM_claim	
Assign		
	Deselect All	
	OF	Caraal
	Assign	Assign Packages and Components OM_claim Deselect All OK

#### Project Specification Project Specification ? × Classpath //Source file: C:\\Contents\\Rational Rose\\Claims\\JavaCode\\OM\_claim\\Accident.java Classpaths package OM\_claim; C:\Contents\Rational Rose\Claims\JavaCode \* 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() Reference Classpath return null; Resolve Pathmap Variables Set up the compiler classpath option with this project option /\*\* \* @return Boolean \* @roseuid 418A4EE5020F \* , 0K public Boolean CerifyAccident() Cancel return null; Returning to the Classpaths dialogue, click on the /\*\* component and click

'≊l×

JDK Auto Search

Apply

'Assign'. The component

```
}
```

/\*\*

\*/

{

}

ł

}

ł

}

\* /

}

\* @return String

return null;

\* @roseuid 418A4EE90124

public String GetStatus()

{

then disappears from the RHS of the screen. When the code generation is complete,



hopefully you will get

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:



# 12. Reverse Engineering a project

🕽 Java Reverse	Engineer				? ×
🤤 IC \Conter	ts\Rational Rose	AlienE Entity Game ShipE ShotE Sprite, Syster	ntity.java java ntity.java ntity.java ntity.java gava Store.java nTimer.java		
•		Filter:	*.java		•
Edit CLASSPA	тн	/	\dd	Add All	Add Recursive
FileName					Size
•					F
		(			

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.



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'.

# 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.

📀 Rational Rose - (untitled)

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.

File Edit View Format Browse Report Query Tools Add-Ins Window Help 🗅 🚅 🔚 🐰 🖻 🛍 🎒 😽 🗖 🖪 월 🕄 🗗 🖙 🗕 🔍 🔍 🛄 🔊 (untitled) 🖻 💼 Use Case View 🖻 💼 Business Use-Case Model 🙀 Global View of Business Actors and Business Use Cases 🗟 Associations 🗄 🔂 Use-Case Model 🗄 🔂 Actors 🕂 🔂 Use Cases हि 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 🗄 🔁 Associations 🖻 🔂 Use-Case Realizations 🗄 💼 <Use-Case Name> 🗄 🔁 Associations Architecturally Significant Model Elements Architecture Overview - Package and Subsystem Layering 🗟 Associations E) Welcome

<u>- 0 ×</u>

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'.