



Aldata
100% Retail-Wholesale



GOLD Stock Development

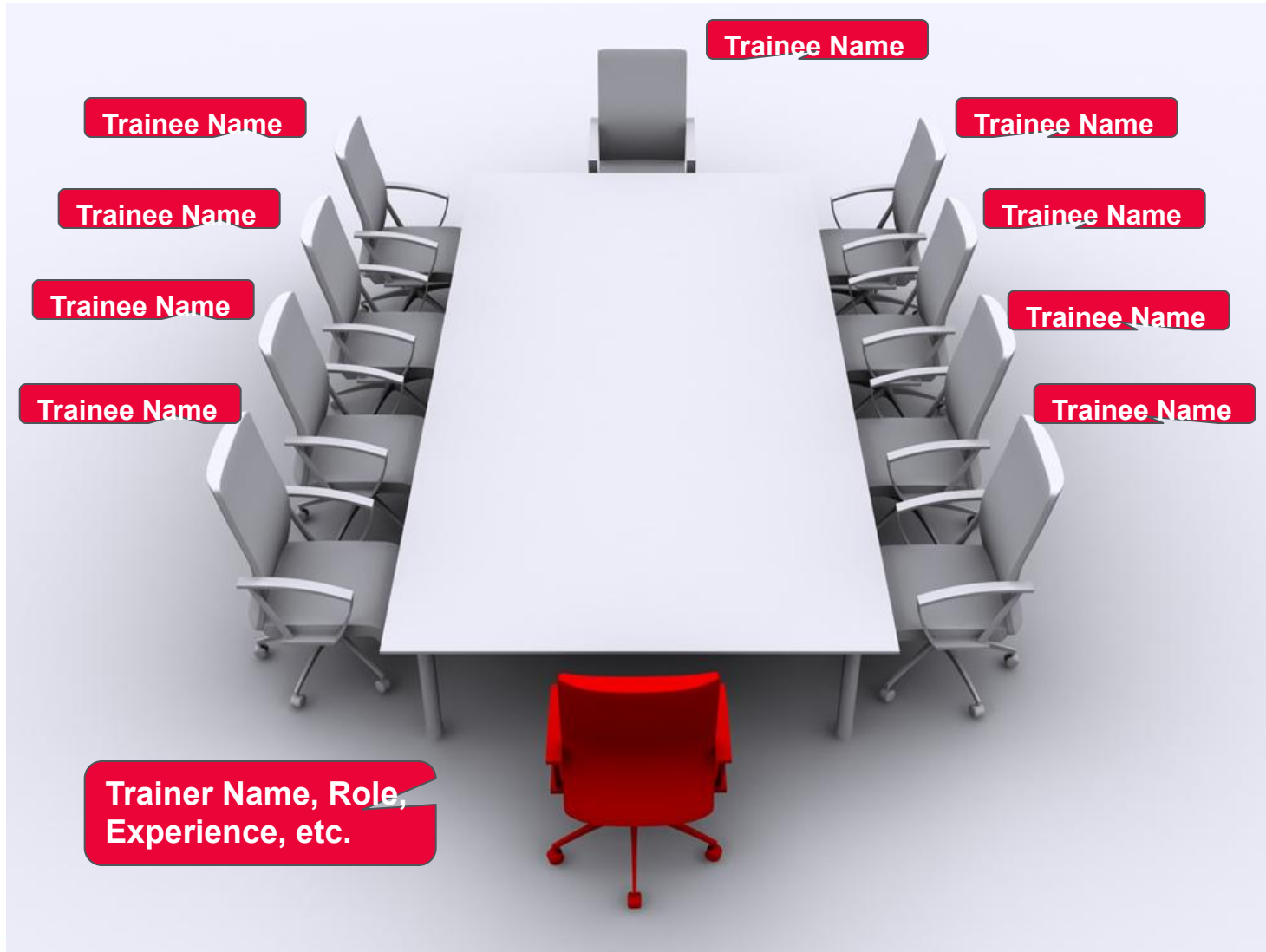
TS200
20090608



Aldata Training



Group Introduction



Objectives & Benefits



- After this training :

Know
Know

- You will understand :
 - The G.O.L.D. architecture
 - How G.O.L.D. is designed
- You will be able to :
 - Modify existing and create new G.O.L.D. screen
 - Modify and create a new report layout
 - Create and deploy a new program
 - Insert new screens and processes into the G.O.L.D application

Know-How
Know-How

Pre-Requisites



Knowledge

- JAVA
- PRO*C
- SQL
- PL-SQL

Software

e

- ADER (mandatory)
- Report Designer (mandatory)
- Eclipse (recommended)
- Crimson /Notepad++ / Ultra edit (or other editor)

Required

Material

- Graphic Framework Documentation (JavaDoc UI)

Table of content



1 General Overview

2 Global Architecture and Application Structure

3 G.O.L.D. Screens

4 Create a new eStock screen

5 Create a new report

Table of content



1 General Overview : **About G.O.L.D**

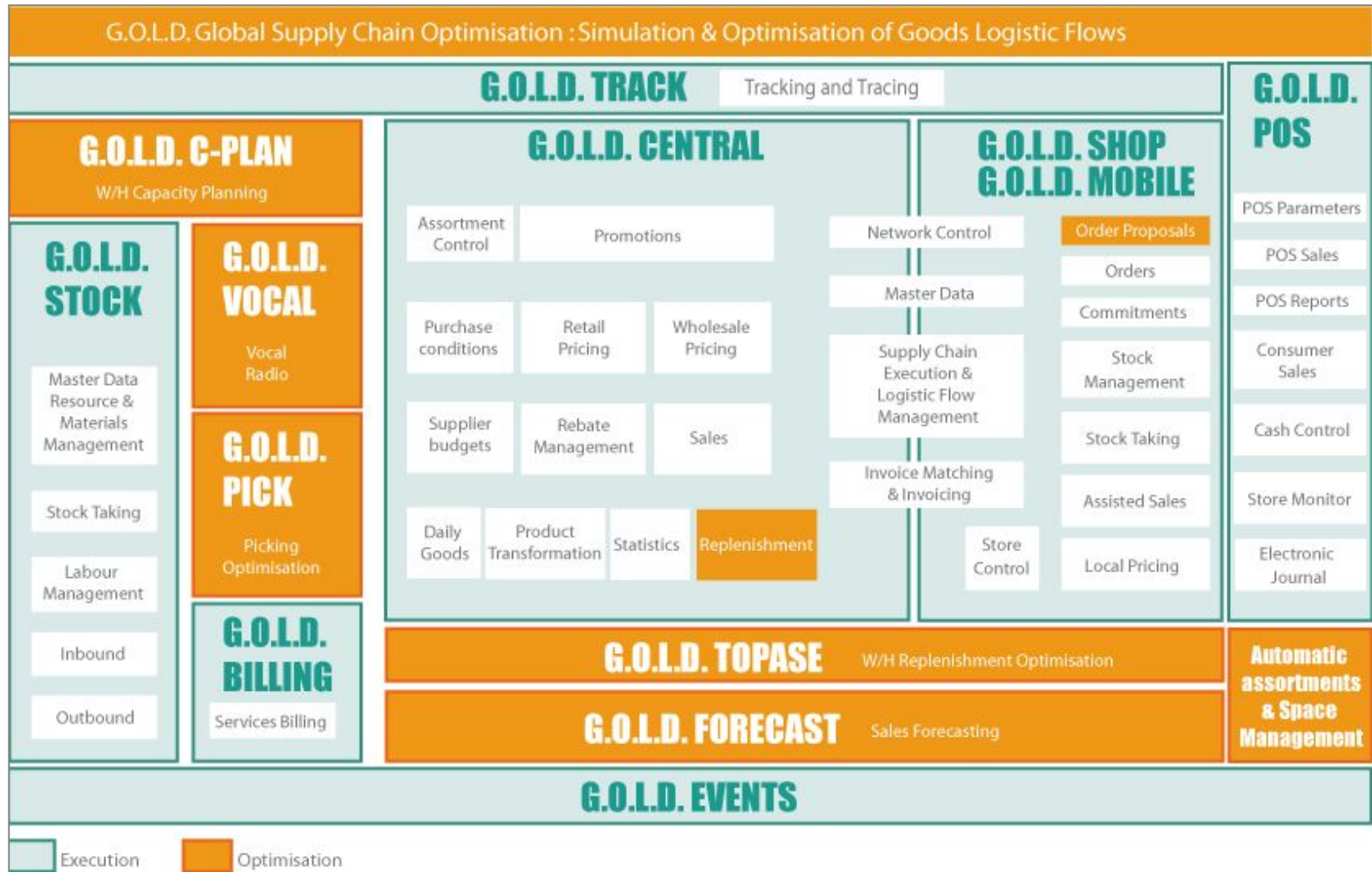
2 Global Architecture and Application Structure

3 G.O.L.D. Screens

4 Create a new eStock screen

5 Create a new report

G.O.L.D. solution map : a reminder



G.O.L.D. Stock



- G.O.L.D. Stock is a set designed to control the following functions:
 - the basic data of a warehouse
 - the physical movements in a warehouse: merchandise reception, pallet addressing, storage,
 - task execution, task scheduling and launch, task preparation and performing
 - the physical organization of the merchandise storage in the warehouse
 - the follow-up and the productivity of the staff in the warehouse
 - the use of the Radio for fork-lift trucks and FLT-drivers
 - the use of the Vocal Radio for preparation clerks
- Once set up, G.O.L.D. Stock offers a warehouse the following contributions:
 1. optimization of special handlings
 2. management of merchandise flows
 3. check of the storage level
 4. safety activities of the warehouse
 5. observance of FIFO rules

Interfacing G.O.L.D.

- **Definition :**
 - G.O.L.D. can be interfaced with third party systems from which it can receive data and to which it can send data.
 - 2 types of interfaces can be defined:
 - Integration interfaces (or inbound)
 - XML-based export interfaces (or outbound)



Table of content



1 General Overview : **Development tools**

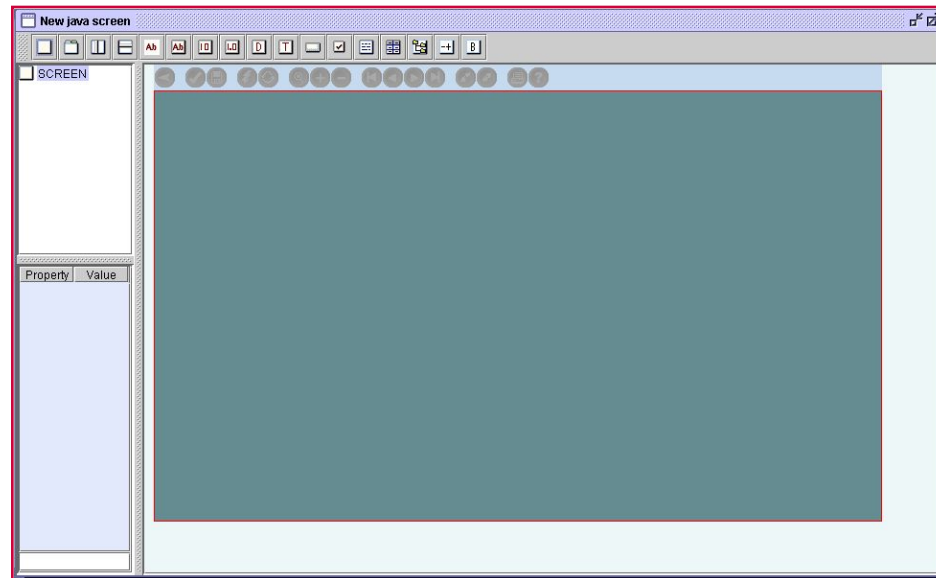
2 Global Architecture and Application Structure

3 G.O.L.D. Screens

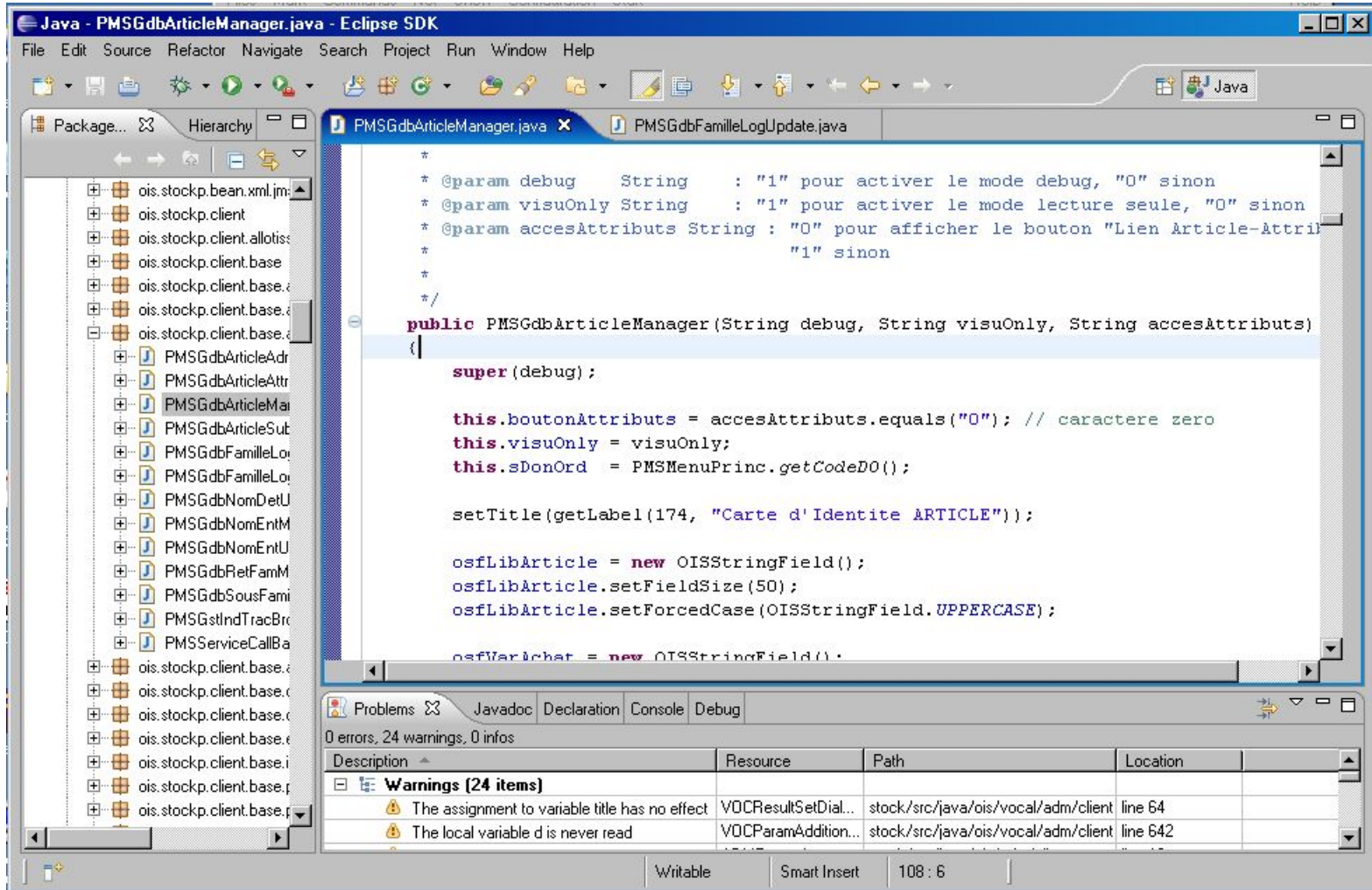
4 Create a new eStock screen

5 Create a new report

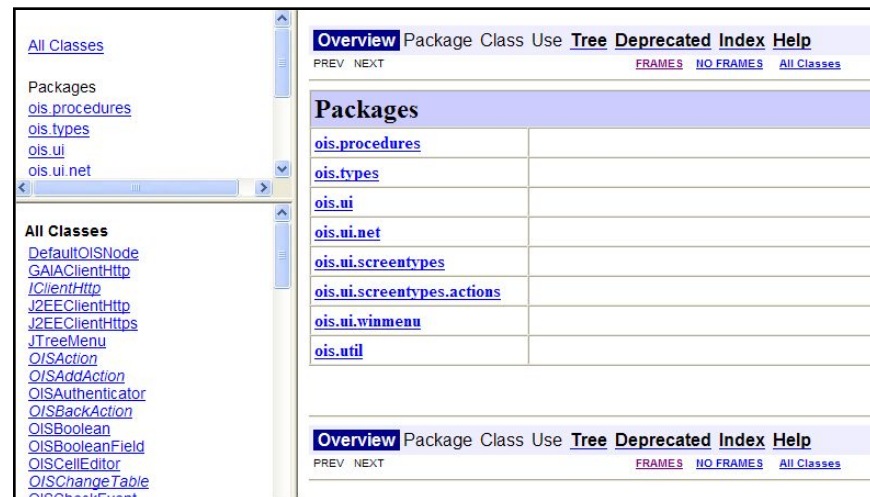
- **ADER :**
 - is a small application developed by the Aldata R&D team
 - is used to design Java screens quickly using the ALDATA Graphical Framework.
 - ADER translates your screen into Java Code
- **ADER uses the G.O.L.D. graphical library :**
 - is the core library of all Java screens in G.O.L.D.
 - 100 % full Java framework based on Java Swing Components



- Java integrated development environment

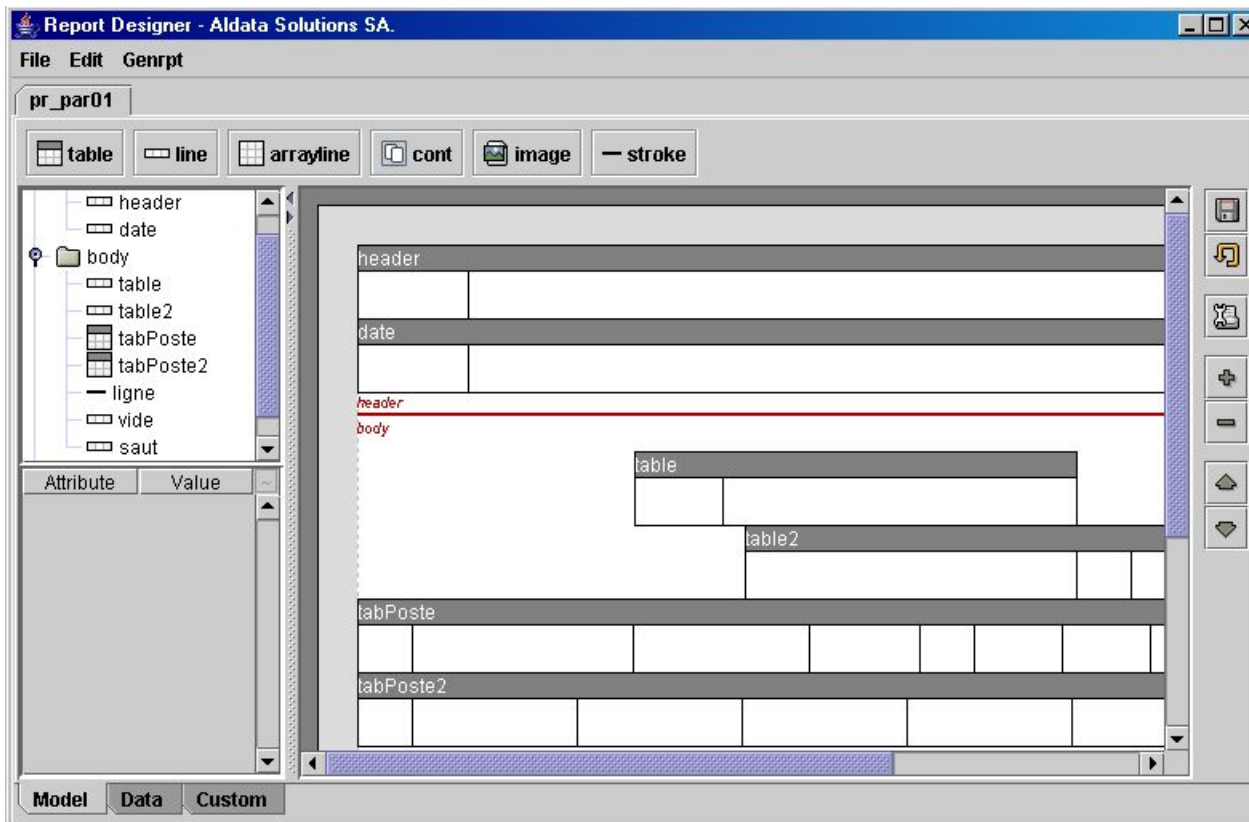


- **Graphic Framework Documentation (JavaDoc UI)**
 - is delivered in ZIP format
 - contains the rules and norms for being compatible with all G.O.L.D. standards including:
 - Programming rules
 - Ergonomic screen rules
 - Services management
 - Error management
 - Data field management

The screenshot shows the JavaDoc UI for the Graphic Framework. On the left, there is a navigation pane with "All Classes" and "Packages" sections. The "All Classes" section lists various classes like DefaultOISNode, GAIAClientHttp, etc. The "Packages" section lists ois.procedures, ois.types, ois.ui, and ois.ui.net. The main content area displays the "Overview" page for the "ois.ui" package, showing a table of packages and their classes. The table has columns for package names and class names. The packages listed are ois.procedures, ois.types, ois.ui, ois.ui.net, ois.ui.screenTypes, ois.ui.screenTypes.actions, ois.ui.winmenu, and ois.util. At the bottom, there is a navigation bar with "Overview", "Package", "Class", "Use", "Tree", "Deprecated", "Index", and "Help" tabs, and "PREV", "NEXT", "FRAMES", "NO FRAMES", and "All Classes" links.

Report Designer

- Report Designer :
 - is a small application developed by the Aldata R&D team
 - is a 100% Java (Swing) tool used to design reports according to a preset format





EXERCISE: Set Up Your Computer



- Setup server and database connection
- Install development tools (c:\sdt\
 - ADER
 - REPORT DESIGNER
 - Graphic Framework Documentation (JavaDoc UI)
 - Install: Crimson, WinMerge
 - Eclipse
- Setup local development environment
 - Configure Eclipse project for developing java screens

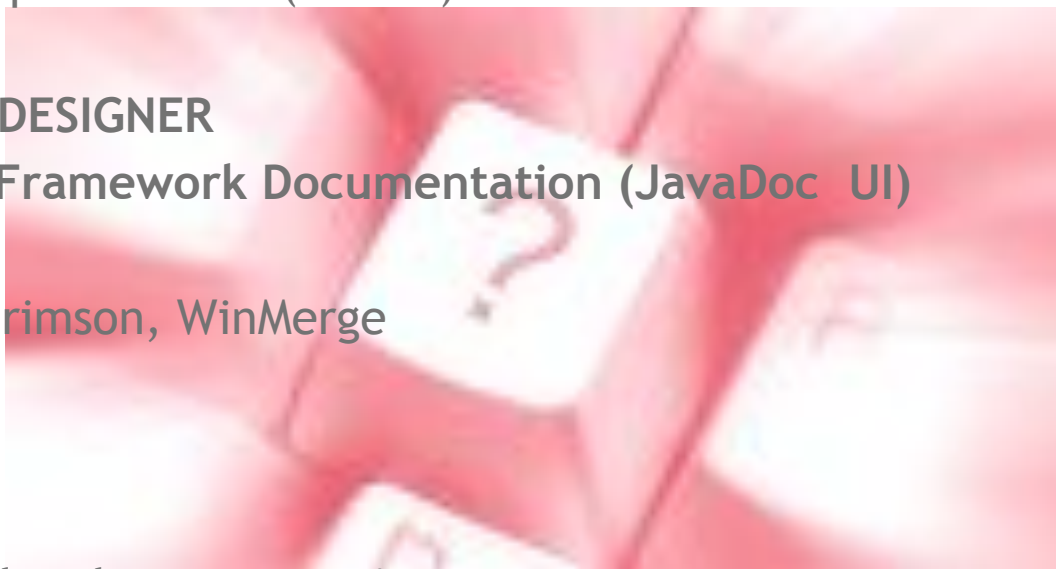


Table of content

1 General Overview

2 **Global Architecture** and Application Structure

3 G.O.L.D. Screens

4 Create a new eStock screen

5 Create a new report

A 3 tiers architecture

G.O.L.D. is a 3 tiers architecture solution where :



The **G.O.L.D. clients** are java-based applications used to access the screens from a PC with a simple browser. For instance

- G.O.L.D. central/shop client : eRetail
- G.O.L.D. Stock client : eStock

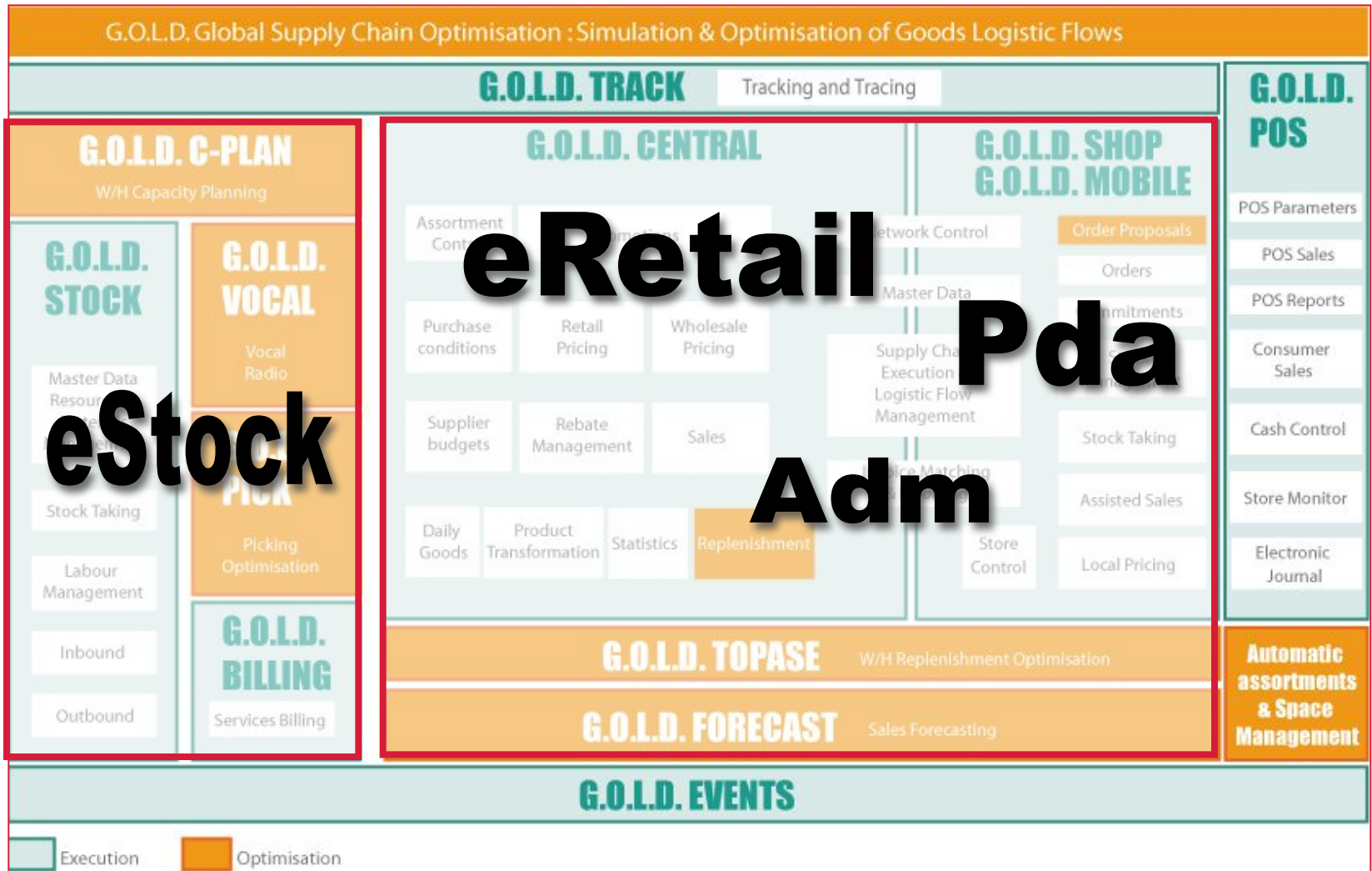


The G.O.L.D. application server **GAIA** handles the communication between the G.O.L.D. clients and the database server and hosts most of the processing.

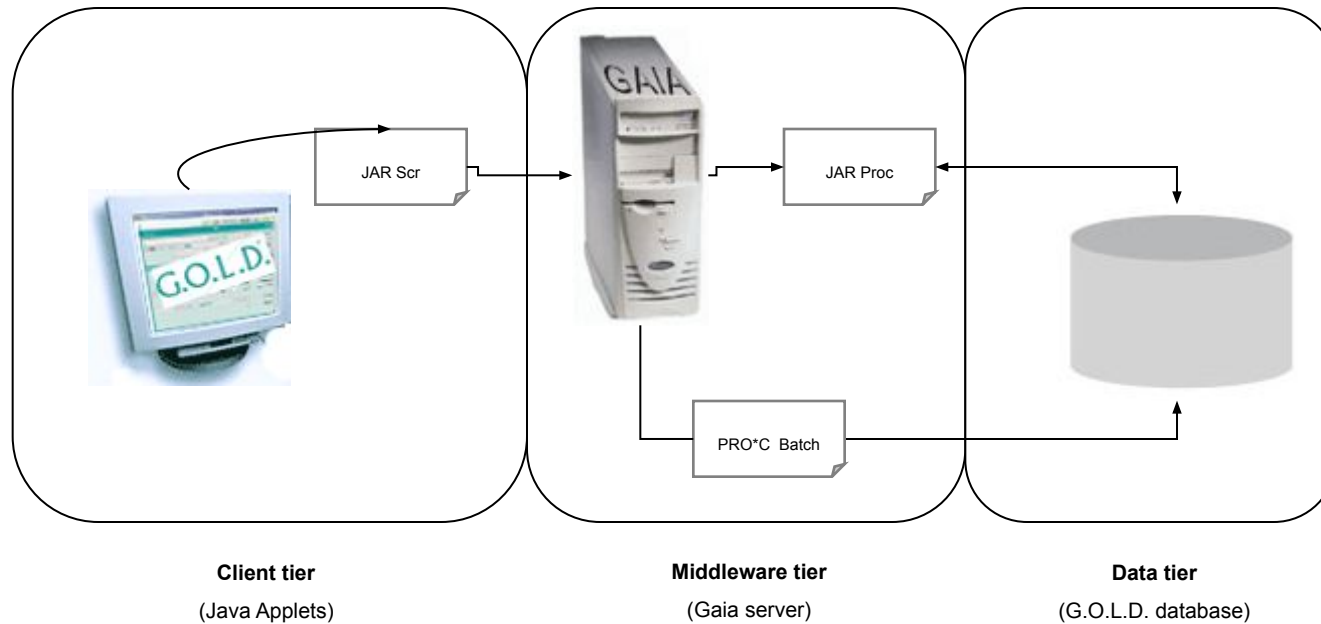


The Oracle **database server** stores the data and part of the application logic

G.O.L.D. Modules managed



eStock Application



- **Client tier:** data display, user events and controls of user interface. Java. Aldata graphical library.
- **Middleware tier:** encapsulates the applicative logic and makes it available to the client. Application framework. Communication.
- **Data tier:** responsible for the data storage and management. Oracle.

Deployment

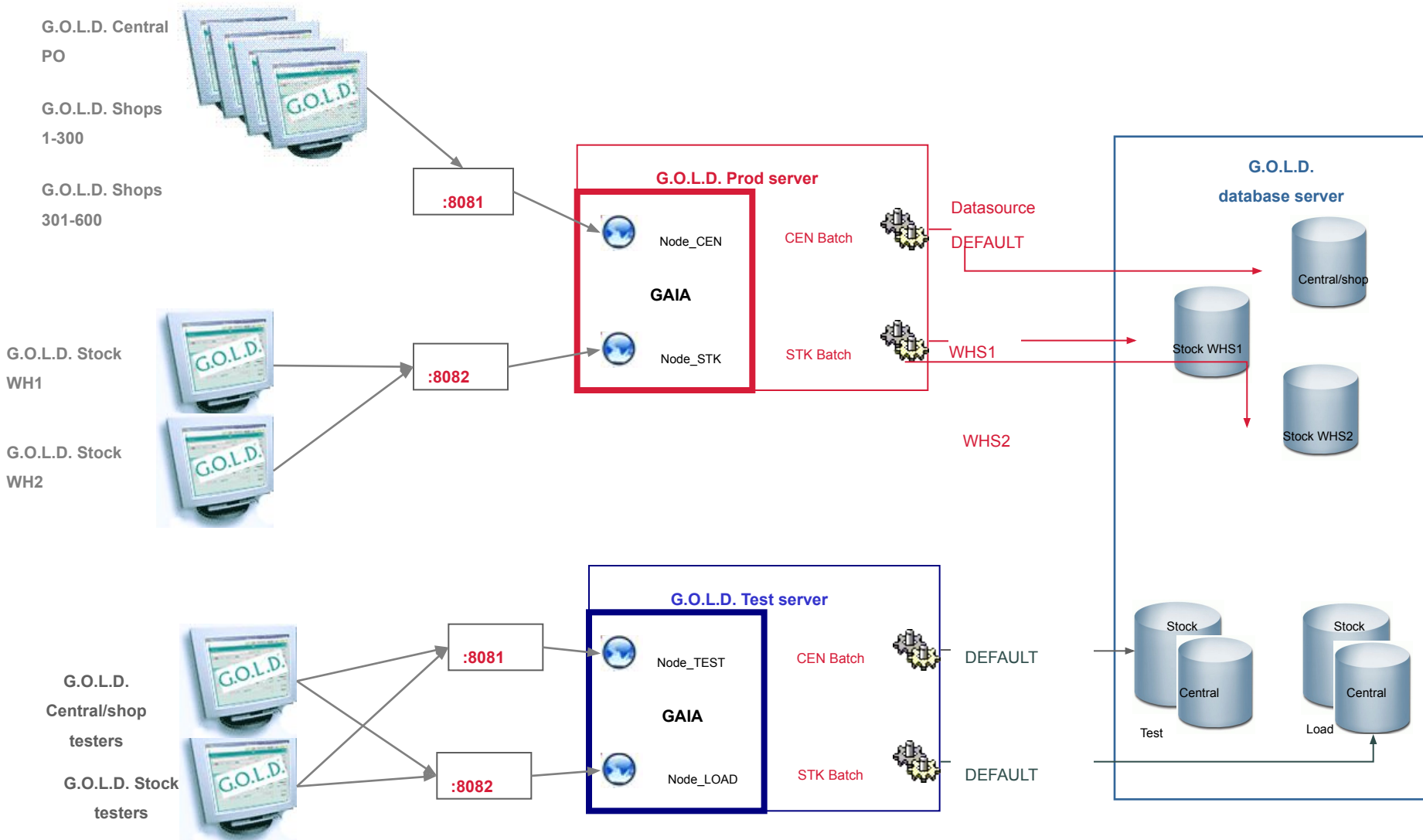


Table of content



1 General Overview

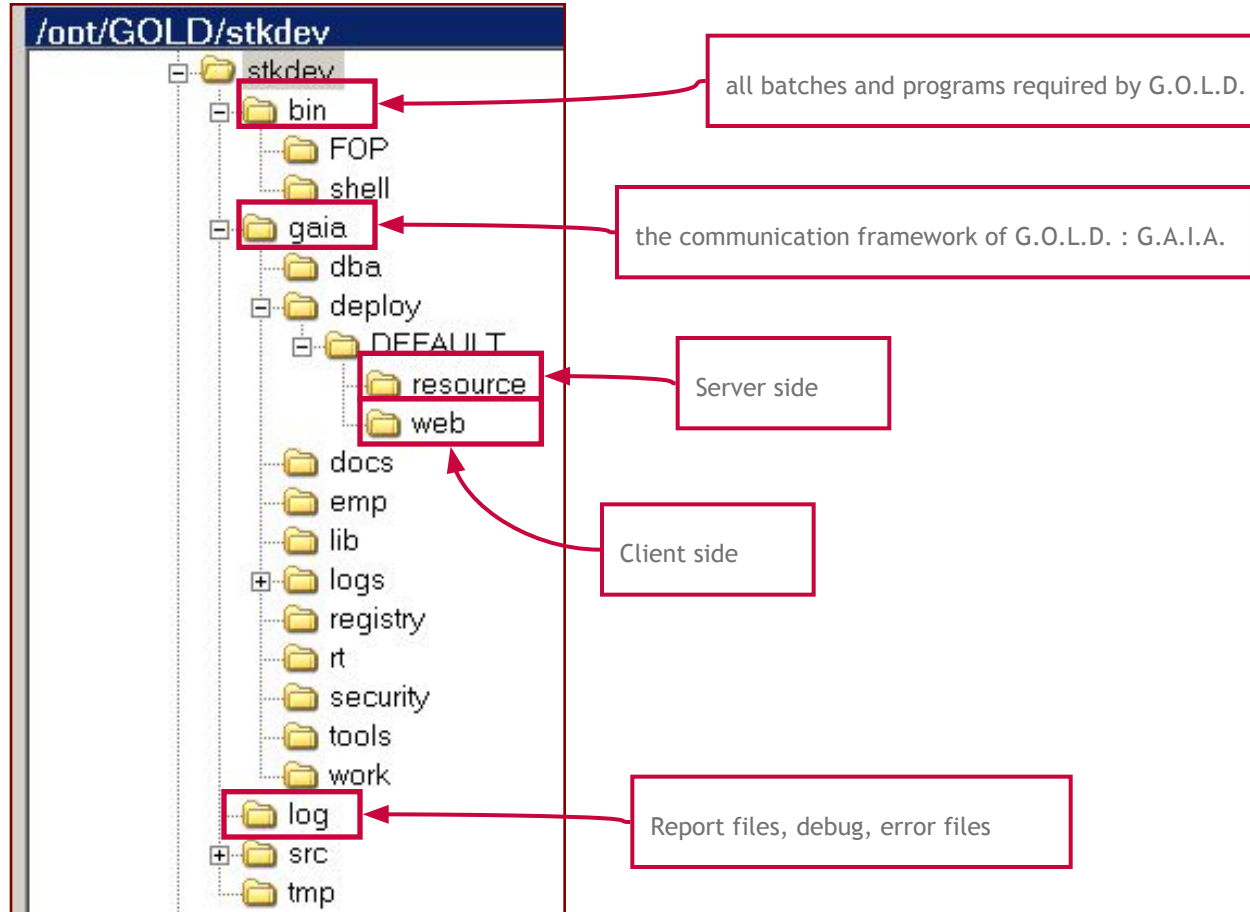
2 Global Architecture and **Application Structure**

3 G.O.L.D. Screens

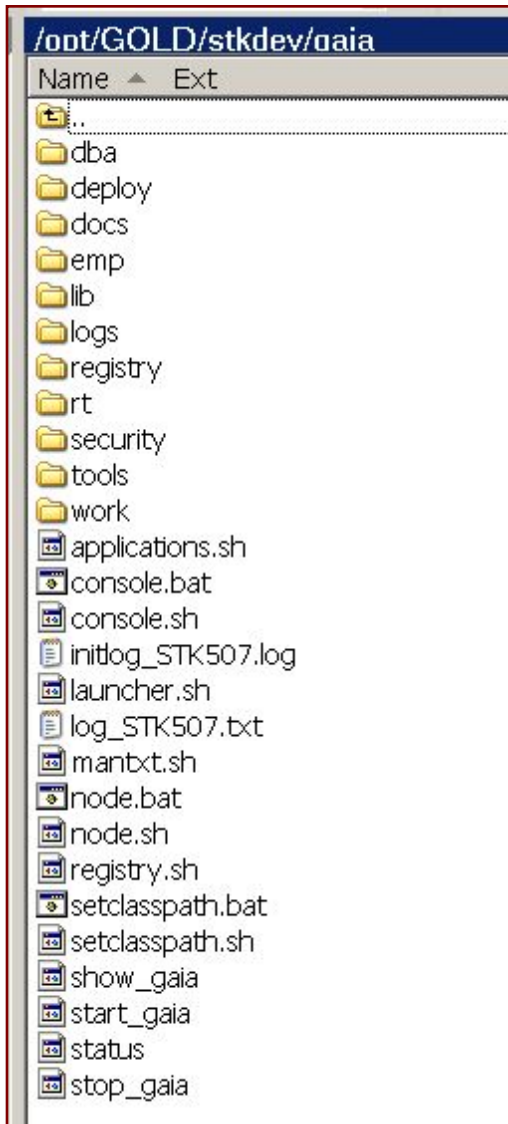
4 Create a new eStock screen

5 Create a new report

Application Structure

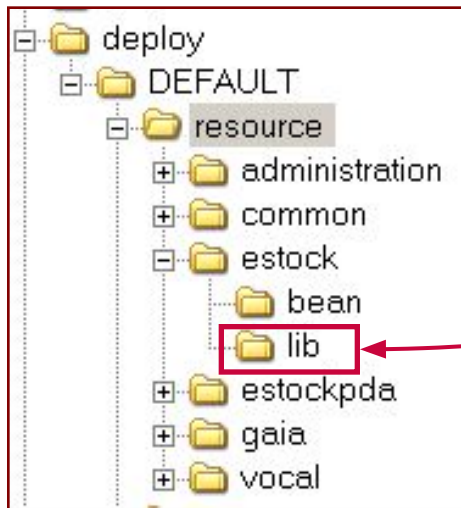


G.A.I.A.



\deploy	XML configuration files of G.A.I.A. default.xml Default directory for server and client installation
\logs	Log files per day, per user
\registry	Registry files
\tools	Tools used by G.A.I.A. beans
node.sh	Start GAIA node
start_gaia	
stop_gaia	
log_*.txt	

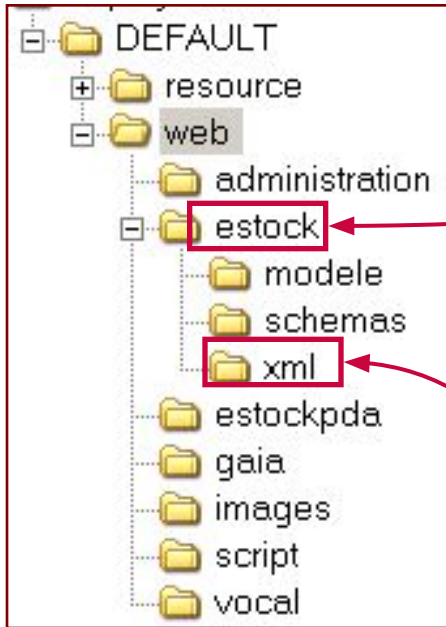
Server Side



```
ois.stockp.server.stock.promotions.jar
ois.stockp.server.supervisionprepa.jar
ois.stockp.server.tournees.affectation.jar
ois.stockp.server.tournees.jar
ois.stockp.server.tournees.parametrage.jar
ois.stockp.server.tracabilite.jar
ois.stockp.server.tracabilite.parametrage.jar
ois.stockp.server.vocal.alloting.jar
ois.stockp.server.vocal.inventaire.jar
ois.stockp.server.vocal.jar
ois.stockp.server.vocal.picking.jar
ois.stockp.server.vocal.repartition.jar
```

\administration	server resources of G.O.L.D.® Administration
\common	server resources common to all applications
\estock	server resources of G.O.L.D.® Stock
\estockpda	server resources of G.O.L.D.® Stock PDA
\gaia	server resources of G.A.I.A.

Client Side



```
gold.jpg
gw.jar
index.html
labor.jar
login.html
ois.stockp.client.allotissement.jar
ois.stockp.client.base.administration.jar
ois.stockp.client.base.alcools.jar
ois.stockp.client.base.articles.jar
ois.stockp.client.base.
ois.stockp.client.base
```

Report templates



\administration	G.O.L.D.® Administration WEB application (.html and .jar)
\estock	G.O.L.D.® Stock WEB application (.html and .jar)
\gaia	G.A.I.A. WEB application
\images	images of the HTML start page
\script	

Table of content



1 General Overview

2 Global Architecture and Application Structure

3 G.O.L.D. Screens

4 Create a new eStock screen

5 Create a new report

G.O.L.D. Screens



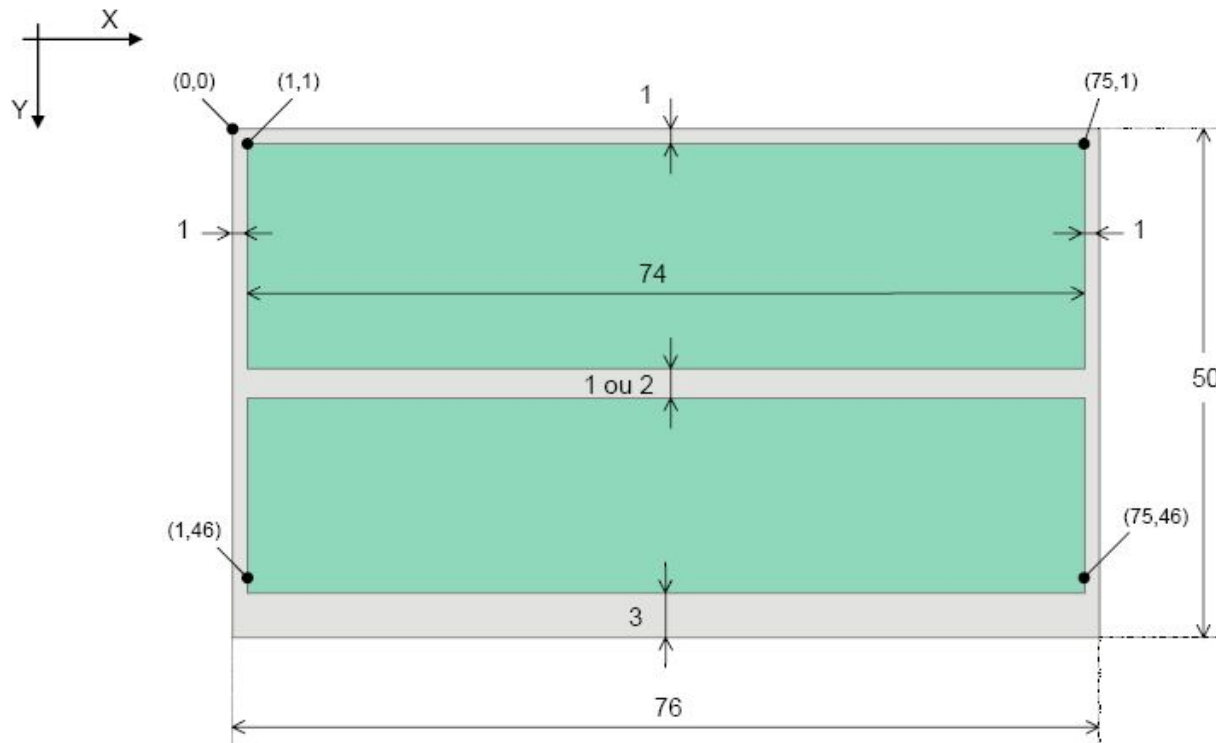
- ADER is used to create screens quickly and easily
 - while maintaining a *common look and feel*

- Following are the guidelines you must follow to design screens

Standard Screen

- Rules :

- a A margin of 1 is set between the panels and the edges of the screen
- b Between two panels, leave a margin of 2 (by default) or 1 (if lack of space).
- c A minimum margin of 3 must be set between the panels and the edge of the lower screen, to display the menu icon.



Standard Screen : example

54 DEPOT NANTES Third party management 12345 DOSTK

Third party type SUPPLIER Status

Internal code Short description

External code

Postcode Town/City

Internal code	External code	Description	Type	Postcode	Town/City	Status
16777	16777	BASF SEM	SUPPLIER	75011	PARIS	ACTIVE
17318	17318	GENERALE I	SUPPLIER		NEW YORK	ACTIVE
17478	17478	COCA COLA	SUPPLIER	59450	DUNKERQUE	ACTIVE
17608	17608	DURACELL	SUPPLIER	75006	PARIS	ACTIVE
17803	17803	EUROTEXTIL	SUPPLIER	ACTIVE
17805	17805	ASTRA CALV	SUPPLIER	75013	PARIS	ACTIVE
17907	17907	CELLIER DE	SUPPLIER	62100	BOULOGNE	ACTIVE
17933	17933	CANON PHOT	SUPPLIER	75001	PARIS	ACTIVE
18823	18823	HEINZ SARL	SUPPLIER		NEW YORK	ACTIVE
20	DEPOT 20	DEPOT 20	SUPPLIER	35000	RENNES	ACTIVE
2136	2136	FRUIT OF T	SUPPLIER	75005	PARIS	ACTIVE
21965	21965	NIVEA	SUPPLIER	75005	PARIS	ACTIVE
22899	22899	EQUIPEMENT	SUPPLIER	75007	PARIS	ACTIVE

Global locations

Small Screen

- Rules :
 - Adjust the size of the screen according to its content.
 - By default: centre.

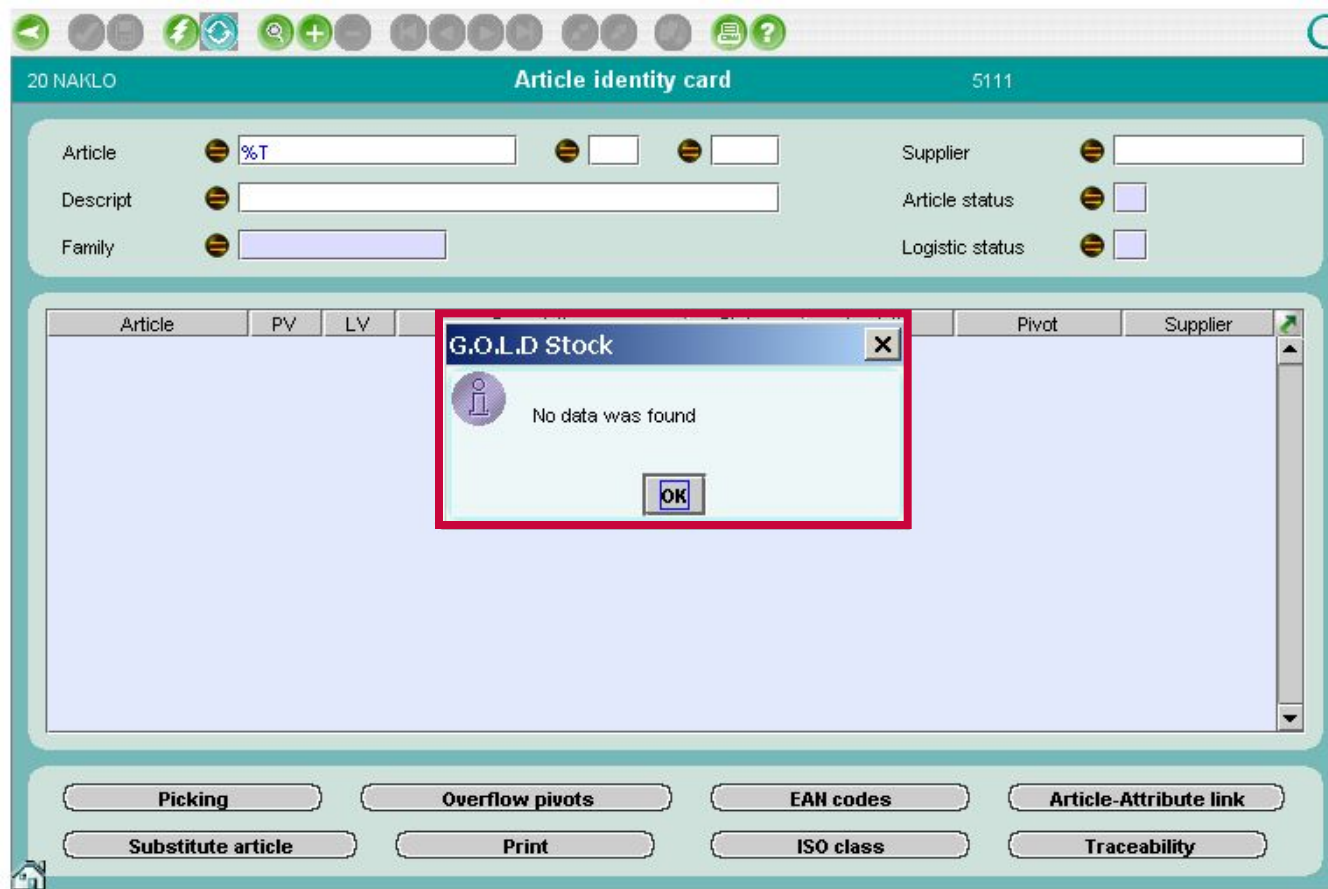
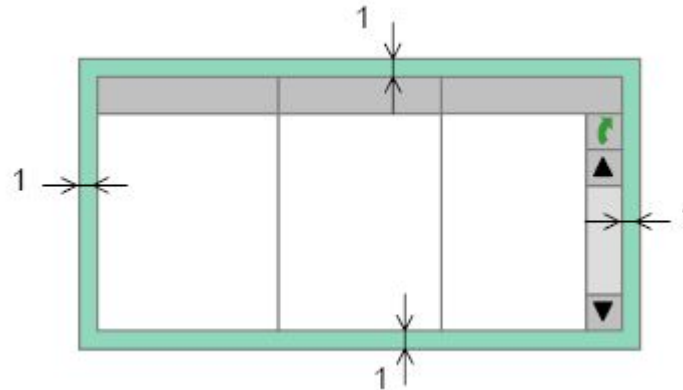


Table Layout – 1/3

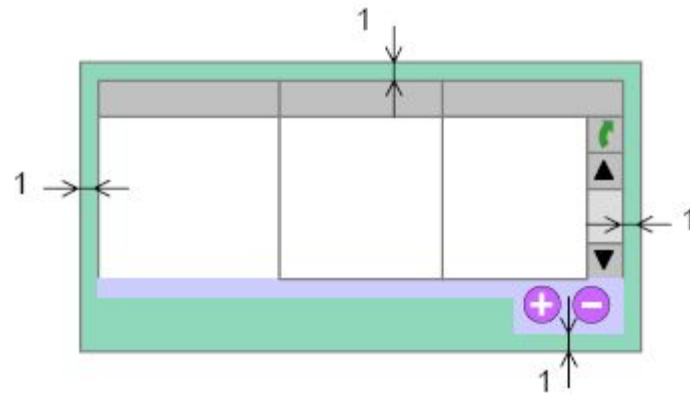
- Standard Panel



Article	PV	LV	Description	Status	Logistic	Pivot	Supplier
DNG050-003	00	000	MATIERE DANGER DLC PDS RE...	CREA LEV 3	COMPLETE		1140
DNG050-004	00	000	MATIERE DANGER DLC PDS RE...	ACTIVE	COMPLETE	54A0101904A	1140
DNG050-005	00	000	MATIERE DANGER DLC PDS RE...	CREA LEV 3	COMPLETE		1140
EMB000-600	00	000	EMBALLAGE PALETTE 80X120	CREA LEV 3	INCOMPLETE		1140
EMB000-601	00	000	EMBALLAGE PALETTE 100X120	CREA LEV 3	INCOMPLETE		1140
EMB000-602	00	000	EMBALLAGE ROLL	CREA LEV 3	INCOMPLETE		1140
EMB000-603	00	000	EMBALLAGE CONTAINER ISOT...	CREA LEV 3	INCOMPLETE		1140
EMB000-604	00	000	EMBALLAGE CASIER PLASTIQUE	CREA LEV 3	INCOMPLETE		1140
EMB000-605	00	000	EMBALLAGE BOUTEILLE 1	CREA LEV 3	INCOMPLETE		1140
EMB000-606	00	000	EMBALLAGE BOUTEILLE 2	CREA LEV 3	INCOMPLETE		1140
EMB000-607	00	000	EMBALLAGE EMBALLAGE UVC	CREA LEV 3	INCOMPLETE		1140
EMB000-608	00	000	EMBALLAGE CARTON	CREA LEV 3	INCOMPLETE		1140
PAQ01	00	000	ARTCILE GENERIQUE PAQ01	ACTIVE	COMPLETE		1140
PAQ02	00	000	ARTCILE GENERIQUE PAQ02	ACTIVE	COMPLETE		1140
PAQ03	00	000	ARTCILE GENERIQUE PAQ03	ACTIVE	COMPLETE		1140

Table Layout – 2/3

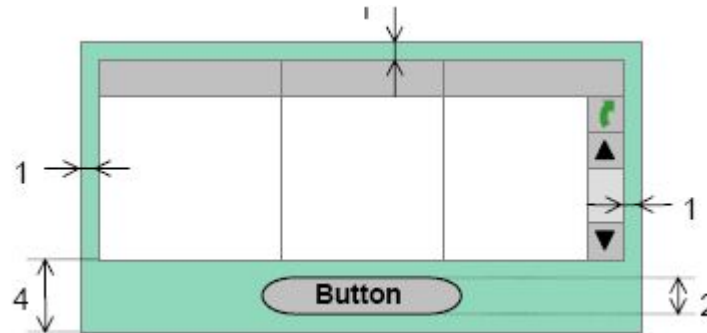
- Modifiable Panel



Printer name	Type	Printer type	Launching instruction	Portrait	Landscape	Depot	Depot desc.
ETI06	E	LABEL	ls>/dev/null			06	DEPOT TOULO...
ETI10	E	LABEL	ls>/dev/null			10	DEPOT BELFORT
ETI154	E	LABEL	ls>/dev/null			54	DEPOT NANTES
ETI20	E	LABEL	lp &/dev/lst			20	DEPOT RENNES
ETI36	E	LABEL	ls>/dev/null			36	DEPOT NARBO...
ETI40	E	LABEL	ls>/dev/null			40	DEPOT PARIS
ETI44	E	LABEL	ls>/dev/null			44	DEPOT LILLE
ETI54	E	LABEL	ls>/dev/null			54	DEPOT NANTES
LIST01	L	LIST	ls>/dev/null			01	DEPOT BORDE...
LIST02	L	LIST	ls>/dev/null			02	DEPOT LYON
LIST03	L	LIST	ls>/dev/null			03	DEPOT MARSE...
LIST06	L	LIST	ls>/dev/null			06	DEPOT TOULO...
LIST10	L	LIST	ls>/dev/null			10	DEPOT BELFORT
LIST154	L	LIST	ls>/dev/null			54	DEPOT NANTES
LIST20	L	LIST	ls>/dev/null			20	DEPOT RENNES
LIST36	L	LIST	ls>/dev/null			36	DEPOT NARBO...
LIST40	L	LIST	ls>/dev/null			40	DEPOT PARIS
LIST44	L	LIST	ls>/dev/null			44	DEPOT LILLE
LIST54	L	LIST	ls>/dev/null			54	DEPOT NANTES

Table Layout – 3/3

- Button Panel (table + button on same panel)
 - Button will be centered if unique

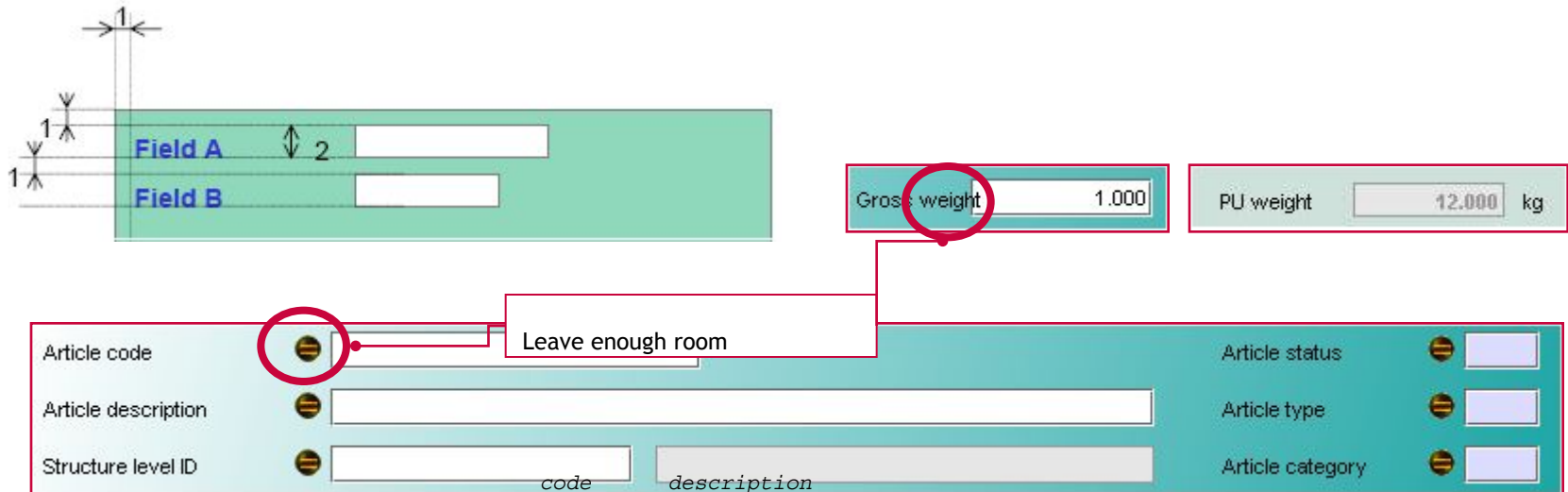


Site	Description	Allowed automatic...	Default address c...	Start date	End date	Order type
3003	PROXI TOULOUSE	<input checked="" type="checkbox"/>	1	12/20/03	12/31/49	▲
3002	PROXI DIJON	<input checked="" type="checkbox"/>	3	12/20/03	12/31/49	
3001	PROXI ORLEANS	<input checked="" type="checkbox"/>	2	12/20/03	12/31/49	
3000	PROXI CAHORS	<input checked="" type="checkbox"/>	1	12/20/03	12/31/49	
1103	SUPER LOOS	<input checked="" type="checkbox"/>	1	12/20/03	12/31/49	
1102	SUPER AUXERRE	<input checked="" type="checkbox"/>	4	12/20/03	12/31/49	
1101	SUPER AVIGNON	<input checked="" type="checkbox"/>	10	12/20/03	12/31/49	▼

Text Fields

- Text Fields :

- Height : 2 units.
- Plan enough space between the description and the field for the translation and for the query symbol
- Size of a screen field = Number of characters to be displayed + 1
 - Exception: long descriptions (i.e. 40-character article description must be inserted in a zone of 35)
- Code-description fields, must be side by side with a one-unit space
- For values in specific units, specify the unit.



Buttons and Checkboxes

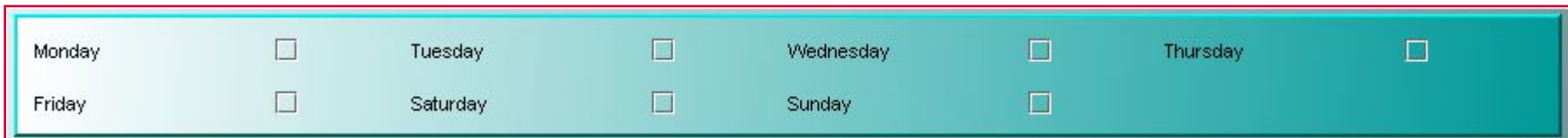
- Buttons :

- Size of Panel : 4 units
- Height : 2 units



- Checkboxes :

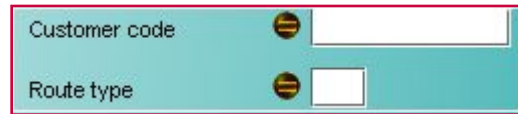
- Check box must be on the right of the description




Field Descriptions

- Descriptions :

- Only the first word of the description has a capital letter.



Customer code 
Route type 

- An abbreviation must always be followed with a dot.



Promo. plan 
Special op. 

- No accent in descriptions.



Unite de mesure 

- Possibility to use : - / ' ()



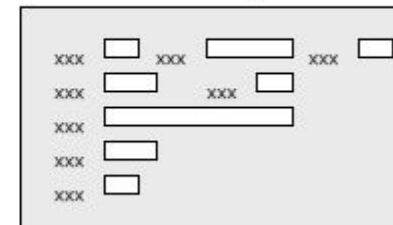
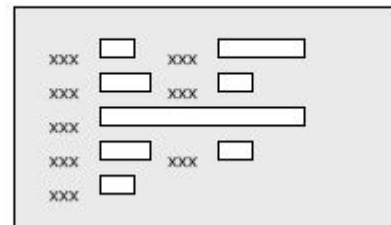
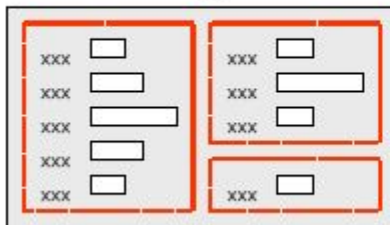
Description (opt.)



Weight / vol.

Layout

- The layout must be thought through so the tool stays user-friendly
- Fields must be vertically aligned, using panels when possible.



External code	<input type="text"/>	Statistic code	<input type="text"/>
Manufacturer code	<input type="text"/>	Model code	<input type="text"/>
Collection code	<input type="text"/>	Brand code	<input type="text"/>
Administrator code	<input type="text"/>	Referencing origin	<input type="text"/>
Certificate of origin	<input type="text"/>	Conformity certificate	<input type="text"/>

Delivering site	<input type="text"/>
End date	<input type="text"/>

Wrong

Scheduling screen



54 DEPOT NANTES **Scheduling**

Route Order

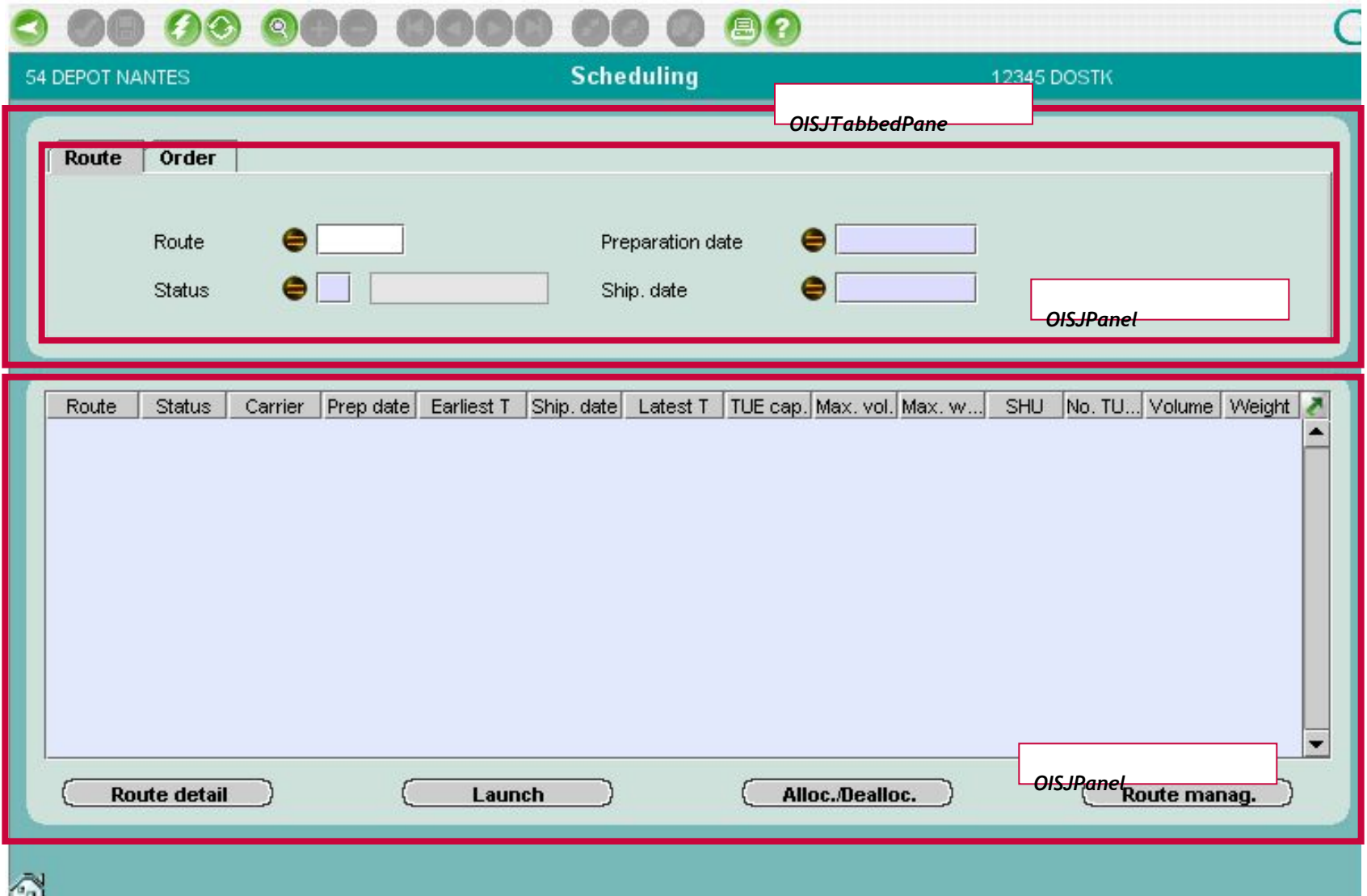
Route Preparation date

Status Ship. date

Route	Status	Carrier	Prep date	Earliest T	Ship. date	Latest T	TUE cap.	Max. vol.	Max. w...	SHU	No. TU...	Volume	Weight
-------	--------	---------	-----------	------------	------------	----------	----------	-----------	-----------	-----	-----------	--------	--------

Route detail Launch Alloc./Dealloc. Route manag.

Scheduling screen



54 DEPOT NANTES Scheduling 12345 DOSTK

OISJTabbedPane

Route Order

Route Preparation date

Status Ship. date

OISJPanel

Route	Status	Carrier	Prep date	Earliest T	Ship. date	Latest T	TUE cap.	Max. vol.	Max. w...	SHU	No. TU...	Volume	Weight
-------	--------	---------	-----------	------------	------------	----------	----------	-----------	-----------	-----	-----------	--------	--------

Route detail Launch Alloc./Dealloc. OISJPanel Route manag.

Scheduling screen

54 DEPOT NANTES Scheduling

ois.stockp.client.common.preparation.PMSRprListTourneeManager

Route **Order** *OISJLabel*

Route Preparation date *OISField*

Status Ship. date

Route	Status	Carrier	Prep date	Earliest T	Ship. date	Latest T	TUE cap.	Max. vol.	Max. w...	SHU	No. TU...	Volume	Weight
<i>OISJTable</i>													

Route detail **Launch** **Alloc./Dealloc.** **Route manag.**

OISJButton



EXERCISE: Screen Ergonomics



- TS200_GOLDStockDevelopment_Exercises - 1
 - Log into G.O.L.D.
 - Open screens and note the different types of layout
 - Open screens and review source code

- Warehouse map
 - Depot management
 - Warehouse area management
 - Warehouse area extension
 - Overflow area allocation



- Basic data - Third parties
 - Third party management
- Administration

Screen class name



EXERCISE: Modify screen



- TS200_GOLDStockDevelopment_Exercises - 2
 - Add new column in the table
 - (Add query field)

The screenshot shows a software interface for 'Third party management' at '54 DEPOT NANTES'. It features a form with fields for 'Third party type', 'Internal code', 'EXternal code', 'Postcode', 'Status', 'Short description', 'Preparation class', and 'Town/City'. Below the form is a table with columns: 'Internal code', 'EXternal code', 'Description', 'Type', 'Postcode', 'Town/City', and 'Status'. A 'Global locations' button is at the bottom.

Annotations on the screenshot:

- A red box labeled 'New query field' points to the 'Preparation class' field.
- A red box labeled 'New column preparation class' points to the 'Preparation class' column in the table.

Table of content



1 General Overview

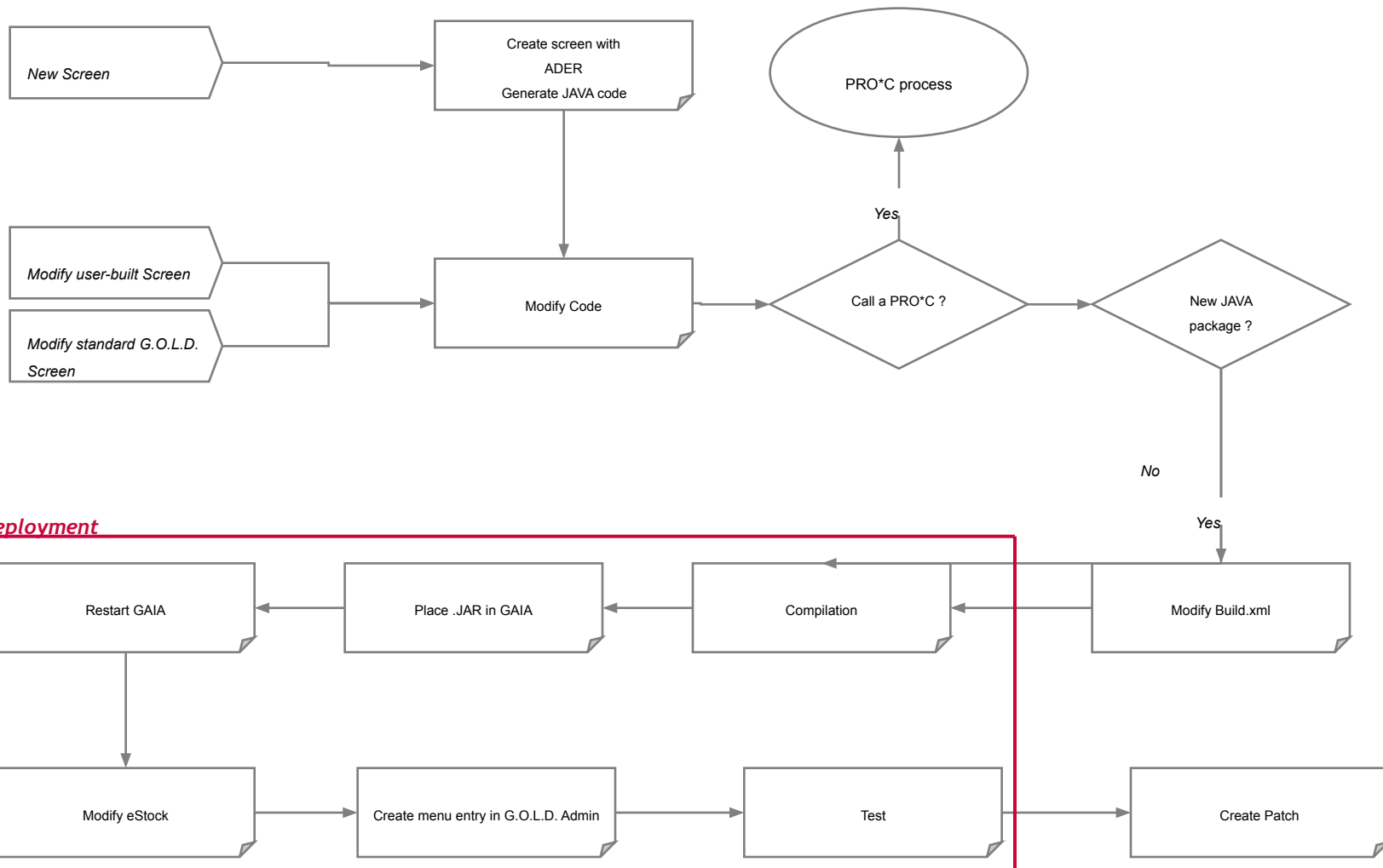
2 Global Architecture and Application Structure

3 G.O.L.D. Screens

4 Create a new screen

5 Create a new report

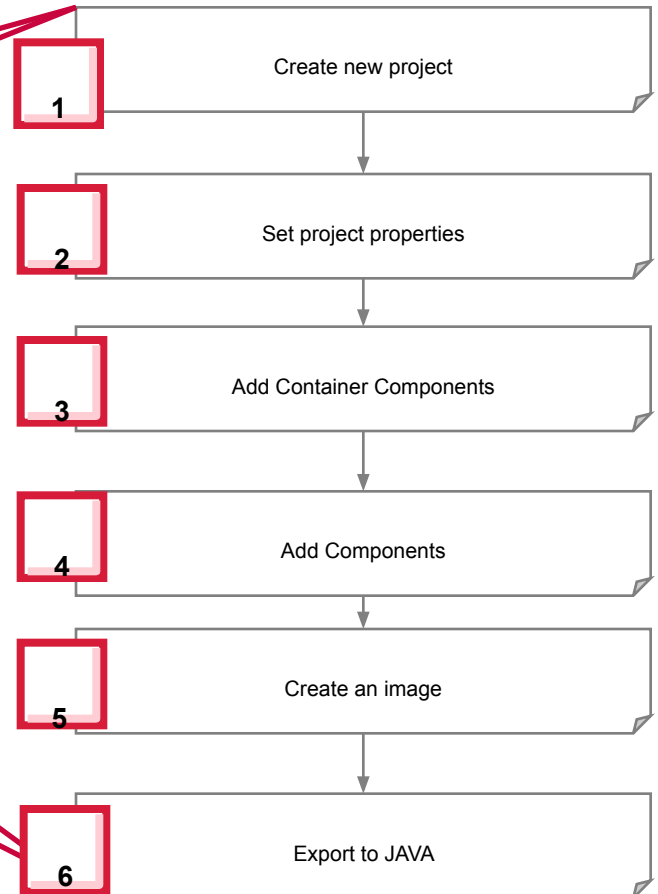
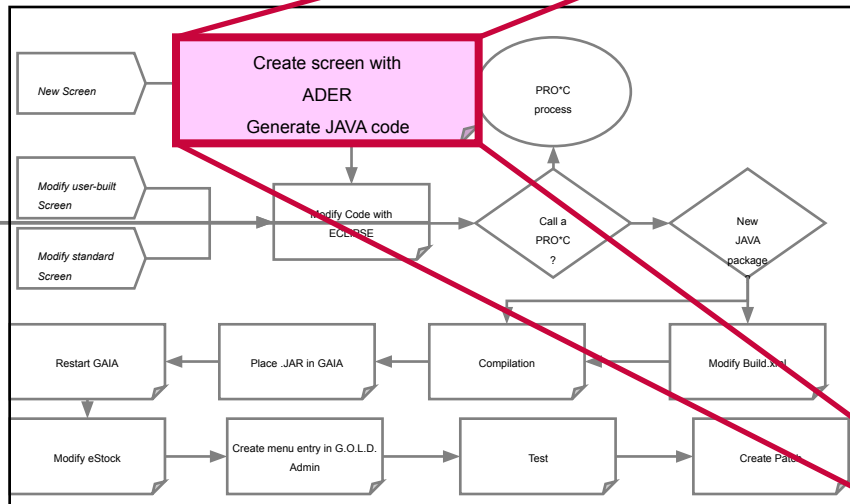
Create a screen process



- ADER is used to create G.O.L.D. screens quickly and easily.
- Screens are developed using the basic classes provided by the JAVA graphic framework.
- The generic G.O.L.D. toolbar is provided.

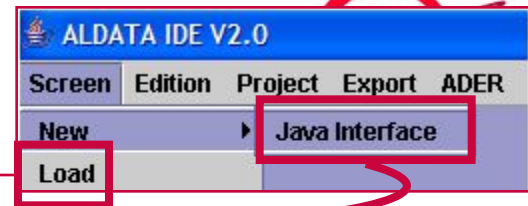
- **A few guidelines :**
 - Except for very particular cases, there must be no direct call to swing components (JAVA basic graphic class).

ADER process

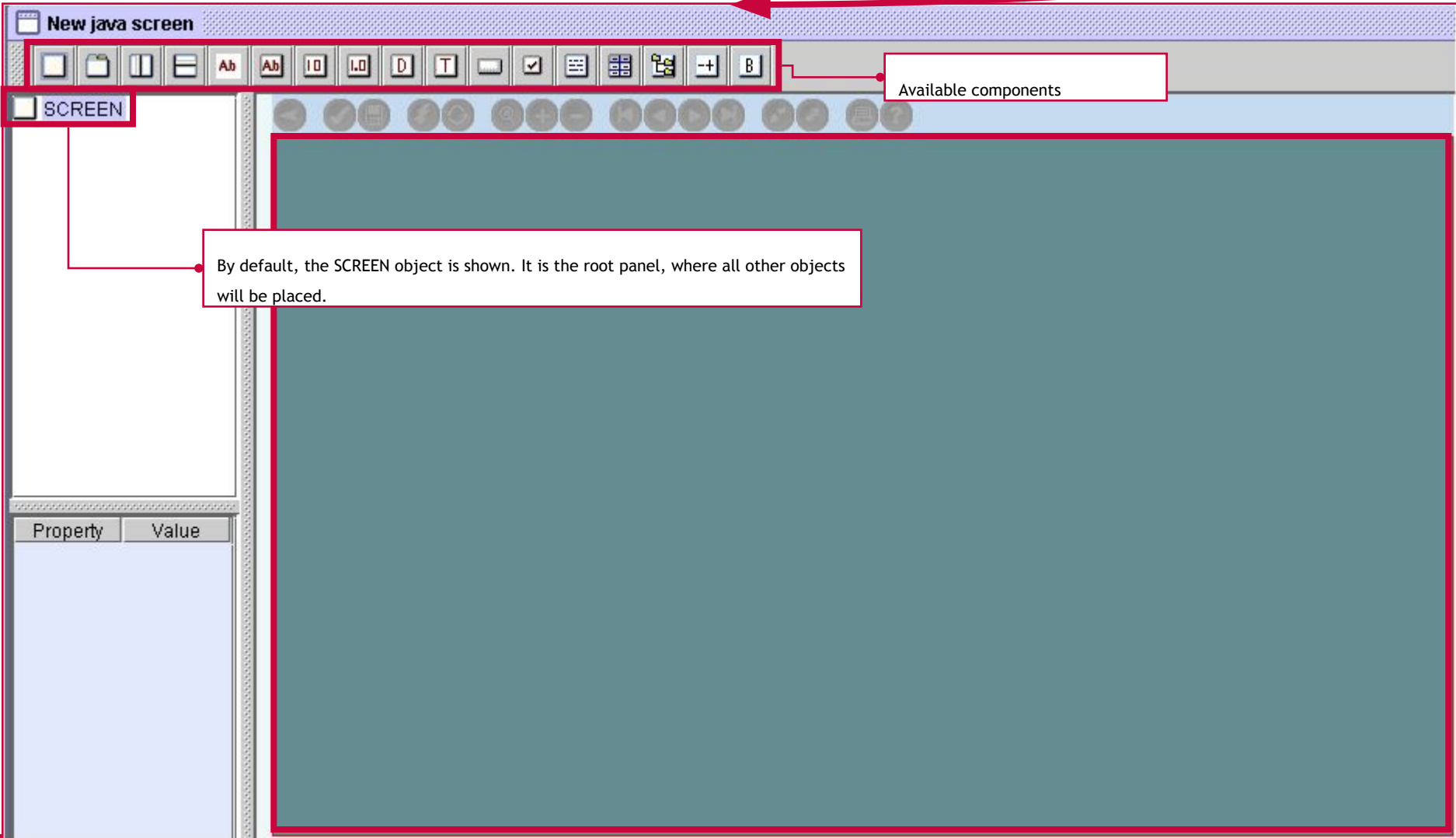


1

Create New Project

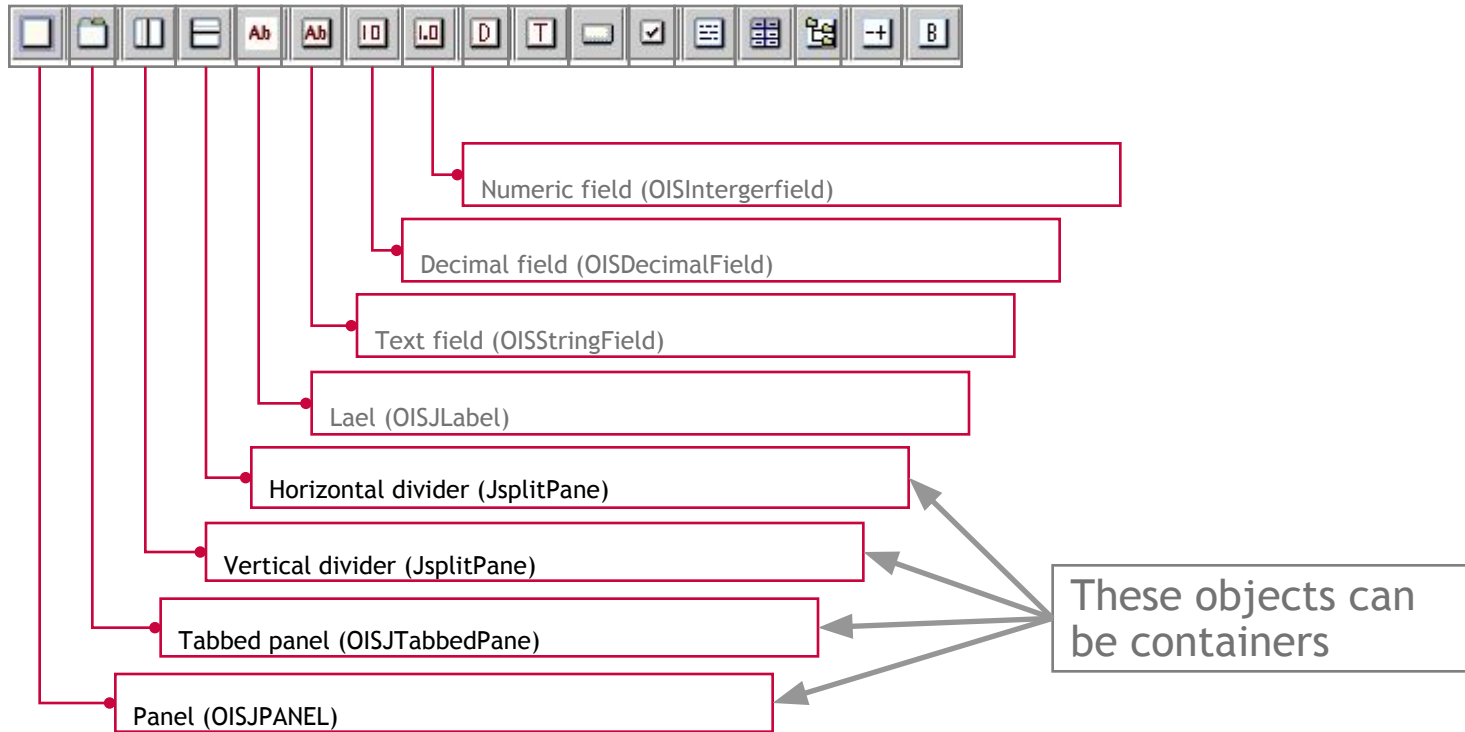


Open existing project

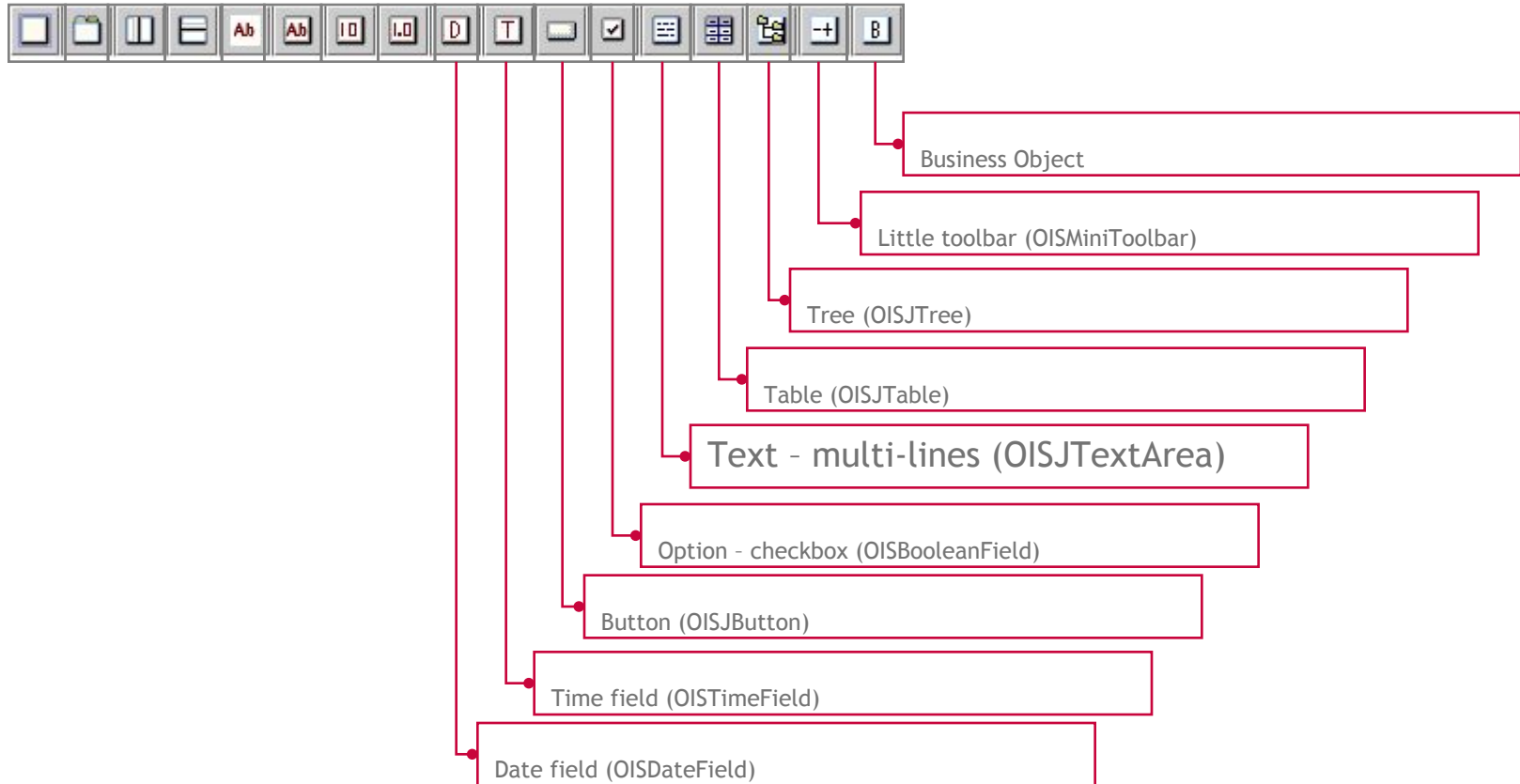


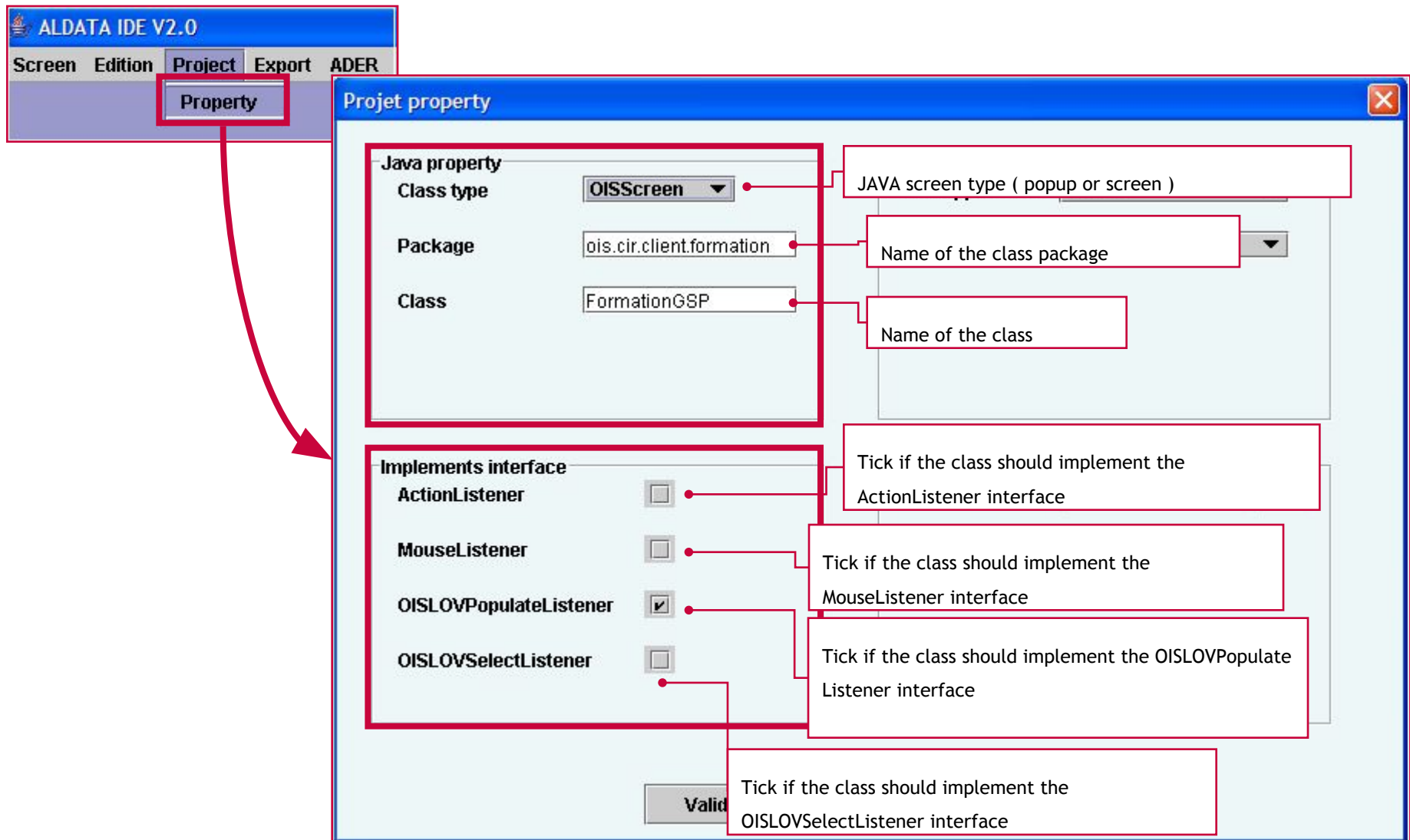
By default, the SCREEN object is shown. It is the root panel, where all other objects will be placed.

Available Components



Available Components





The screenshot shows the ALDATA IDE V2.0 interface with the **Project** menu open and the **Property** option selected. The **Projet property** dialog box is displayed, containing the following sections and fields:

- Java property**
 - Class type**: OISScreen (dropdown menu)
 - Package**: ois.cir.client.formation (text field)
 - Class**: FormationGSP (text field)
- Implements interface**
 - ActionListener**:
 - MouseListener**:
 - OISLOVPopulateListener**:
 - OISLOVSelectListener**:
- Valid**:

Annotations with red boxes and lines point to the following elements:

- JAVA screen type (popup or screen)**: points to the **Class type** dropdown.
- Name of the class package**: points to the **Package** text field.
- Name of the class**: points to the **Class** text field.
- Tick if the class should implement the ActionListener interface**: points to the **ActionListener** checkbox.
- Tick if the class should implement the MouseListener interface**: points to the **MouseListener** checkbox.
- Tick if the class should implement the OISLOVPopulate Listener interface**: points to the **OISLOVPopulateListener** checkbox.
- Tick if the class should implement the OISLOVSelectListener interface**: points to the **OISLOVSelectListener** checkbox.

Set Project Properties – 1/2

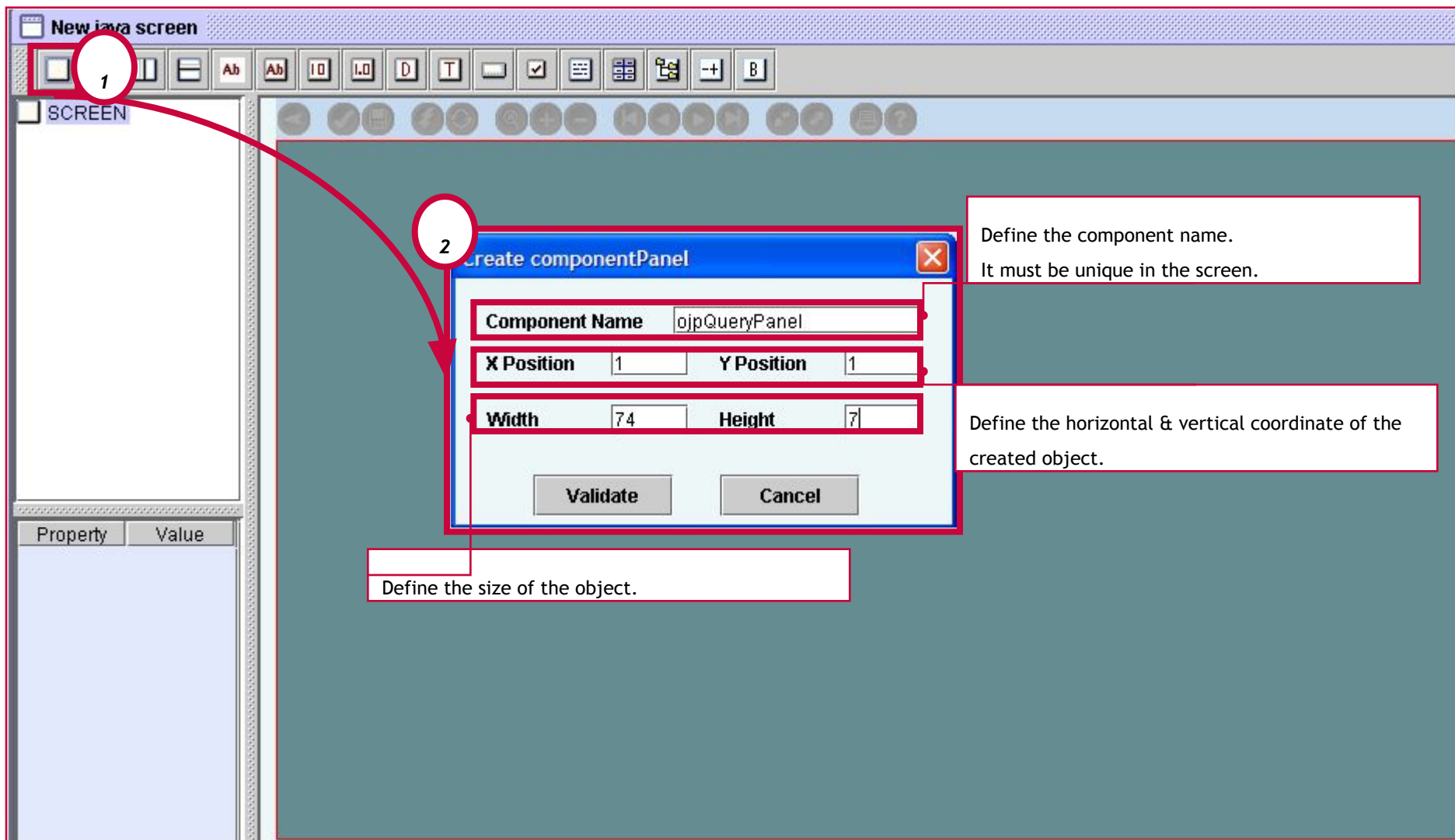


The screenshot shows the ALDATA IDE V2.0 interface with the 'Project property' dialog box open. The 'Project' menu is selected, and the 'Property' option is highlighted. The dialog box is titled 'Projet property' and contains the following sections:

- Java property:**
 - Class type: OISScreen
 - Package: ois.cir.client.formation
 - Class: FormationGSP
- GOLD informations:**
 - GOLD application: eRetail
 - Screen type: Query Screen
- Implements interface:**
 - ActionListener:
 - MouseListener:
 - OISLOVPopulateListener:
 - OISLOVSelectListener:
- others:**
 - External import: [] ...

Annotations with red boxes and arrows point to specific elements:

- A box labeled 'What application will use the class (retail, stock)' points to the 'GOLD application' dropdown.
- A box labeled 'Screen Type (query screen, modification...)' points to the 'Screen type' dropdown.
- A box labeled 'Link to a file with JAVA import list that should be included in the class' points to the 'External import' field.
- A box labeled 'Validate' highlights the 'Validate' button.



1

2

Define the component name.
It must be unique in the screen.

Define the horizontal & vertical coordinate of the created object.

Define the size of the object.

Property	Value
Component Name	ojpQueryPanel
X Position	1
Y Position	1
Width	74
Height	7

New java screen

SCREEN

- ojpButton
- ojpQueryPane**
- ojp

List of components in the screen

```
OISJPanel    ojpbButton;
OISJPanel    ojpbQueryPane;
OISJPanel    ojpb;
```

Properties of selected component.

Property	Value
Name	ojpQueryPane
Comment	...
X	1
Y	1
Width	74
Height	7

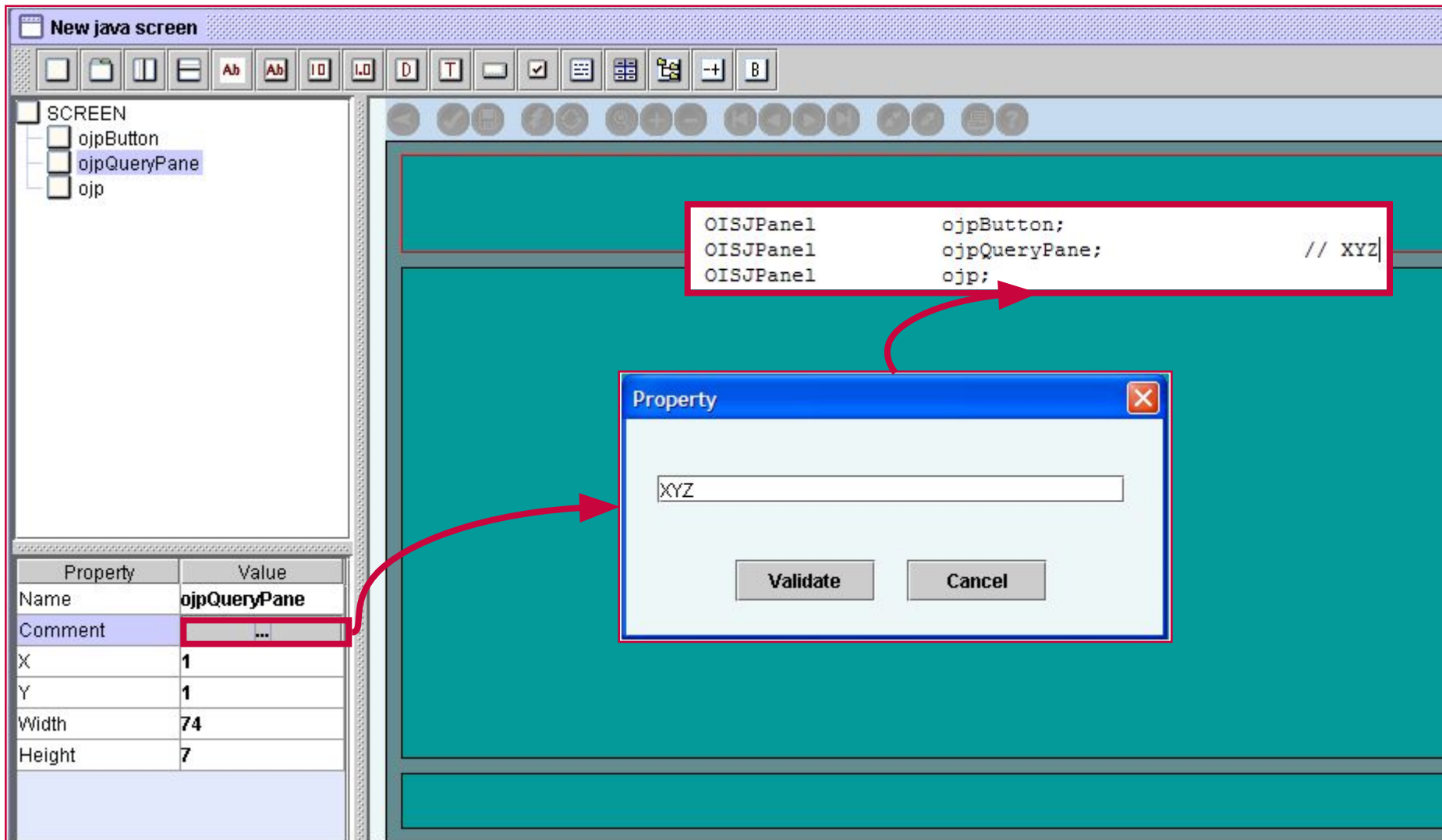
If necessary, modify properties

Property dialog box:

XYZ

Validate Cancel

3



The screenshot shows the Aldata IDE interface. On the left, a tree view shows the project structure with 'SCREEN' containing 'objButton', 'objQueryPane', and 'obj'. Below the tree is a Properties table for the selected 'objQueryPane' component.

Property	Value
Name	objQueryPane
Comment	...
X	1
Y	1
Width	74
Height	7

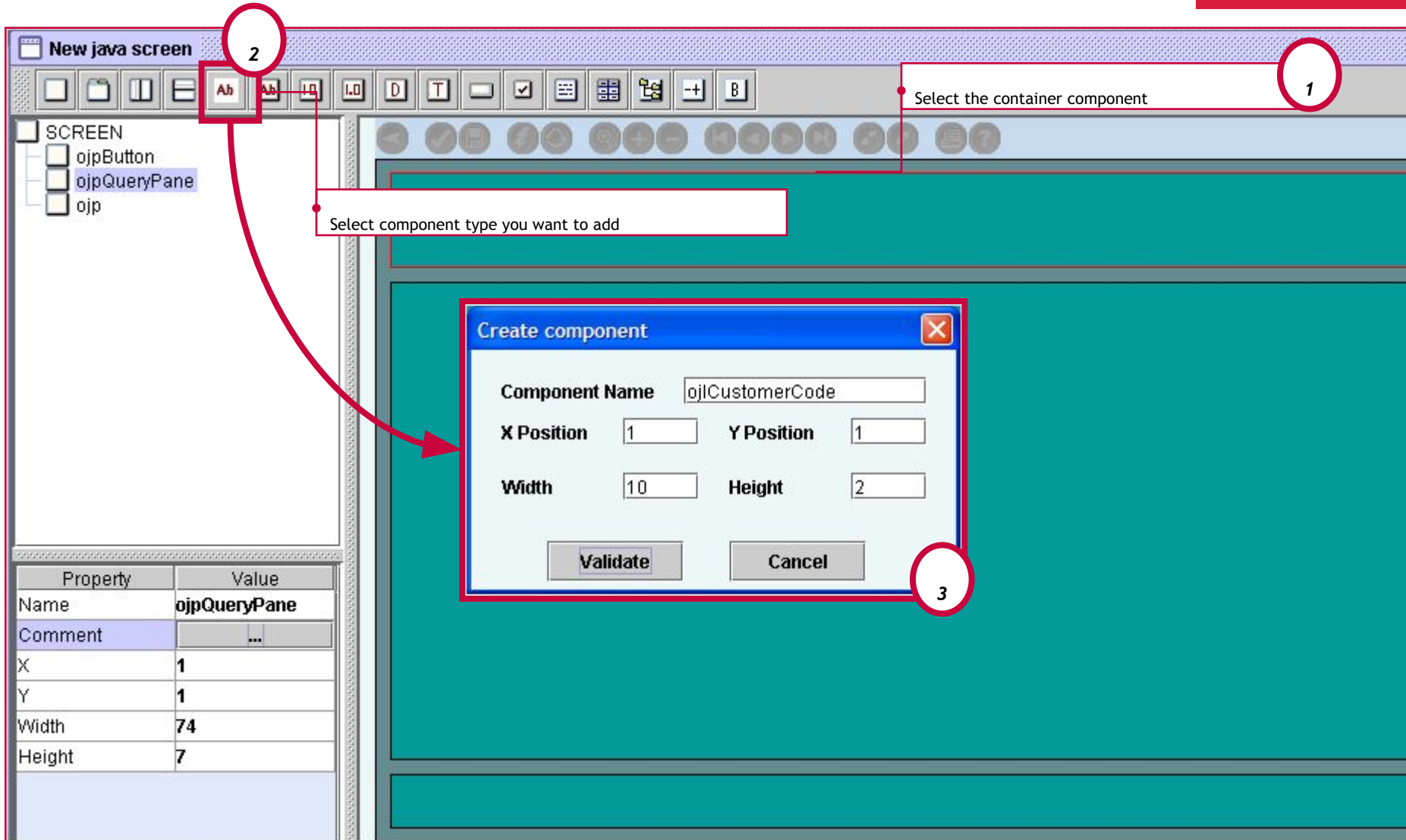
The main editor area shows a code snippet with three lines of code. A red box highlights the second line, and a red arrow points from the 'Property' dialog box to this line.

```
OISJPanel      objButton;  
OISJPanel      objQueryPane; // XYZ  
OISJPanel      obj;
```

The 'Property' dialog box is open, showing the 'Comment' field with 'XYZ' entered. The 'Validate' and 'Cancel' buttons are visible at the bottom of the dialog.

Adding components

4



1 Select the container component

2 Select component type you want to add

3

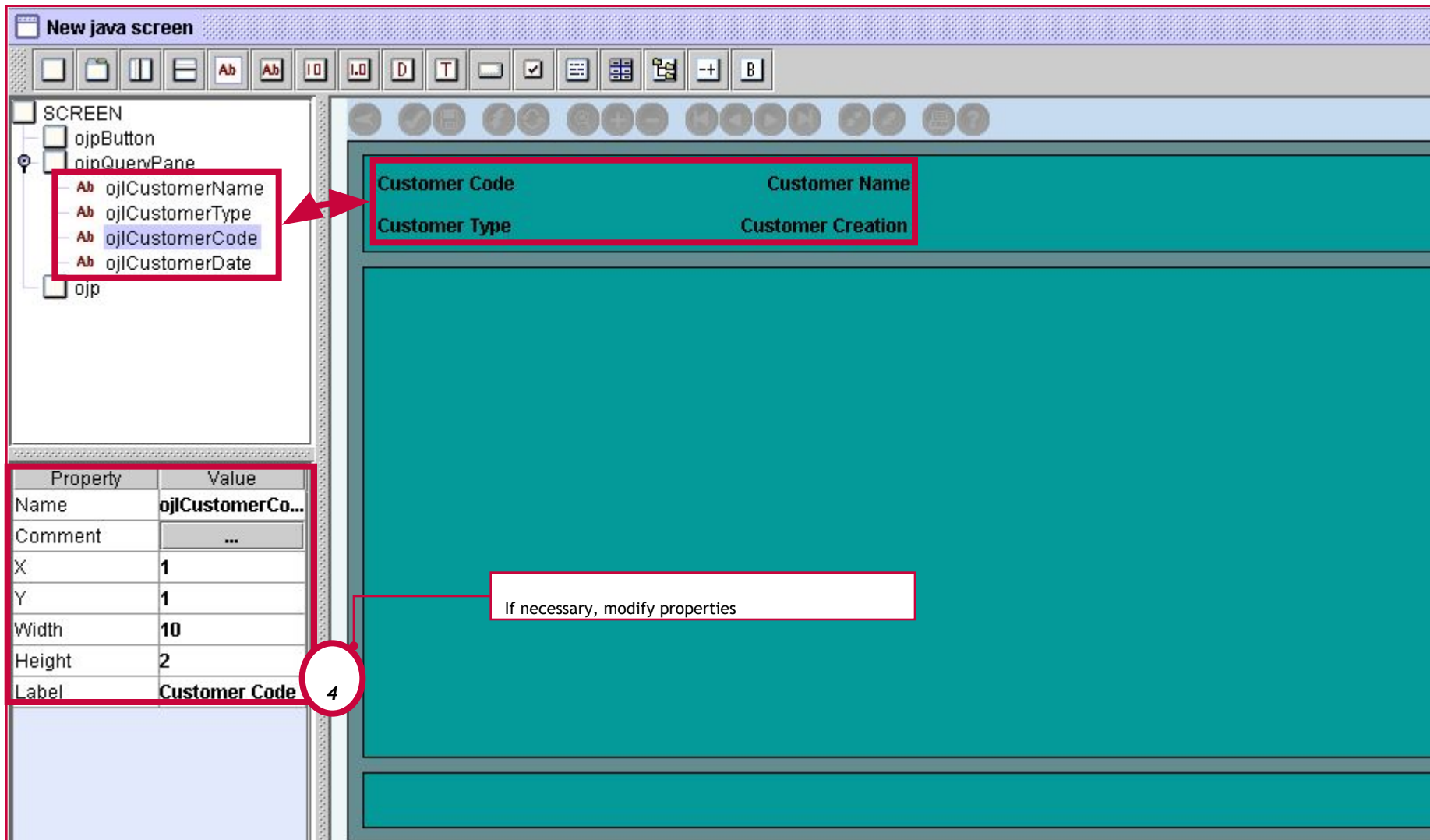
Create component

Component Name

X Position Y Position

Width Height

Property	Value
Name	ojpQueryPane
Comment	...
X	1
Y	1
Width	74
Height	7



The screenshot shows the Aldata IDE interface for creating a new Java screen. On the left, a tree view shows the component hierarchy under 'SCREEN', with 'ojlCustomerCode' selected. Below the tree is a property table for the selected component.

Property	Value
Name	ojlCustomerCo...
Comment	...
X	1
Y	1
Width	10
Height	2
Label	Customer Code

The main workspace displays a table with the following content:

Customer Code	Customer Name
Customer Type	Customer Creation

A callout box with the text "If necessary, modify properties" points to the property table. A red circle with the number "4" is located at the bottom of the property table.

New java screen

SCREEN

- ojpButton
- ojpQueryPane
 - osfNomClient
 - oijCustomerName
 - osfCodeClient**
 - oijCustomerType
 - oijCustomerCode
 - osfLOVCustomerType
 - oijCustomerDate
 - ojdaCustomerCreation
- ojp

Property	Value
Name	osfCodeClient
Comment	...
X	11
Y	1
Width	10
Height	2
Query_Name	CUSTOMER_CODE
CheckPerformed	<input type="checkbox"/>
Mandatory	<input type="checkbox"/>
Field_size	
Case_type	UPPERCASE

Customer Code Customer Name

Customer Type Customer Creation

To create a query field, enter here the query label (used in the SQL request)

Should be controlled by checkListener

Mandatory field

Max number of characters in field

UPPERCASE, LOWERCASE, ANYTHING

Lists of Values (LOV)

New java screen

SCREEN

- ojpButton
- ojpQueryPane
 - osfNomClient
 - oijCustomerName
 - osfCodeClient
 - oijCustomerType
 - oijCustomerCode
 - osfLOVCustomerType**
 - oijCustomerDate
 - ojdaCustomerCreation
- ojp

Customer Code Customer Name

Customer Type Customer Creation

Property	Value
Name	osfLOVCustomerType
Comment	...
X	11
Y	4
Width	10
Height	2
Query_Name	
CheckPerformed	<input type="checkbox"/>
Mandatory	<input type="checkbox"/>
Field_size	
Case_type	ANYTHING

Same properties as text field

New java screen

SCREEN

- ojpButton
- ojpQueryPane
 - osfNomClient
 - ojlCustomerName
 - osfCodeClient
 - ojlCustomerType
 - ojlCustomerCode
 - osfLOVCustomerType
 - ojlCustomerDate
 - ojdaCustomerCreation**
- ojp

Property	Value
Name	ojdaCustomerCreation
Comment	...
X	36
Y	4
Width	10
Height	2
Query_Name	CUSTOMER_DATE
CheckPerformed	<input type="checkbox"/>
Mandatory	<input type="checkbox"/>

Customer Code Customer Name

Customer Type Customer Creation

New java screen

SCREEN

- objButton
- objQueryPane
- obj
 - objClient

Property	Value
Name	objClient
Comment	...
X	1
Y	1
Width	72
Height	33
Columns_Name	...

Customer Code Customer Name

Customer Type Customer Creation

Code Client	Nom Client	Type Client	Date creation
<div data-bbox="730 706 1449 1056"> <p>Property</p> <p>Code Client;Nom Client;Type Client;Date creation</p> <p>Validate Cancel</p> </div>			

Print report

Button



The screenshot shows a Java IDE window titled "New java screen". On the left, a tree view shows a hierarchy: SCREEN > ojbButton > ojbEdit. The ojbEdit component is selected, and its properties are shown in a table below:

Property	Value
Name	ojbEdit
Comment	...
X	33
Y	1
Width	10
Height	2
Label	Print report
Action_Command	

The main workspace shows a form with fields for "Customer Code", "Customer Name", "Customer Type", and "Customer Creation". Below these is a table with columns "Code Client", "Nom Client", and "Type Client". A button labeled "Print report" is located at the bottom right of the workspace. Callouts point to the button's label and action command.

Text on Button

Button JAVA name (used by actionPerformed)

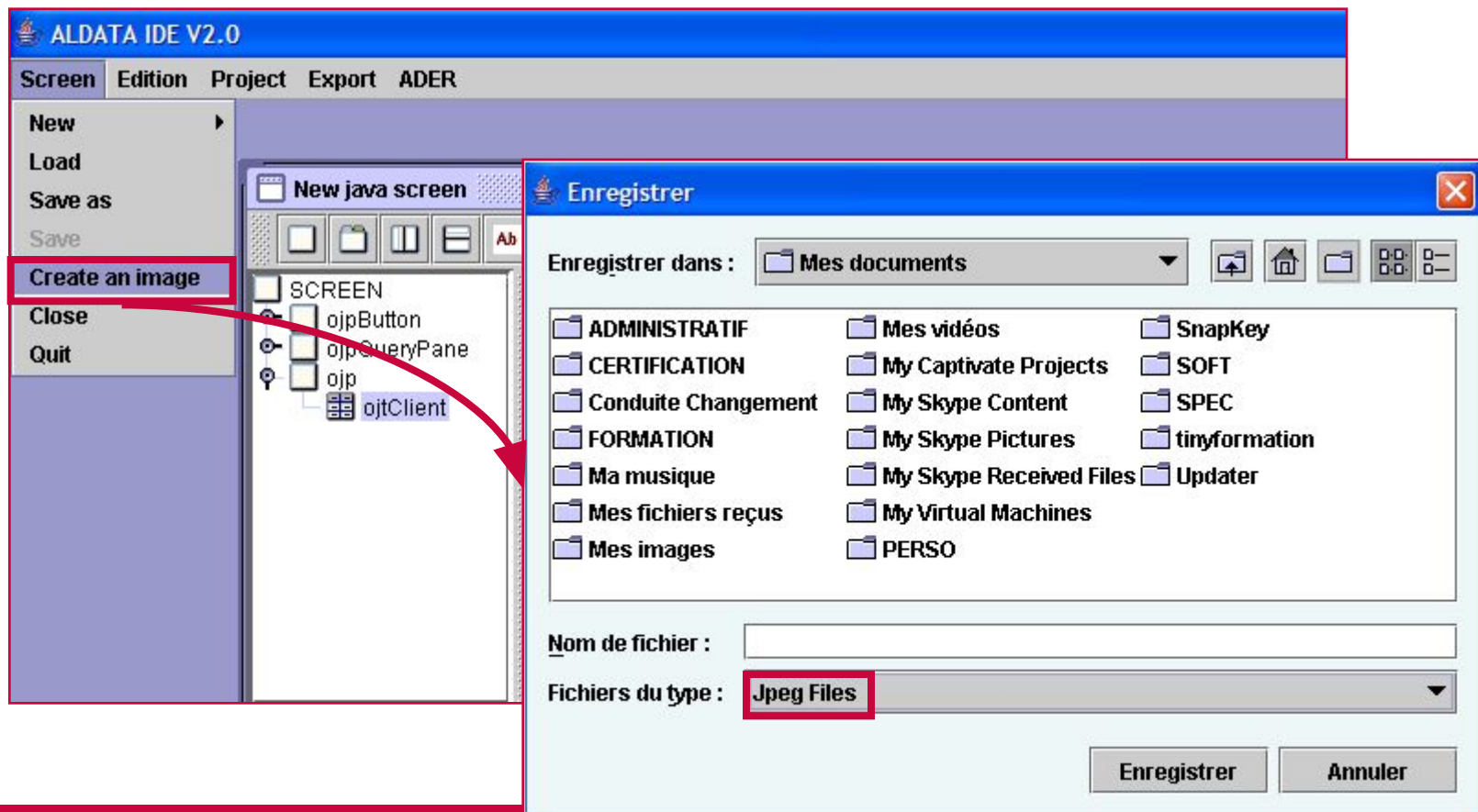
Delete a component



The screenshot shows the ALDATA IDE V2.0 interface. On the left, the project tree shows a component named 'osfCodeClient' which is circled with a red circle and the number '1'. A red arrow points from this component to a 'Customer Code' field in the main design area, which is also circled with a red circle. The 'Delete' menu item in the 'Edition' menu is circled with a red circle and the number '2'. A callout box on the right contains the text 'You can also Right-click on object and delete' and shows a context menu with 'Delete' highlighted.

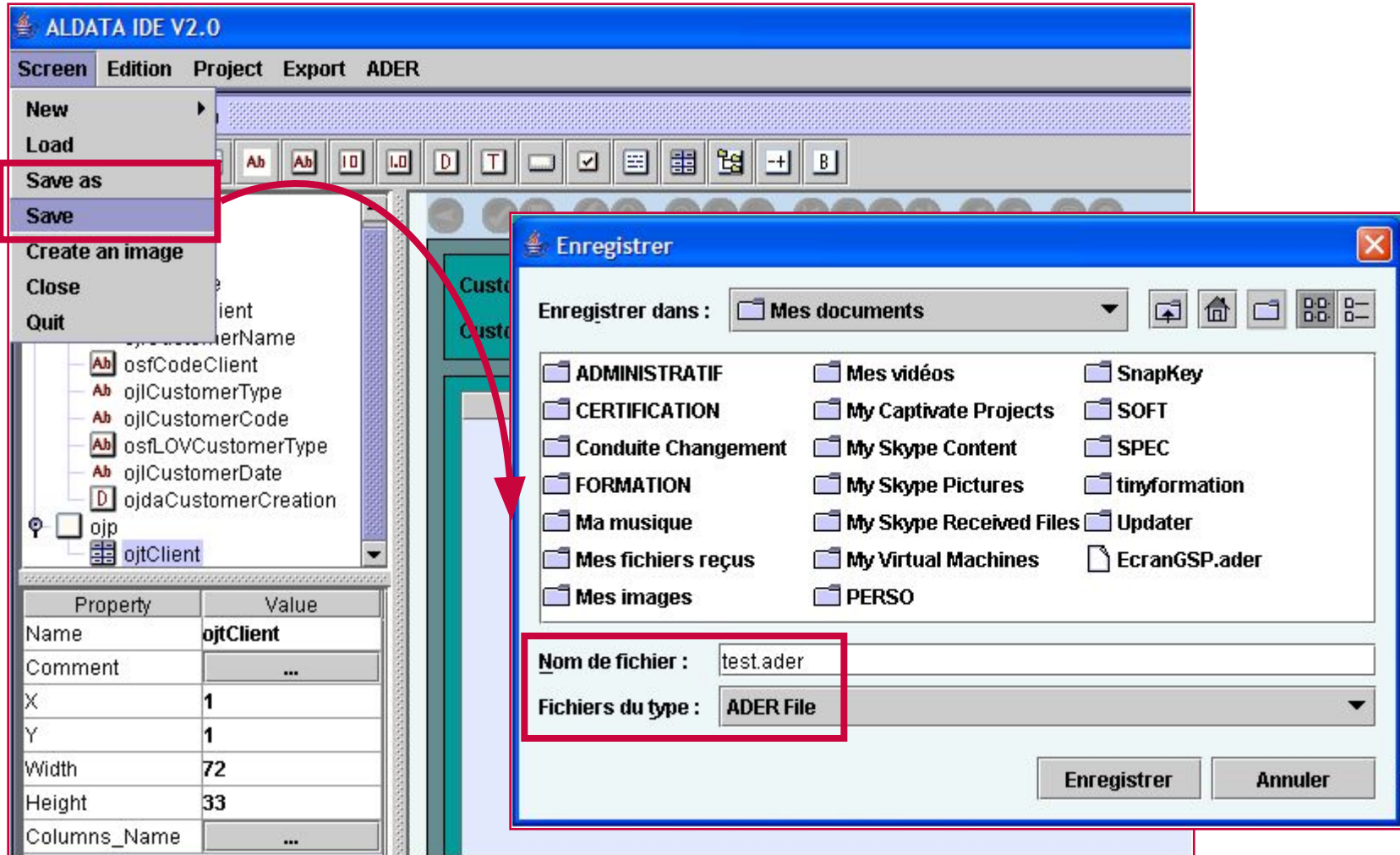
- Use ADER to print mock-up.

 **Tip !** Use image to get screen validated before you export and start coding !



Save Project

- Save project as .ADER before exporting so you can modify your screen if necessary.



The screenshot shows the ALDATA IDE V2.0 interface. The 'Screen' menu is open, and the 'Save as' option is highlighted. A red arrow points from 'Save as' to the 'Enregistrer' dialog box. The dialog box shows the file name 'test.ader' and the file type 'ADER File' selected. The 'Enregistrer' dialog box is titled 'Enregistrer' and shows the current directory 'Mes documents'. It lists various folders and files, including 'ADMINISTRATIF', 'CERTIFICATION', 'Conduite Changement', 'FORMATION', 'Ma musique', 'Mes fichiers reçus', 'Mes images', 'Mes vidéos', 'My Captivate Projects', 'My Skype Content', 'My Skype Pictures', 'My Skype Received Files', 'My Virtual Machines', 'PERSON', 'SnapKey', 'SOFT', 'SPEC', 'tinyformation', 'Updater', and 'EcranGSP.ader'. The 'Nom de fichier' field contains 'test.ader' and the 'Fichiers du type' dropdown is set to 'ADER File'. The 'Enregistrer' and 'Annuler' buttons are at the bottom right.

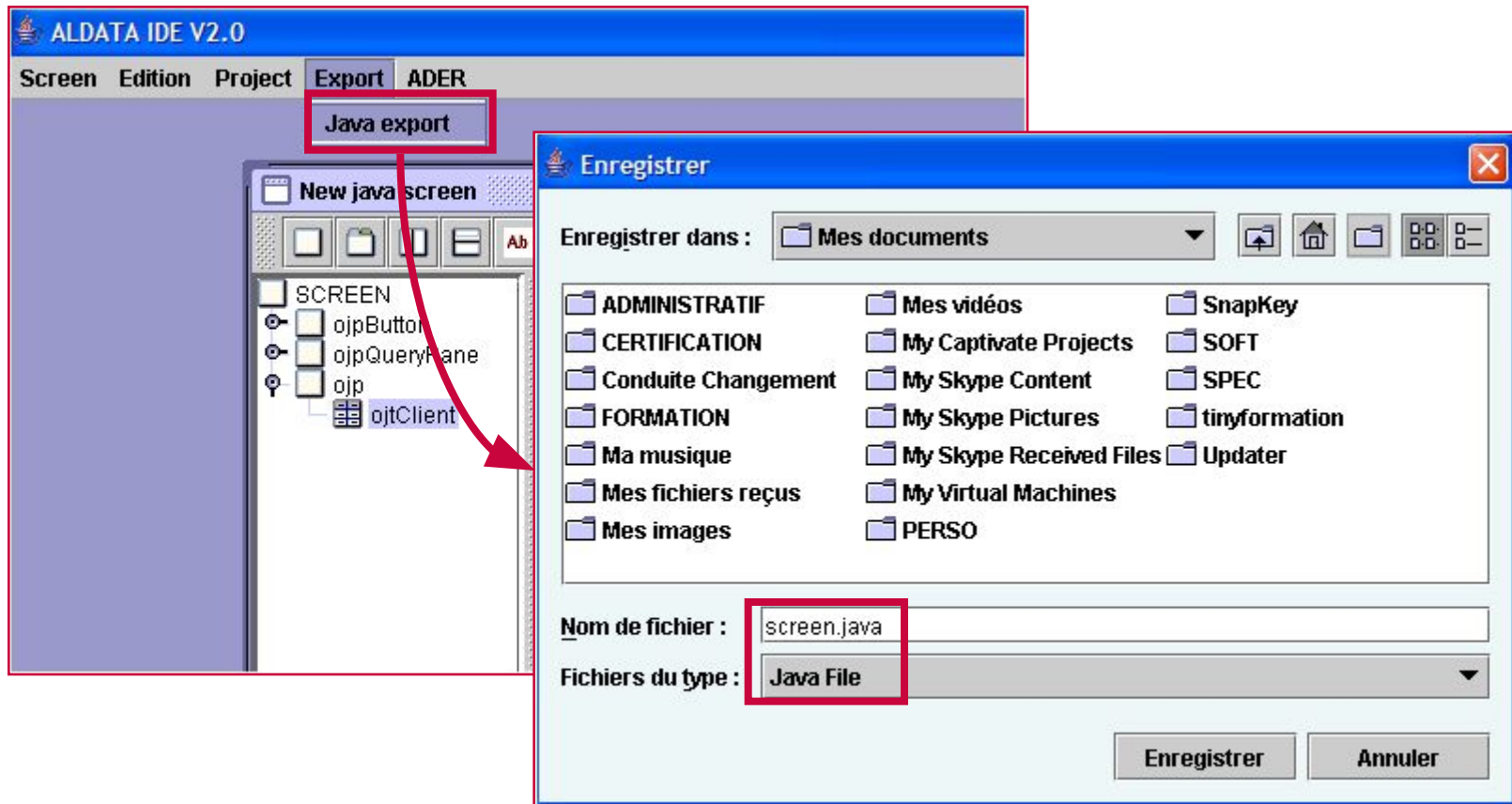
JAVA Export

6

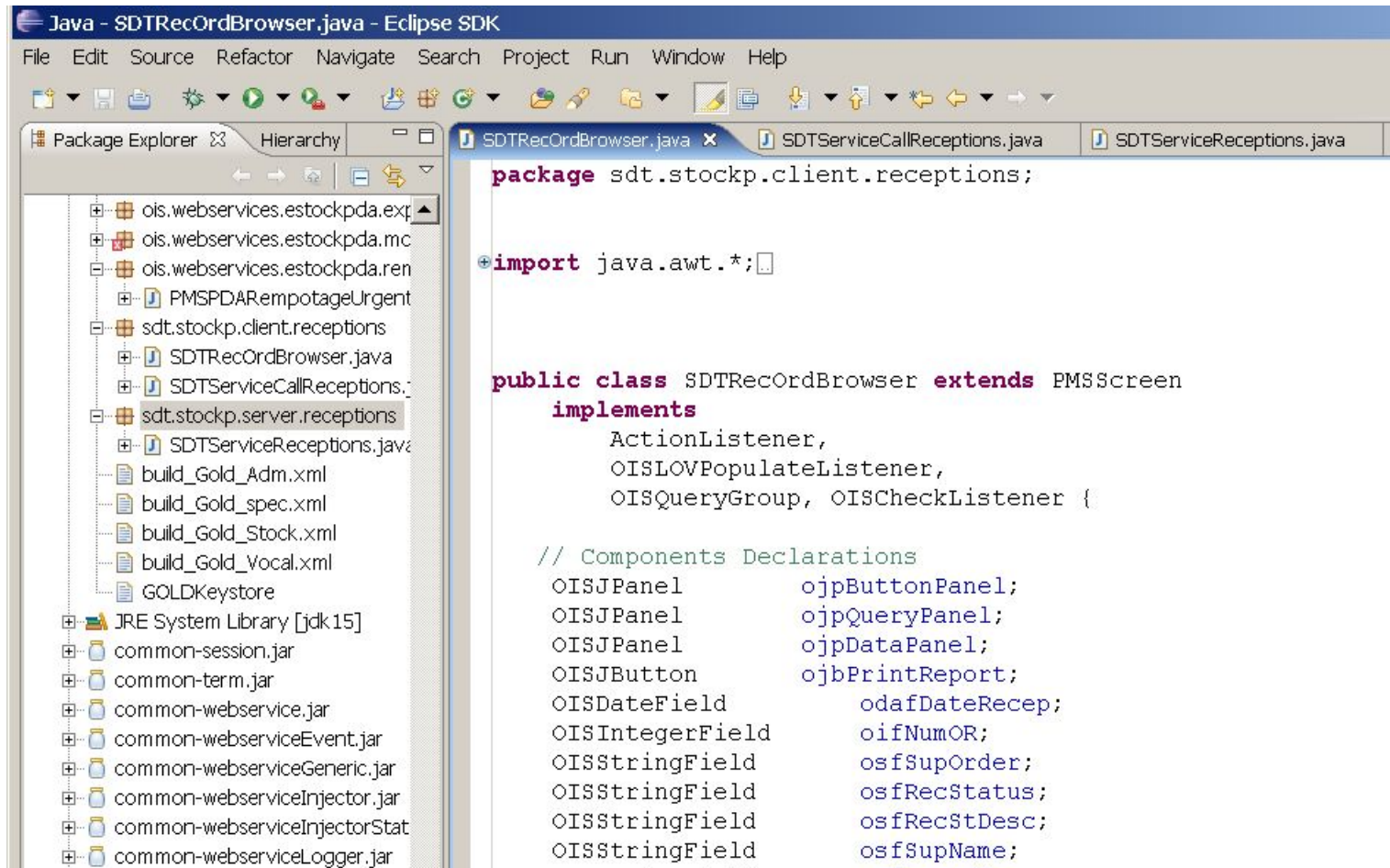


Project properties must be set-up before export.

- If you forgot to set them up, the properties window will pop-up when you click on export.



- Export java source code to directory that match package name



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project structure with a package named `sdt.stockp.client.receptions`. The main editor window shows the source code for `SDTRecOrdBrowser.java`, which is part of the `sdt.stockp.client.receptions` package. The code includes an import for `java.awt.*` and a class declaration for `SDTRecOrdBrowser` that extends `PMSScreen` and implements `ActionListener`, `OISLOVPopulateListener`, and `OISQueryGroup`. The code also contains several component declarations.

```

package sdt.stockp.client.receptions;

import java.awt.*;

public class SDTRecOrdBrowser extends PMSScreen
    implements
        ActionListener,
        OISLOVPopulateListener,
        OISQueryGroup, OISCheckListener {

    // Components Declarations
    OISJPanel        ojpButtonPanel;
    OISJPanel        ojpQueryPanel;
    OISJPanel        ojpDataPanel;
    OISJButton       ojbPrintReport;
    OISDateField     odafDateRecep;
    OISIntegerField  oifNumOR;
    OISStringField  osfSupOrder;
    OISStringField  osfRecStatus;
    OISStringField  osfRecStDesc;
    OISStringField  osfSupName;
  
```



EXERCISE: Create a screen

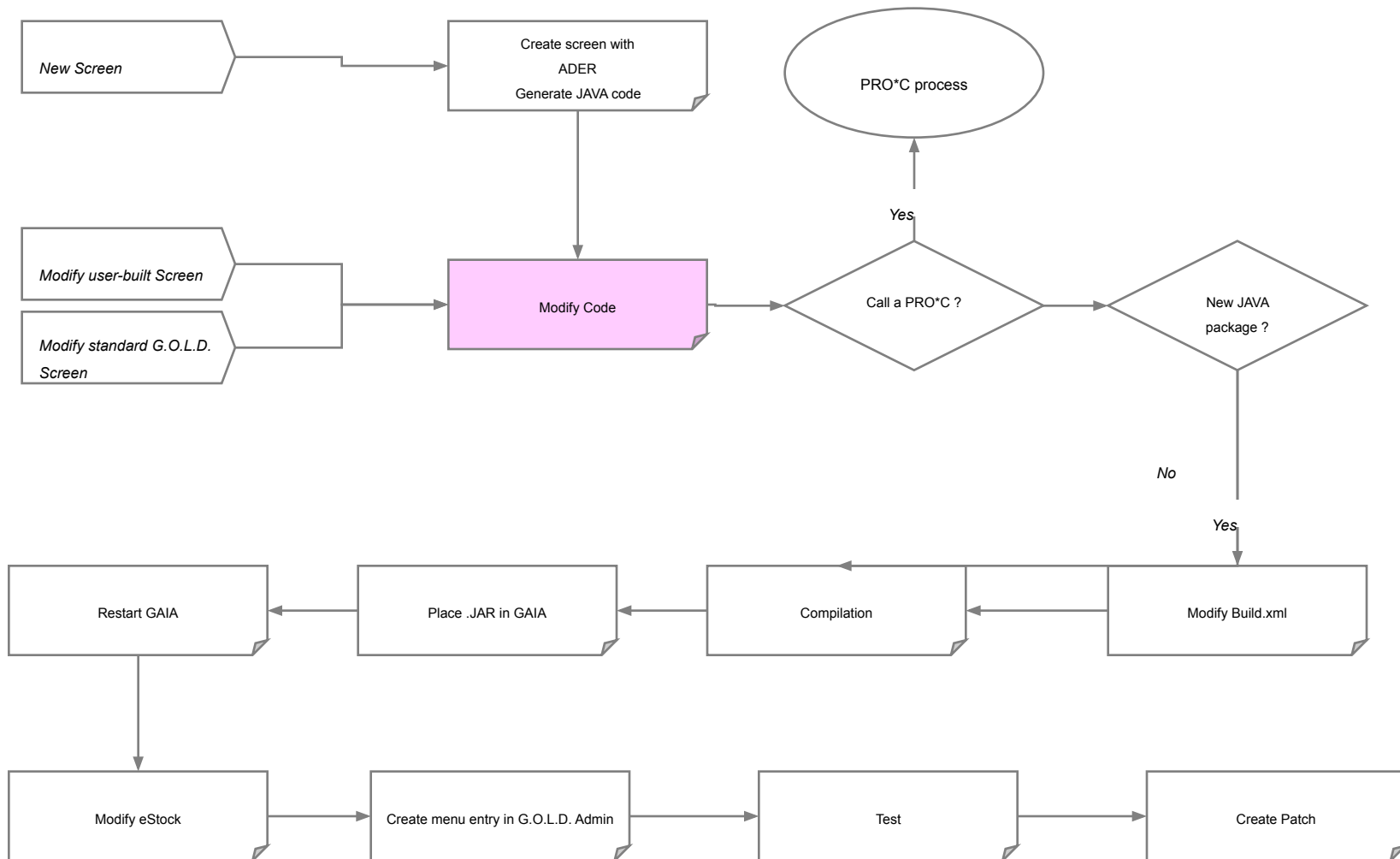
- Open the following file for more details on this exercise

TS200_GOLDStockDevelopment_Exercises - 3

1. Launch Ader
2. Set project properties
3. Add Container Components
4. Add Components
5. Create an image
6. Export to JAVA



Create a screen process



Modify Code – Implement logic



- Replace hard coded labels
getLabel(613,"Advanced Goods Receiving Note management")
- Add constraints on enterable fields
addChangeListener(this);
checkPerformed
- Add and implement LOVs
LOVPopulate
- Implement actions on buttons
actionPerformed
- Implement logic to load data from database
searchAction

Modify Code – Service call



- Client
 - Java screen

```
// Execute Search
public int searchAction()
{
    HashMap param = OISTools.makeDynamicParam(ojpQueryPanel);

    sendTrace("Lancement du service tournée.getListeTournéeBase");
    ojtRecOrders.setData(SDTServiceCallReceptions.getRecOrderList(oja, param, oja.sLangueGOLD));
    if (ojtRecOrders.getData() == null)
    {
        oja.showMessageDialog(getLabel(139, "No data found"));
    }
    return 0;
}
```

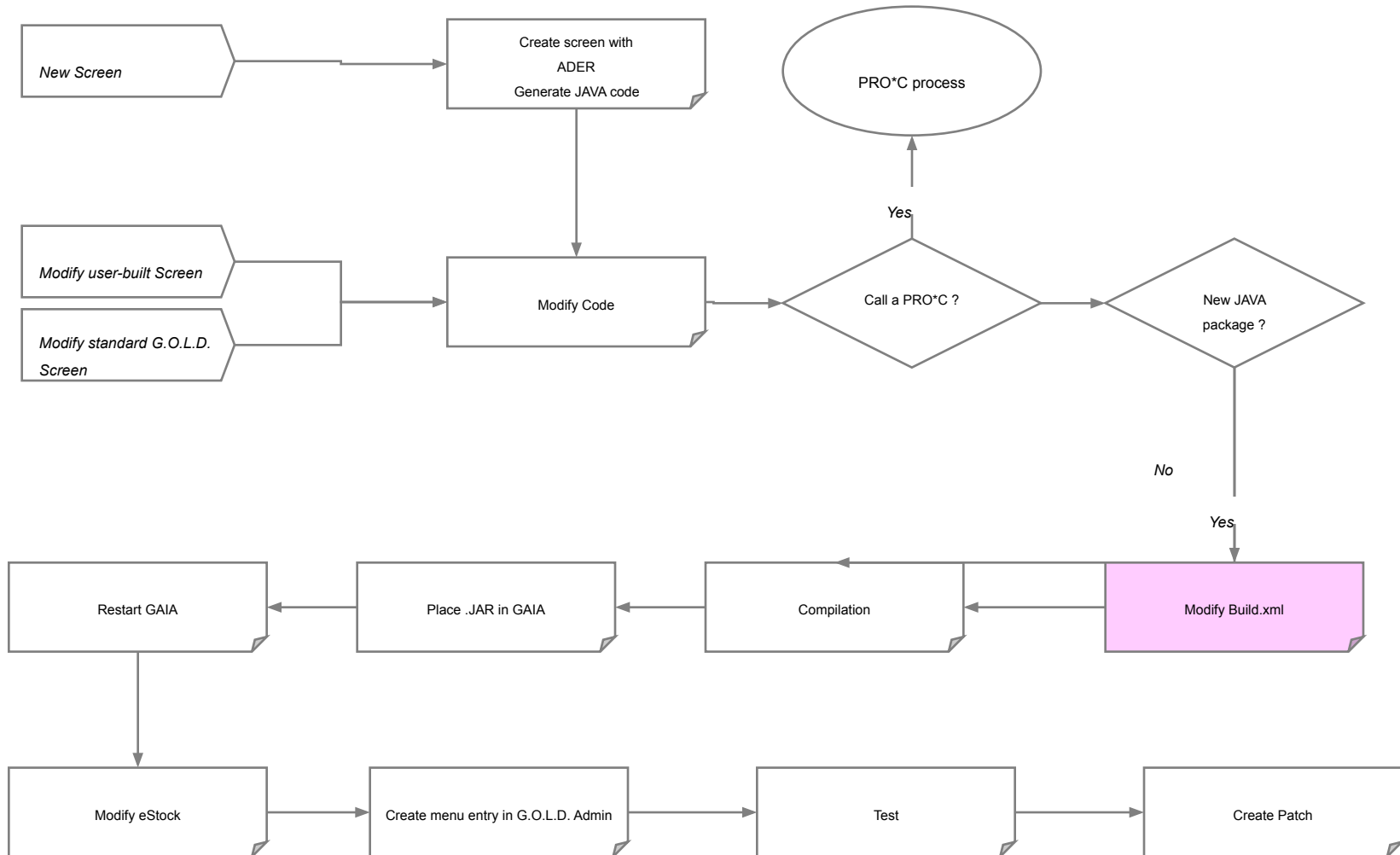
- Service call

```
final static public Object getRecOrderList(OISJApplet oja, HashMap hmp, String sLang)
{
    return oja.getData("sdt.stockp.server.receptions.SDTServiceReceptions.getRecOrderList",
        new Object[]{hmp, sLang});
}
```

- Server

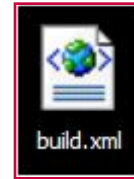
```
public class SDTServiceReceptions implements Invokable
{
    static public Object[] getRecOrderList(HashMap hmp, String sLang) throws Exception
    {
        Log.debug("getRecOrderList...");
    }
}
```

Create a screen process



Build.XML structure – 1/2

- The file build.xml is used to compile JAR files
- Structure of build.xml



- Variables

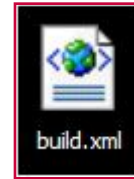
```
<!-- ===== -->
<!-- Path properties -->
<!-- ===== -->
<property environment="env"/>
<property name="GaiaRoot" value="${env.GAIA_HOME}"/>
<property name="GaiaNamespace" value="${env.NAMESPACE}"/>
<property name="GoldVob" value="${env.SOURCEVOB}"/>
<property name="GoldSource" value="${GoldVob}/java"/>
<property name="ClassDirectory" value="${env.COMP_JAVA}/class"/>
<property name="GoldResource" value="${env.EGOLD_INT}/resource"/>
<property name="GoldWeb" value="${env.EGOLD_INT}/web"/>
<property name="PatchLevel" value="${env.PATCH_LEVEL}"/>
```

- Class Path

```
<path id="GOLD_CLASSPATH">
  <pathelement location="${GaiaRoot}/lib/gas.jar"/>
  <pathelement location="${GaiaRoot}/lib/ext/api/jdom.jar"/>
  <pathelement location="${GaiaRoot}/rt/j2ee14/jmx/jmxri.jar"/>
  <pathelement location="${GaiaRoot}/rt/tomcat/servlet-api.jar"/>
  <pathelement location="${GaiaRoot}/deploy/${GaiaNamespace}/resource/common/lib/beanapi.jar"/>
  <pathelement location="${GaiaRoot}/deploy/${GaiaNamespace}/resource/common/bean/common-term.jar"/>
  <pathelement location="${GaiaRoot}/deploy/${GaiaNamespace}/resource/common/bean/common-session.jar"/>
```


Build.XML structure – 2/2

- The file build.xml is used to compile JAR files



- Structure of build.xml

- Build Client

```
<macrodef name="build_client_jar">
  <attribute name="JarName" default=""/>
  <attribute name="IncludedClass" default=""/>
  <sequential>
    <delete file="@{JarName}" verbose="true"/>
    <jar destfile="@{JarName}"
      basedir="${ClassDirectory}"
      includes="@{IncludedClass}">
      <manifest>
        <attribute name="Patch-Level" value="${PatchLevel}"/>
      </manifest>
    </jar>
    <signjar jar="@{JarName}"
      alias="${KeyName}" storepass="${KeyPass}" keypass="${KeyPass}"
      keystore="${KeyFile}"/>
  </sequential>
</macrodef>
```

- Sign JAR

```
<macrodef name="signature">
  <attribute name="JarName" default=""/>
  <sequential>
    <signjar jar="@{JarName}"
      alias="${KeyName}" storepass="${KeyPass}" keypass="${KeyPass}"
      keystore="${KeyFile}"/>
  </sequential>
</macrodef>
```

Modify build.xml

- Create a new <target> in build.xml when you create a new package

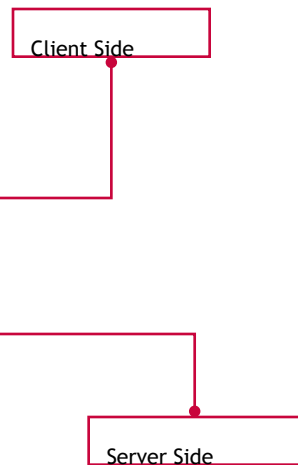
```

<target name="buildspec">
  <echo>Compile specific files</echo>
  <javac destdir="${ClassDirectory}" fork="yes" failonerror="no" debug="yes"
    debuglevel="lines,vars,source" memoryMaximumSize="512M"
    source="1.4" target="1.4">
    <classpath refid="GOLD_CLASSPATH"/>
    <src path="${GoldSource}/sdt"/>
  </javac>

  <echo>SDT Client jar files</echo>
  <build_client_jar JarName="${GoldWeb}/estock/sdt.client.jar"
    IncludedClass="sdt/stockp/client/**"/>

  <echo>SDT Server jar files</echo>
  <build_server_jar JarName="${GoldResource}/estock/lib/sdt.server.jar"
    IncludedClass="sdt/stockp/server/**"/>
</target>

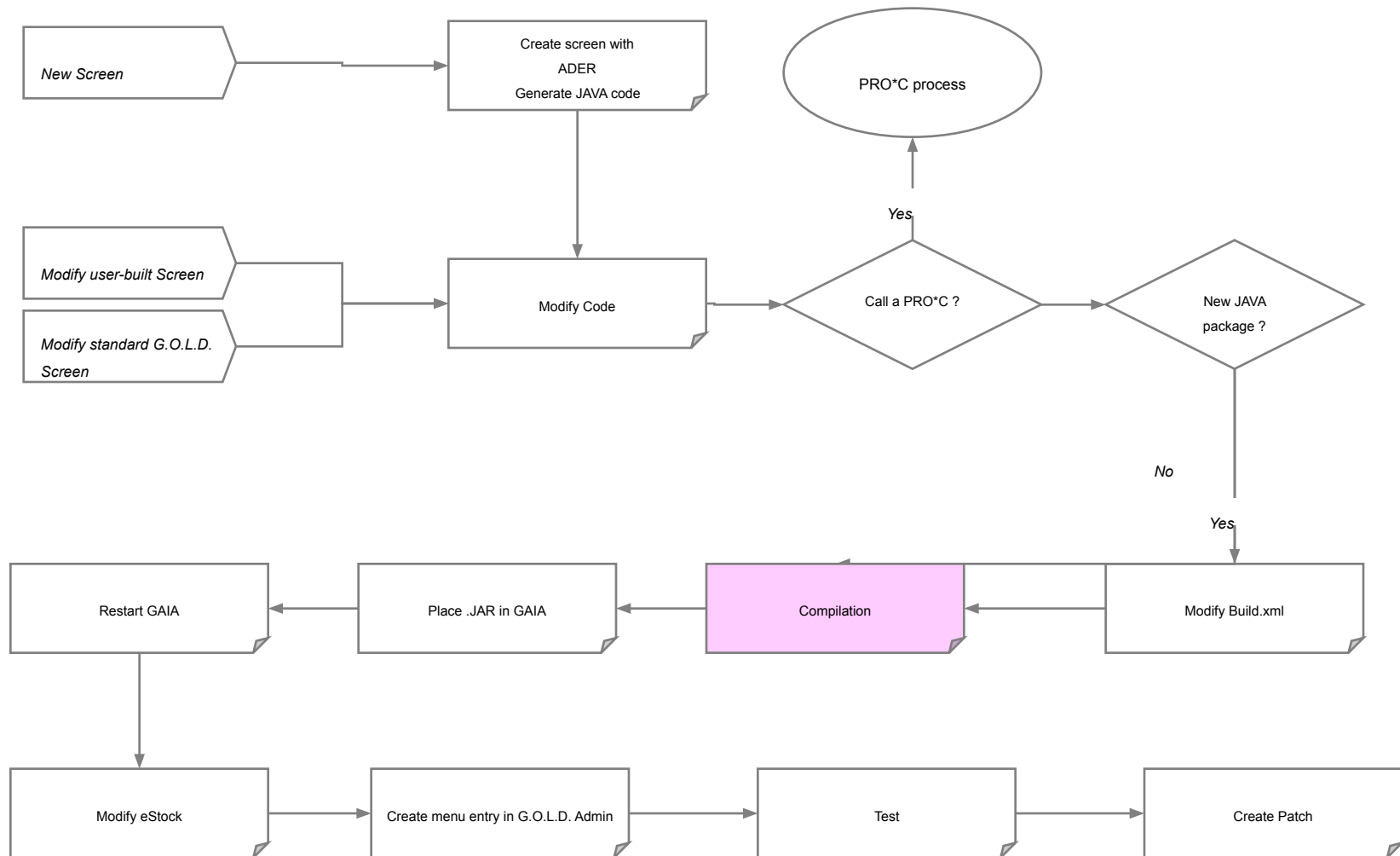
```



Note :

- JAR files (destfile) are named :
 - On client side ois.clinet.XXX.jar
 - On server Side : ois.server.XXX.jar

Create a screen process



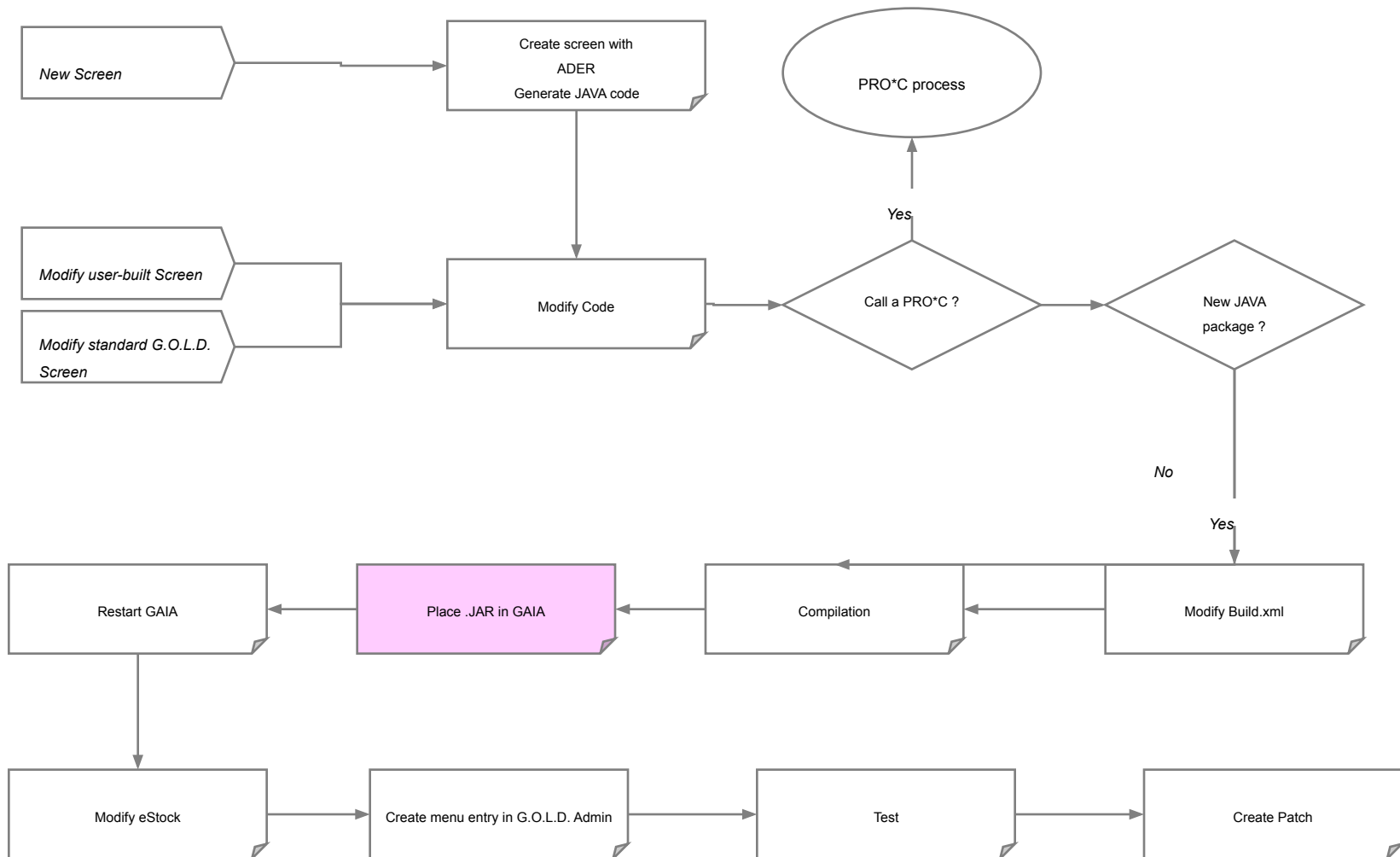
Compilation



- Connect to an Unix
- Compile (launch build.xml)
 - ant [-f buildfile] Target_Name

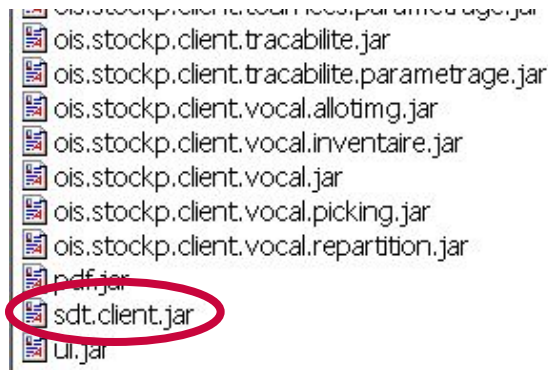
```
[egold@DevLinux java]$  
[egold@DevLinux java]$ ant -f build_Gold_Stock.xml buildspec  
Buildfile: build_Gold_Stock.xml  
  
buildspec:  
  [echo] Compile All Source files  
  [javac] Compiling 3 source files to /opt/GOLD/stk507/src/java/class  
  [echo] SDT Client jar files  
  [delete] Deleting: /opt/GOLD/stk507/gaia/deploy/DEFAULT/web/estock/sdt.  
  [jar] Building jar: /opt/GOLD/stk507/gaia/deploy/DEFAULT/web/estock/s  
  [signjar] Signing JAR: /opt/GOLD/stk507/gaia/deploy/DEFAULT/web/estock/s  
  [signjar] Enter Passphrase for keystore:  
  [echo] SDT Server jar files  
  [delete] Deleting: /opt/GOLD/stk507/gaia/deploy/DEFAULT/resource/estock  
  [jar] Building jar: /opt/GOLD/stk507/gaia/deploy/DEFAULT/resource/es  
  
BUILD SUCCESSFUL
```

Create a screen process

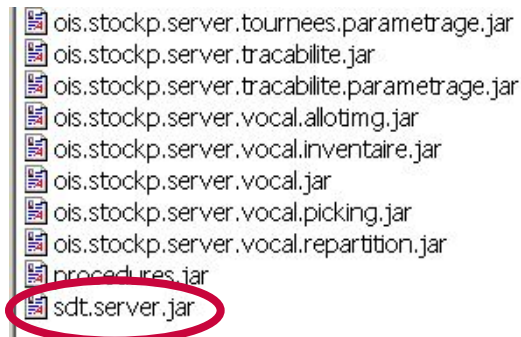


Place .JAR files

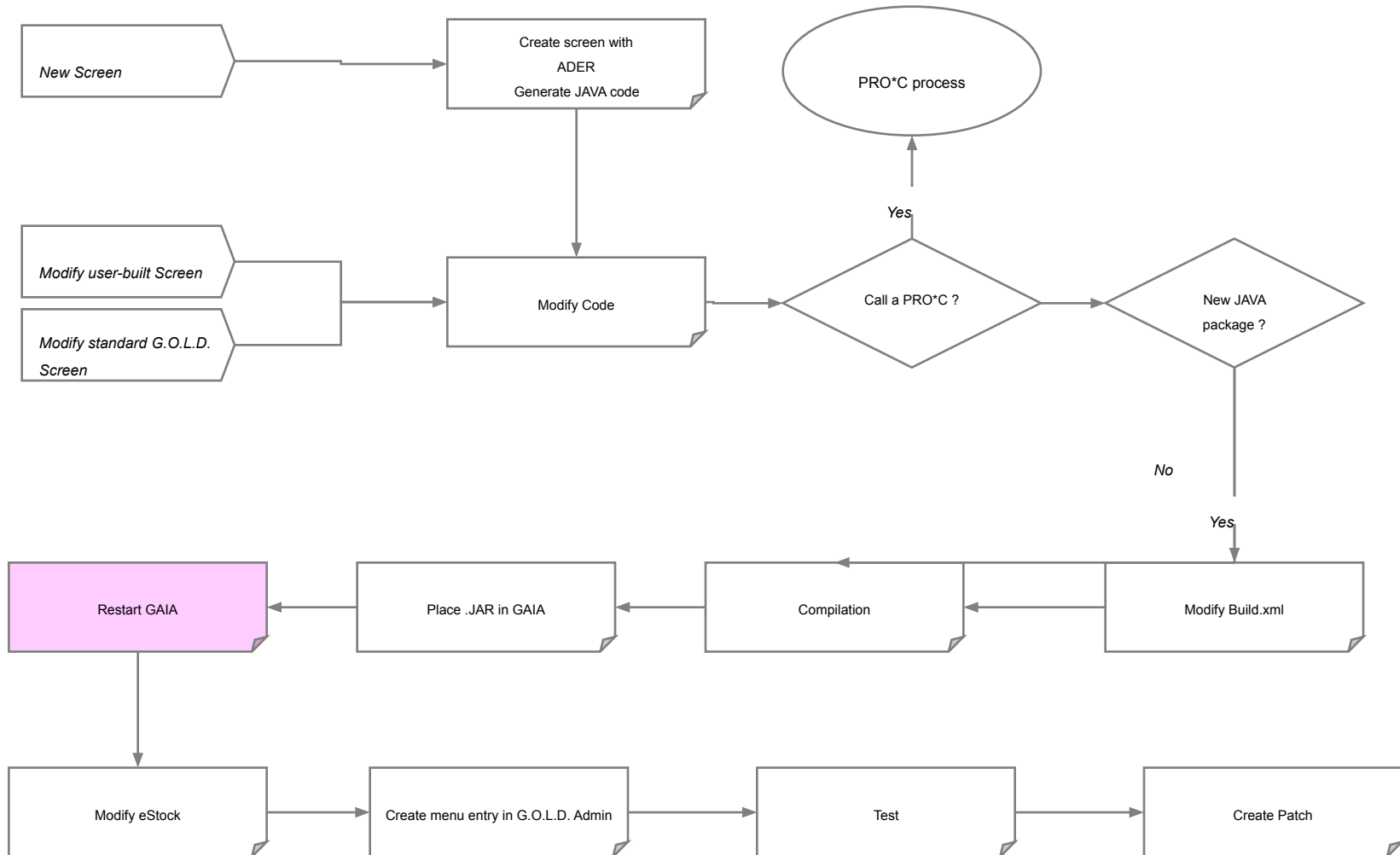
- Client Side :
 - \$GOLD_HOME\gaia\deploy\DEFAULT\web\estock



- Server Side
 - \$GOLD_HOME\gaia\deploy\DEFAULT\resource\estock\lib



Create a screen process



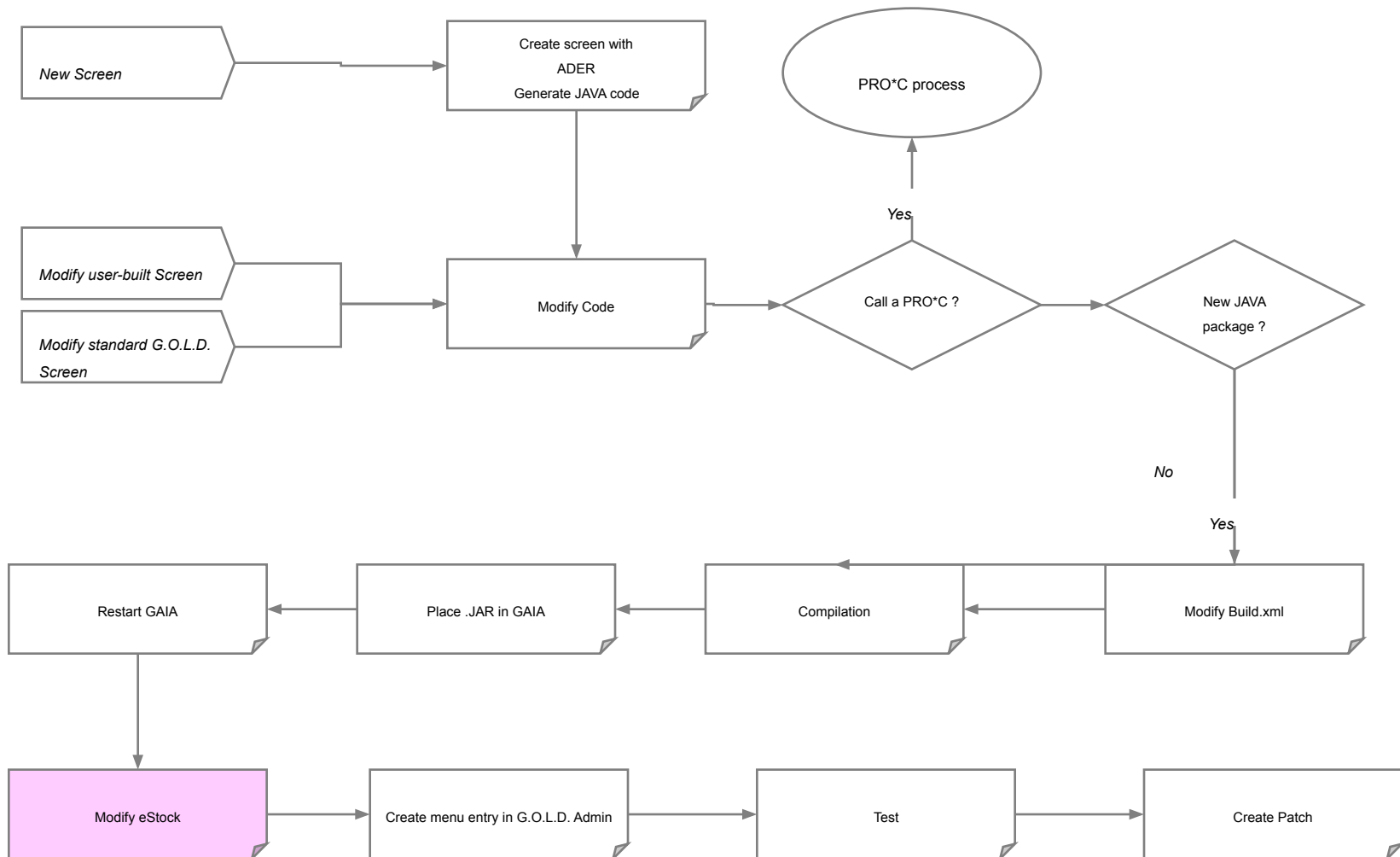
Restart GAIA



- Connect to an Unix and go to the GAIA folder with `cd $GAIA_HOME`
- `show_gaia` to check node's name
- To stop a node in GAIA :
 - `stop_gaia node_name`
- To start a node in GAIA :
 - `start_gaia node_name`

```
[egold@DevLinux stk507]$ cd $GAIA_HOME
[egold@DevLinux gaia]$ ./show_gaia
Detecting GAIA instances...
GAIA registry : <none>
GAIA launcher : <none>
GAIA is running on following node(s) :
STK507
[egold@DevLinux gaia]$ ./stop_gaia STK507
* Detected the following GAIA instance(s) : *
STK507
--- GAIA node (STK507) being stopped...
    GAIA node (STK507) stopped !
[egold@DevLinux gaia]$ ./start_gaia STK507
Using JAVA_HOME [/opt/java14]
Logs redirected to [./log_STK507.txt]
Starting GAIA Node (STK507) ...
** GAIA (STK507) is now running **
[egold@DevLinux gaia]$
```


Create a screen process



Modify estock.html

- Add .jar call to eStock.html

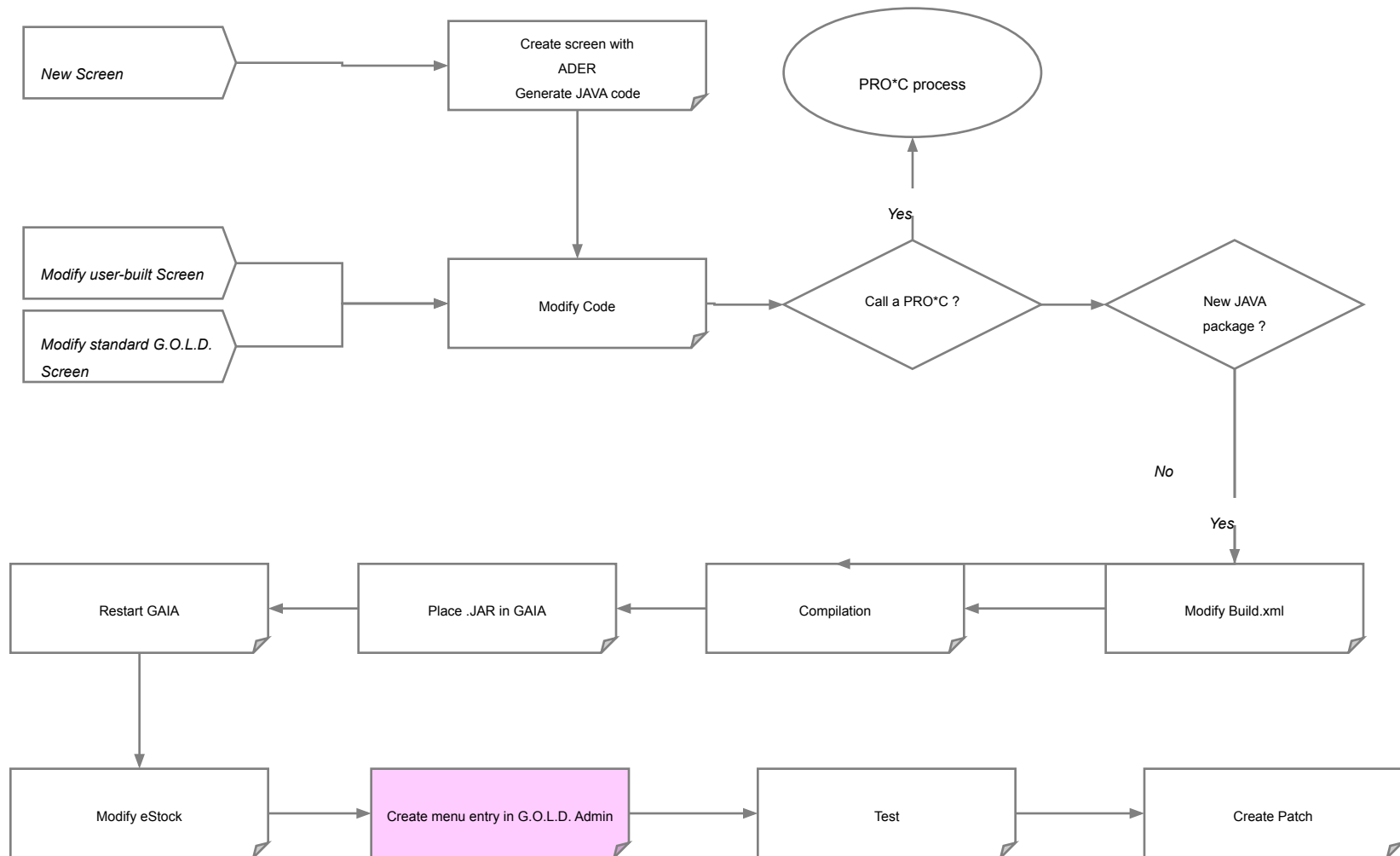


estock.html

```
<PARAM NAME="applicationServerDriver" VALUE="ois.ui.net.GAIAClientHttp">
<PARAM NAME="cache_option" VALUE="PLUGIN">
<PARAM NAME="cache_archive" VALUE="ui.jar
, sdt.client.jar
, gw.jar
, ESTOCKlabelfr_FR.jar
, ESTOCKlabelgb_GB.jar
, estock_icons.jar
, pdf.jar
, ois.stockp.client.jar
, ois.stockp.client.allotissement.jar
```

```
ENABLE_USER_PROPERTIES_SAVE="true"
cache_option="PLUGIN"
cache_archive="ui.jar
, sdt.client.jar
, gw.jar
, ESTOCKlabelfr_FR.jar
, ESTOCKlabelgb_GB.jar
, estock_icons.jar
, pdf.jar
, ois.stockp.client.jar
, ois.stockp.client.allotissement.jar
, ois.stockp.client.base.jar
, ois.stockp.client.base.administration.jar
```

Create a screen process



Create menu entry in G.O.L.D.

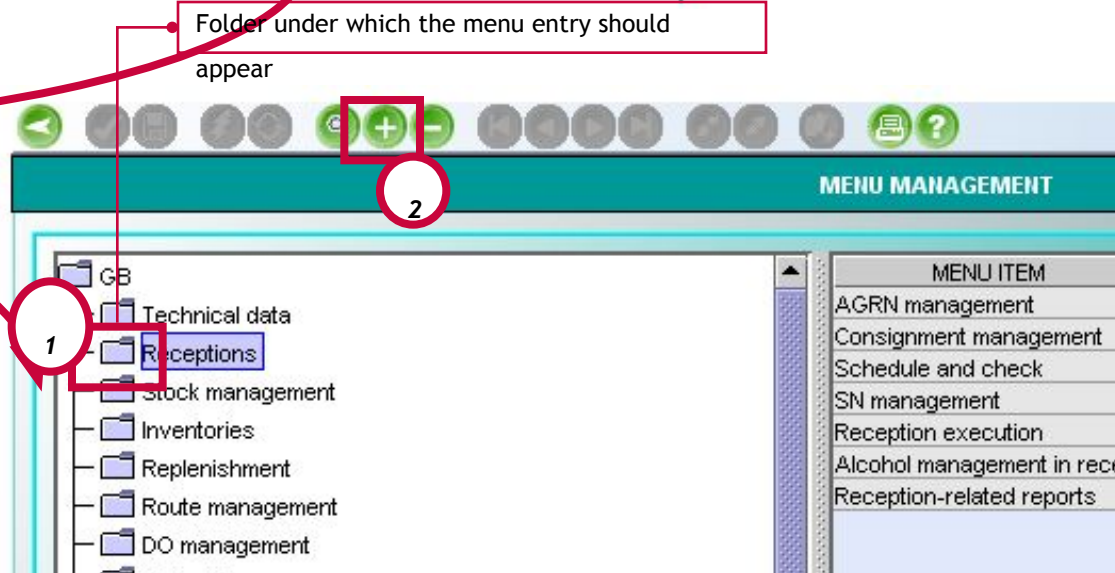
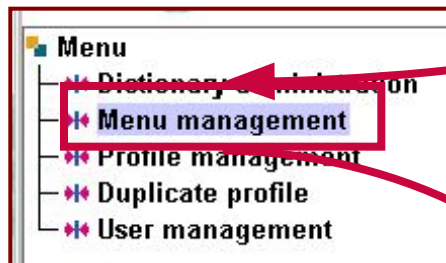
- Two ways to add menu entry :
 - Manually (see next screen)
 - SQL

```
INSERT INTO adm_menus
(ameident, ameobjet, ameparent, ameseq, ameappli, amedcre, amedmaj, ameutil)
VALUES
(52001, 'sdt.stockp.client.receptions.SDTRecOrdBrowser',
  NULL, 5201, 'ESTOCK', SYSDATE, SYSDATE, 'SCRIPT');

INSERT INTO tra_adm_menus
(tameident, tameappli, langue, tamedesc, tamedcre, tameutil, dtraduction, etraduction)
VALUES
(52001, 'ESTOCK', 'GB', 'SDT - RECEPTION STATEMENT', SYSDATE, 'SCRIPT', SYSDATE, 3);

INSERT INTO adm_profiles
(aprprof, aprdesc, aprident, apraces, aprparam, aprappli, aprdcre, aprdmaj, aprutil)
(SELECT aprprof, aprdesc, 52001, 1, '', 'ESTOCK', SYSDATE, SYSDATE, 'SCRIPT'
  FROM adm_profiles
  WHERE aprprof = 999
  AND rownum = 1);
```

Create menu entry in G.O.L.D.



Create menu entry in G.O.L.D.

G.O.L.D. Administration-MENU ITEM CREATION

MENU ITEM CREATION 16/06/09

Identificator: 20801

Menu Item: SDT - RECEPTION STATEMENT

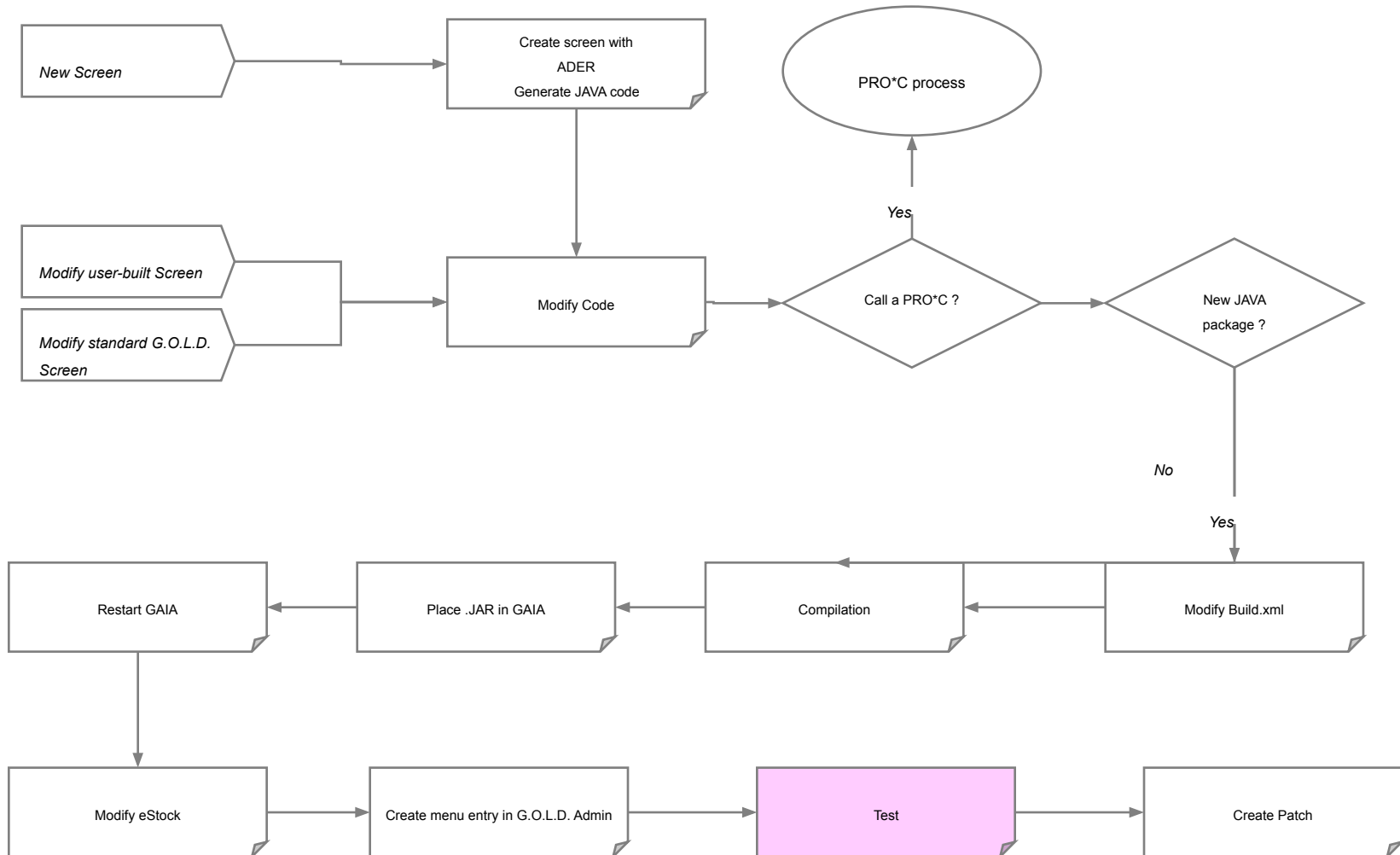
Object: sdt.stockp.client.receptions.SDTRecOrBr

Language: GB

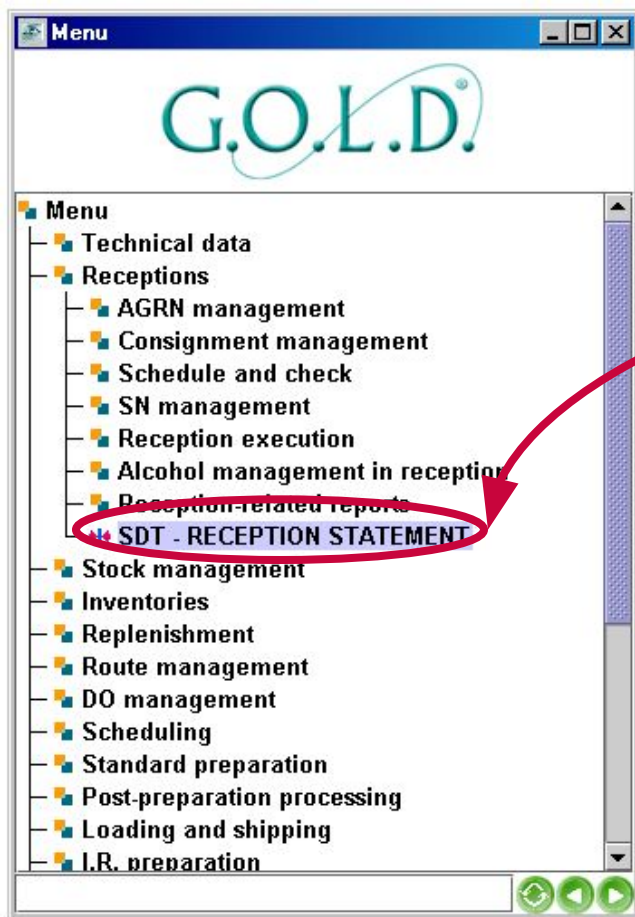
Profile	Description	Parameters	Enabled	
0000	PROFILE VIDE		<input checked="" type="checkbox"/>	▲
004	Admin profile		<input checked="" type="checkbox"/>	
999	Admin profile		<input checked="" type="checkbox"/>	
USPROF	USPROF		<input checked="" type="checkbox"/>	▼

Select all Unselect all

Create a screen process



Test in G.O.L.D.



A screenshot of a software menu window titled "Menu". The window displays the G.O.L.D. logo at the top. Below the logo is a list of menu items, each preceded by a small blue square icon. The items are: Technical data, Receptions, AGRN management, Consignment management, Schedule and check, SN management, Reception execution, Alcohol management in reception, Reception-related reports, SDT - RECEPTION STATEMENT, Stock management, Inventories, Replenishment, Route management, DO management, Scheduling, Standard preparation, Post-preparation processing, Loading and shipping, and I.R. preparation. The item "SDT - RECEPTION STATEMENT" is circled in red. A red arrow points from this circled item to the "G.O.L.D." tab in the main application window.



A screenshot of the G.O.L.D. application window. The window features the G.O.L.D. logo at the top. Below the logo is a navigation bar with three tabs: "Administration", "G.O.L.D.", and "Download". The "G.O.L.D." tab is highlighted with a red box. A red arrow points from the "SDT - RECEPTION STATEMENT" item in the menu window to this tab.

Test in G.O.L.D.

54 DEPOT NANTES **Advanced Goods Receiving Note management** 12345 DOSTK

AGRN no. Supplier order Reception date Reception status YES

AGRN no.	Order	Supplier	Recep. date	AGRN type	Type desc.	Received	On Consignment
13801		01F001	06/26/08	1	SUPPLIER	YES	<input checked="" type="checkbox"/>
13821		01F001	06/27/08	1	SUPPLIER	YES	<input checked="" type="checkbox"/>
13822		01F001	07/07/08	1	SUPPLIER	YES	<input checked="" type="checkbox"/>
13841		01F001	08/06/08	1	SUPPLIER	YES	<input checked="" type="checkbox"/>

Supplier name Address

Print report



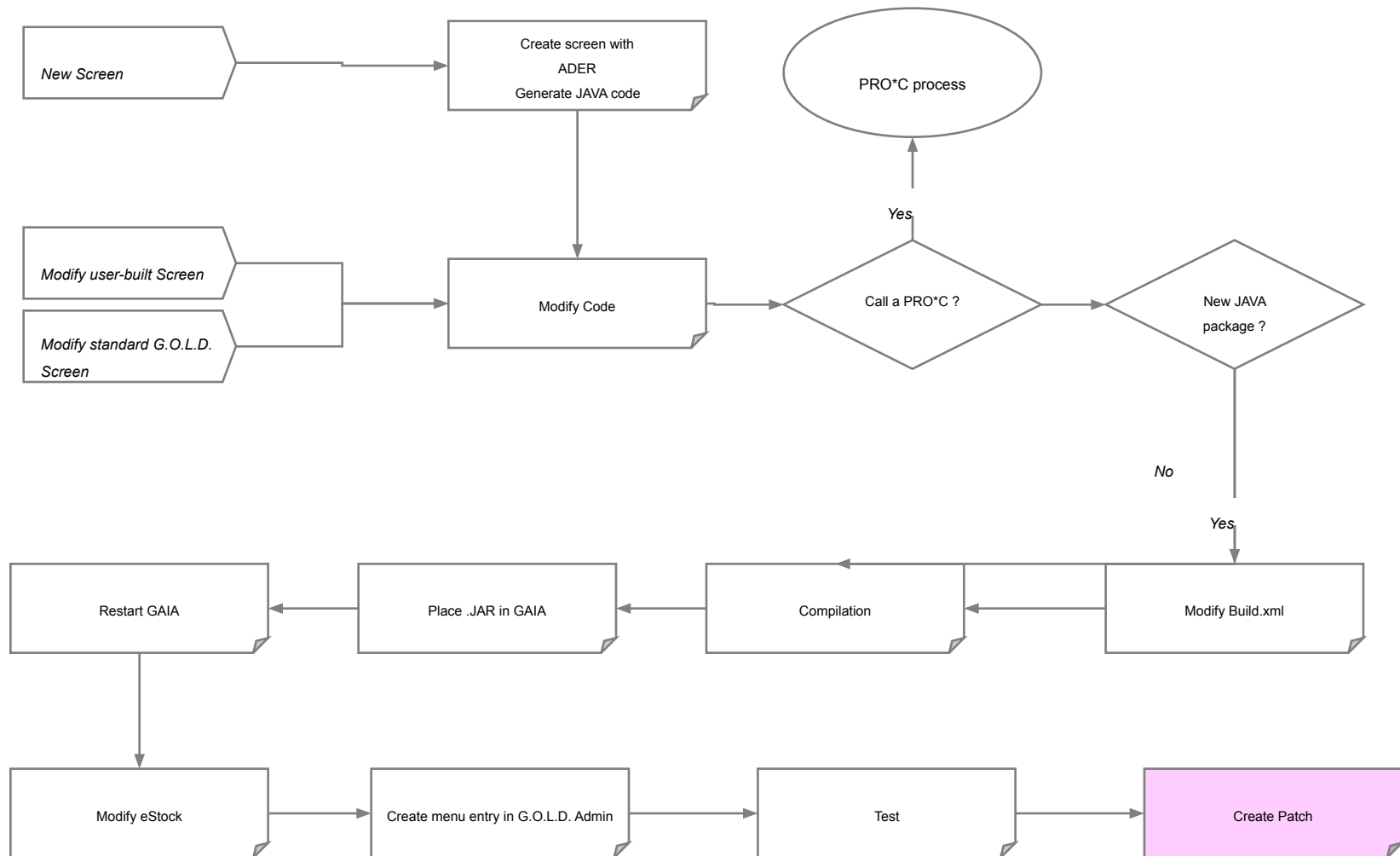
EXERCISE: Create a screen

- Open the following file for more details on this exercise:
TS200_GOLDStockDevelopment_Exercises - 4

1. Open .java in Eclipse
2. Modify code
3. Modify build.xml, compile, deploy
4. Restart gaia
5. Modify estock.html
6. Create menu entry
7. Test



Create a screen process



Patch Content



The Patch is subdivided in following directories:

- **/sql**
Contains SQL scripts for database modification, and also SQL scripts for creation of new parameters, messages, labels, menus entries ...
It contains also shell script for SQL scripts execution.
- **/pkg** - packages & triggers, functions ...
- **/gaia** - subdirectories of binary jar files and gaia application
- **/bin** - Pro*C binaries and object files
- **/shell** - shell sources
- **/misc** - all other documents necessary for installation of pack
- **/doc** - contains all necessary documents for installation



EXERCISE: Patch Content

- According to what we have just seen, if you create a new screen for your client, what will you put in the patch ?
 - Hint : 3 files are mandatory



- ***Answer on next screen***

Create Patch



- Files you MUST add to patch after a screen was created :
 - xxx.server.yyy.jar
 - xxx.client.yyy.jar
 - estock.html

- Files you may add to patch after a screen was created :
 - Script with “Descriptions/Labels” if new ones are used in report
 - Script to insert menu entry in the eStock application

Table of content



1 General Overview

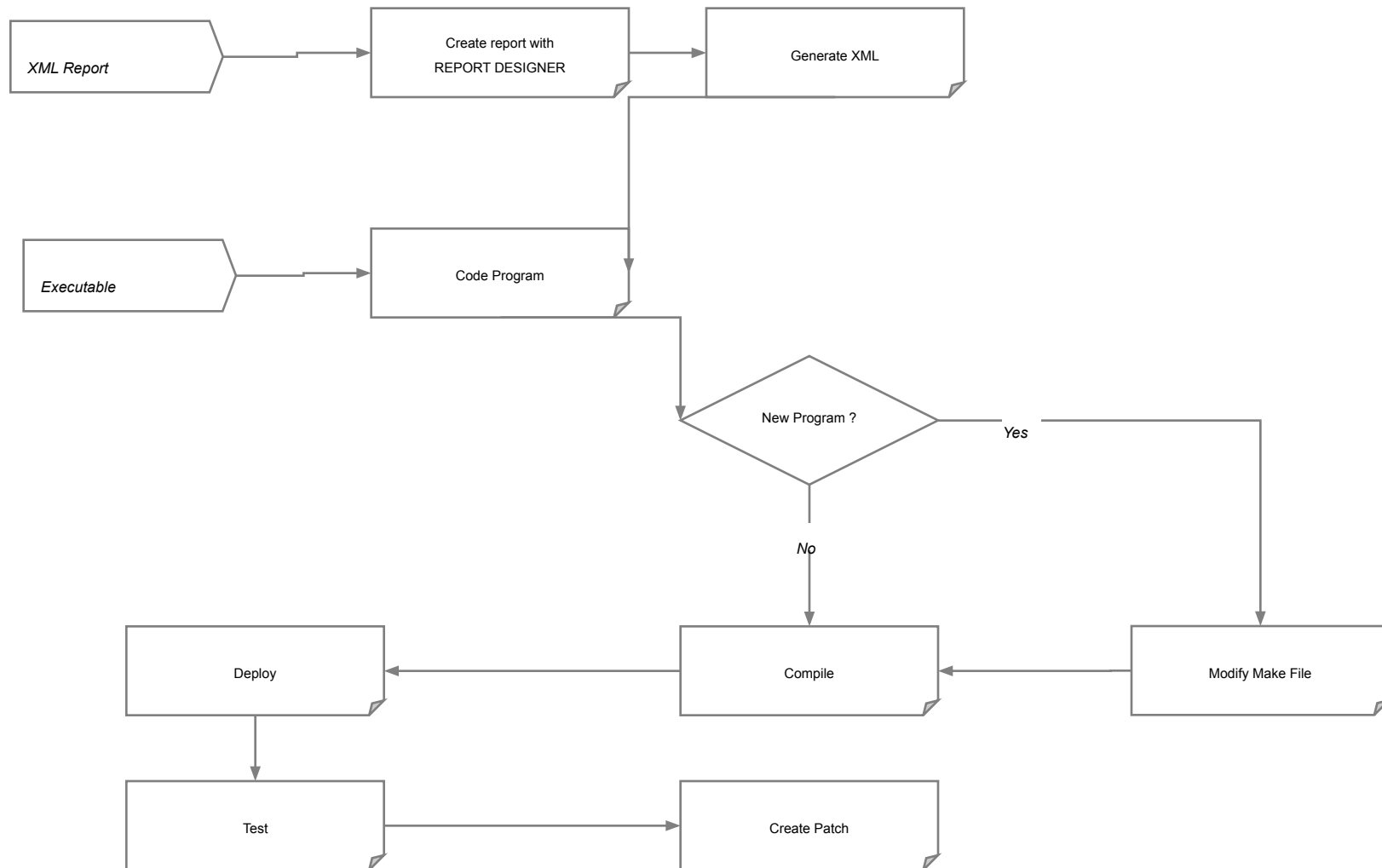
2 Global Architecture and Application Structure

3 G.O.L.D. Screens

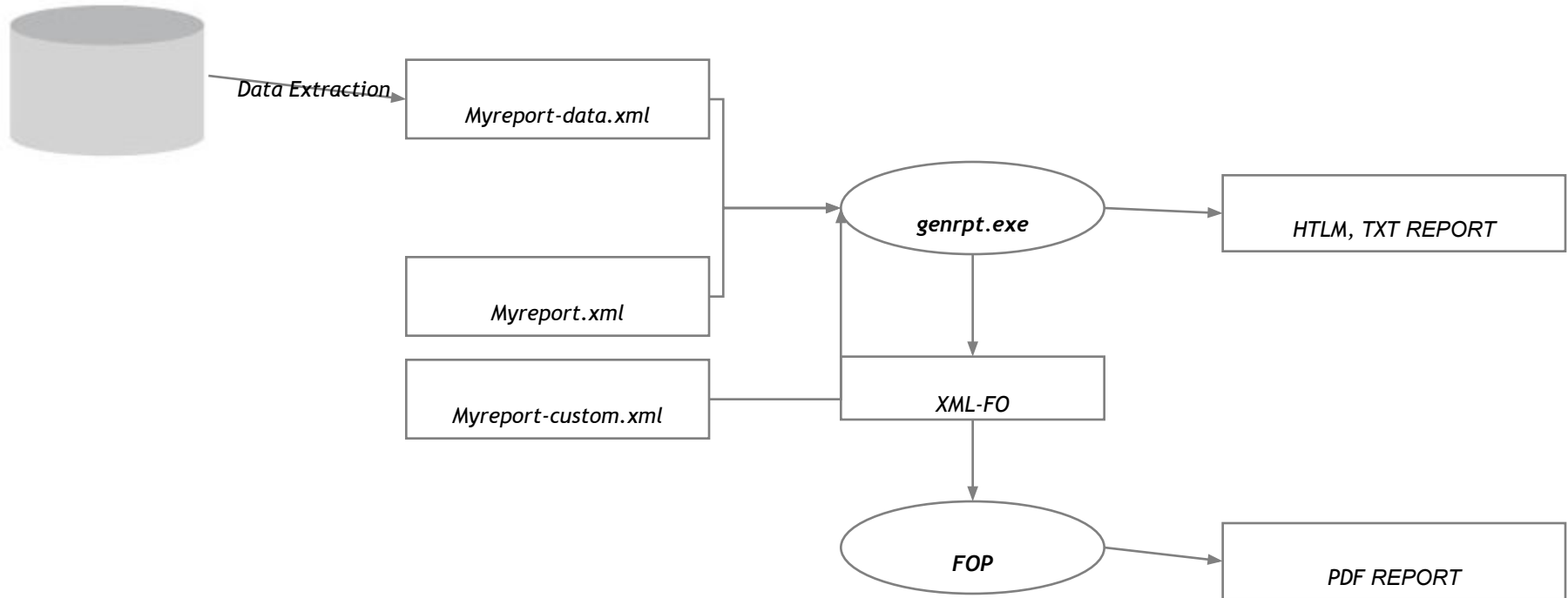
4 Create a new eStock screen

5 Create a new report

Create a program process



Report Generation



The creation of a report consists of two steps:

- Program that provides the data - (Myreport-data.xml)
- XML model for report data formatting - (Myreport.xml)

Report model



- XML language is used to describe the model.
- A model set is composed of several objects : each object is child a parent node.
- The characteristics of each object are set as attributes.
 - The name of the parent node is always: **Modeles**
 - The name of a node: **Modele**
- Example :

```
<Modeles>  
  <Modele Name="Title" ... />  
  <Modele Name="Logo" ... />  
  <Modele Name="Head-line" ... />  
  <Modele Name="CustomerTable" ... />  
  <Modele Name="Subtotal" ... />  
</Modeles>
```

Report model templates



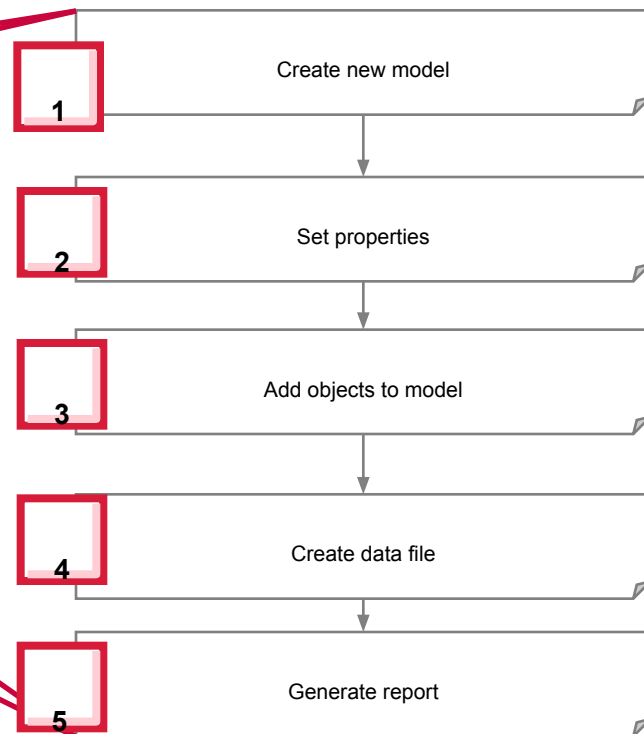
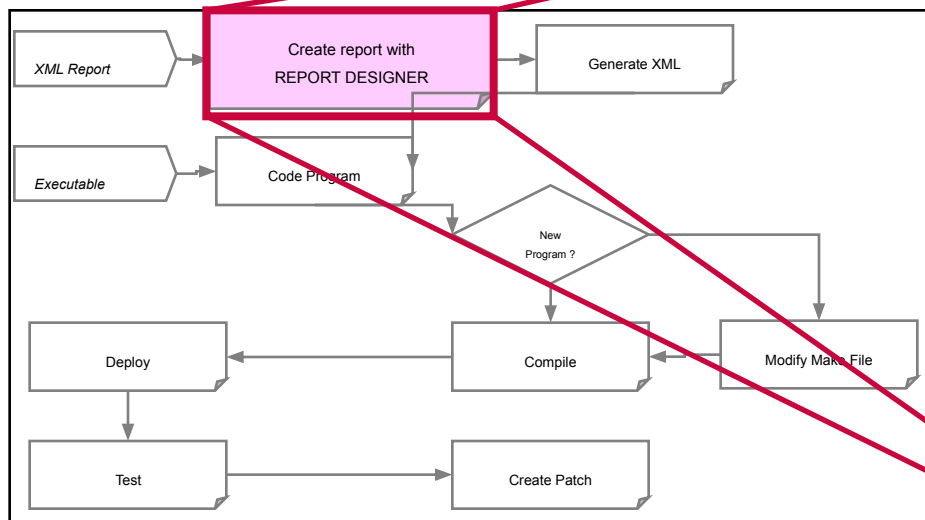
- Each model file contains the definition of several templates corresponding to a given report.
- There are two template types:
 - the simple template, which can be of several types (line, table ...),
 - the container, which will contain several simple templates.
- The position of each template is defined in the 3 different parts of the document.
 - "Header": header of the document,
 - "Footer": footer, bottom of each page.
 - "Body": rest of the document
- Basic properties for each template:
 - **Name** : (up to 50 characters without spaces)
 - **Type** : (Line, Table, arrayline, Stroke, Image)
 - **Location** : (header, body, footer)
 - **Left** : (left margin).
 - **Pageafter**
 - **Keep**

Report Designer



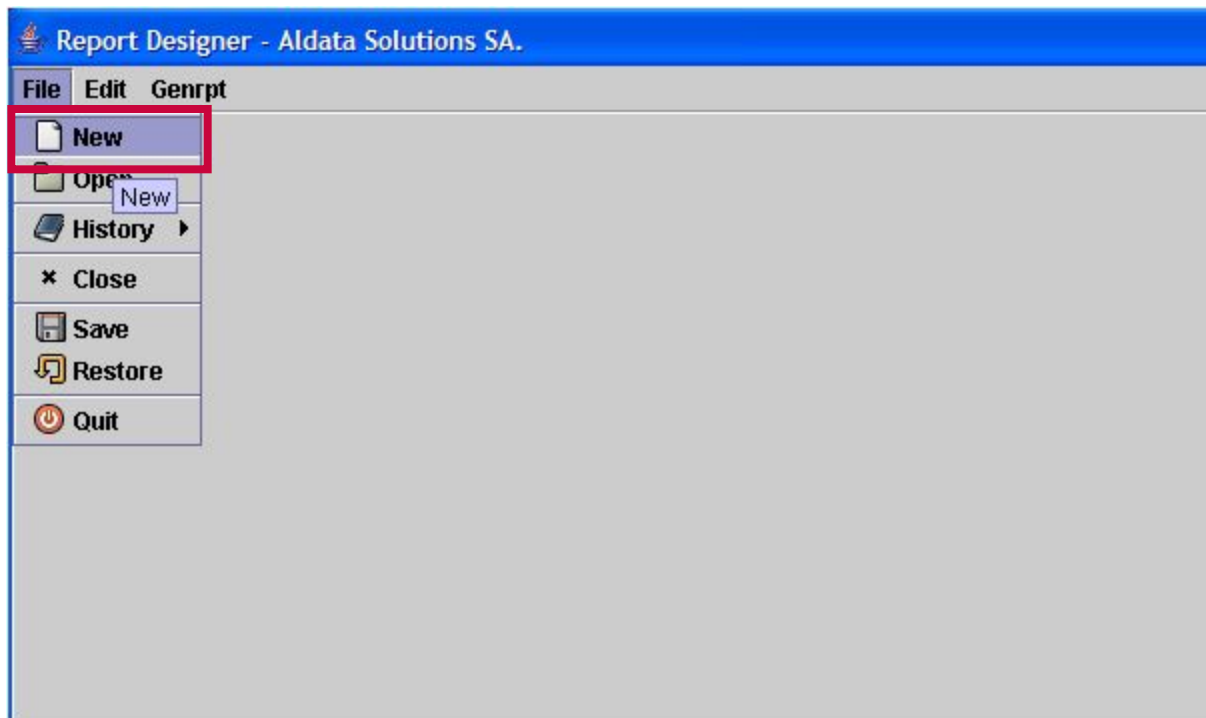
- *Report Designer* is a **reporting** tool 100% Java (Swing) use to publish reports into preset formats (PDF, HTML, TXT).
- It is used to create three files :
 - **Model :**
 - to print the publishing model set (`myreport.xml`)
 - **Custom:**
 - to alter objects of the model set (`myreport-custom.xml`)
 - **Data:**
 - to manage the publishing data (`myreport-data.xml`)
- **Required Reading :**
 - *Construction of the publishing model sets*
 - *Use and compilation of Genrpt*

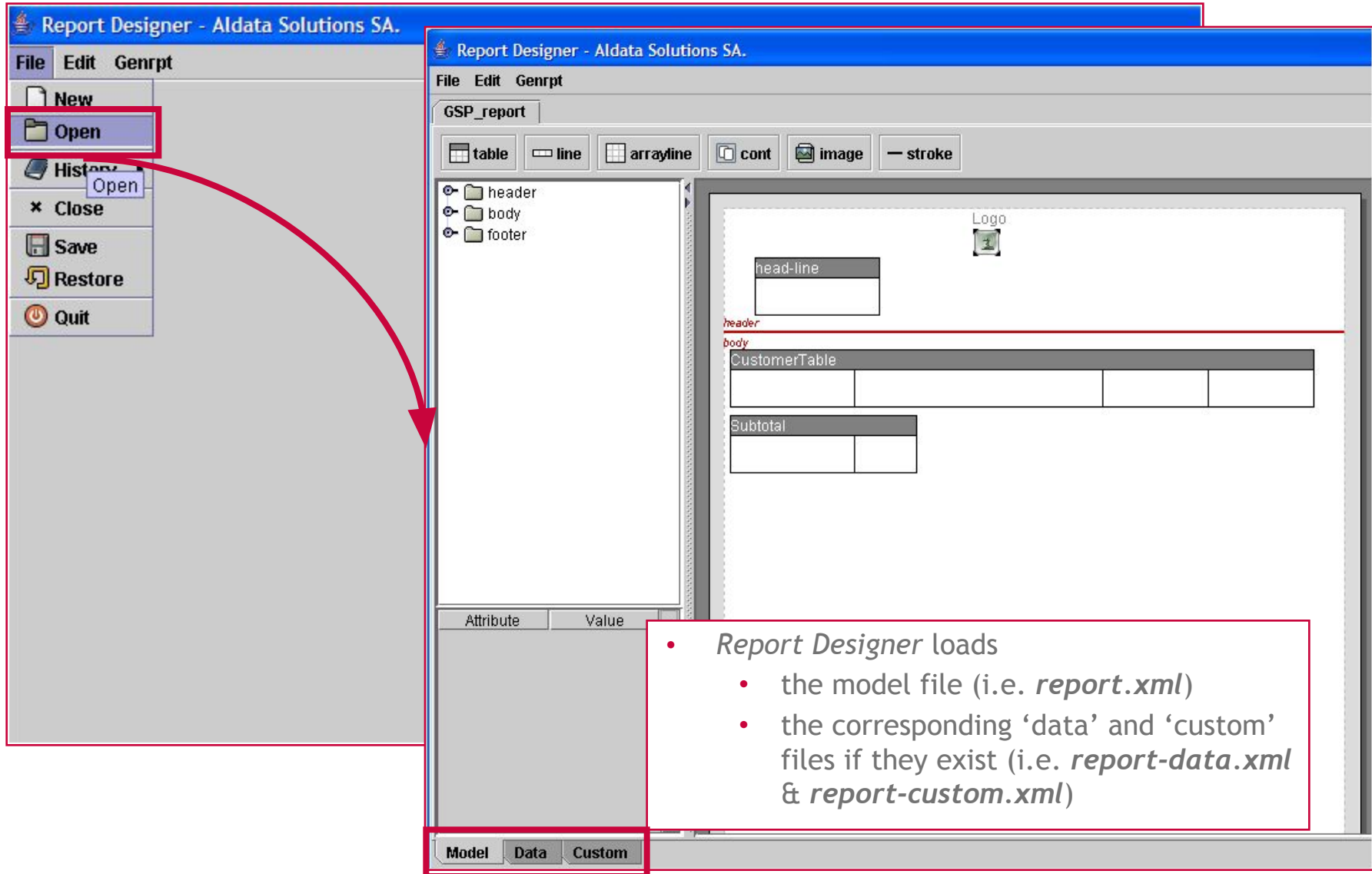
Report Designer



New Model Set Creation

1

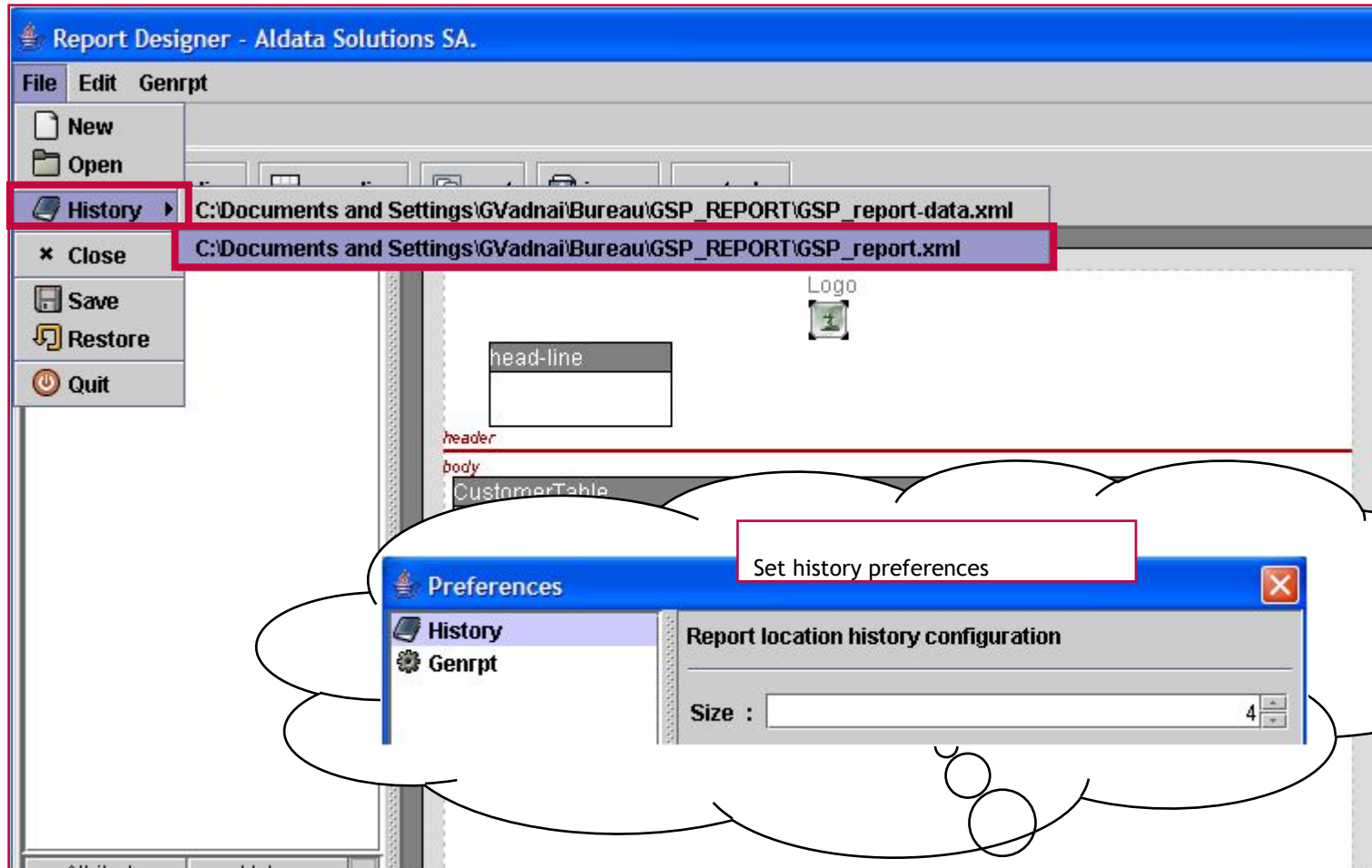


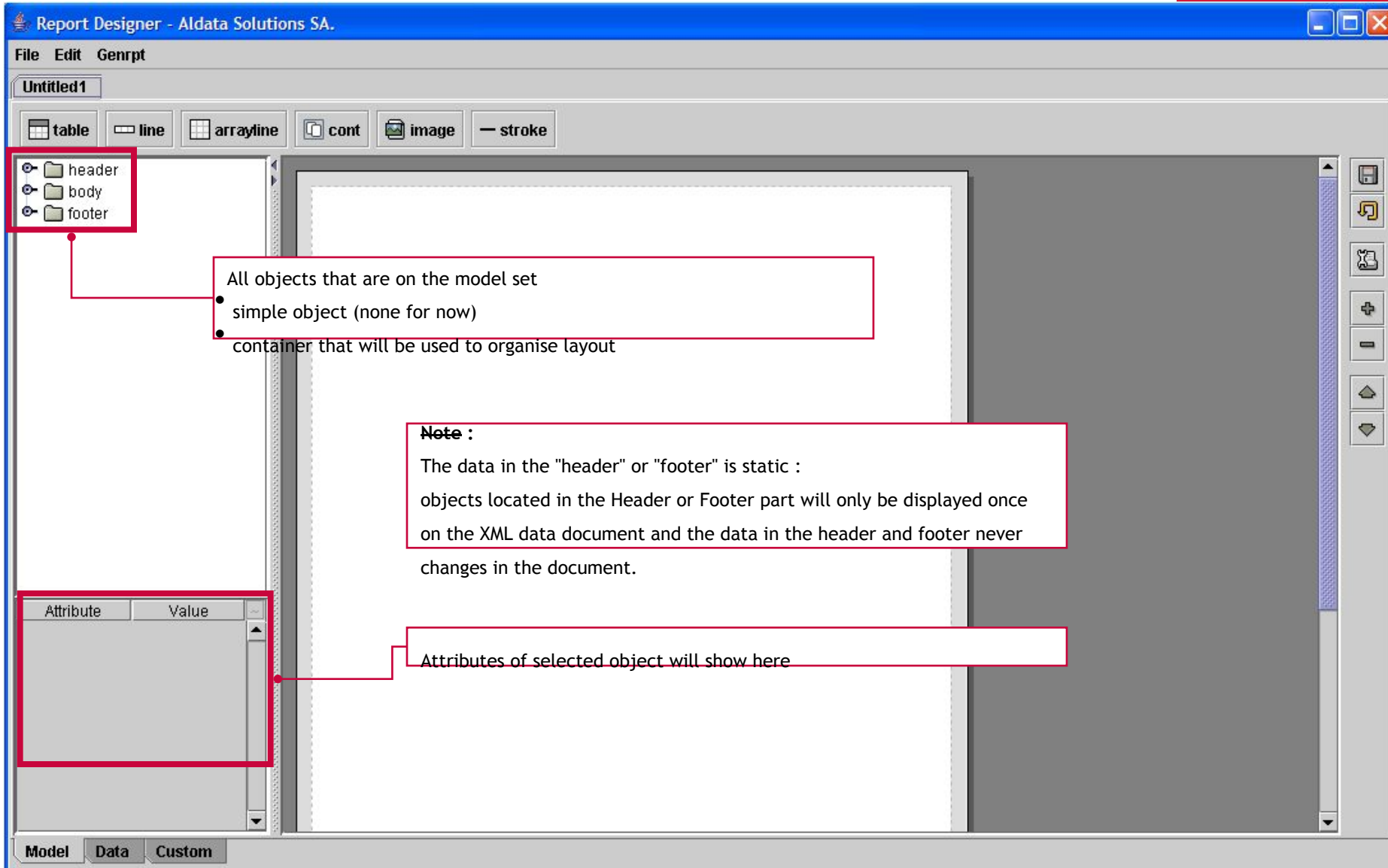


The screenshot displays the Report Designer interface. The 'File' menu is open, with the 'Open' option highlighted. A red arrow points from the 'Open' menu item to the 'GSP_report' window. The 'GSP_report' window shows a report design with a table and a subtotal section. The table has a header row and a body with columns. The subtotal section is below the table. The 'Model' tab is selected at the bottom of the window.

- Report Designer loads
 - the model file (i.e. *report.xml*)
 - the corresponding 'data' and 'custom' files if they exist (i.e. *report-data.xml* & *report-custom.xml*)

- *Report Designer* stores the loaded reports in the option **History**





The screenshot shows the 'Report Designer - Aldata Solutions SA.' application window. The title bar includes standard window controls. The menu bar contains 'File', 'Edit', and 'Genrpt'. Below the menu bar is a toolbar with icons for 'table', 'line', 'arrayline', 'cont', 'image', and 'stroke'. On the left side, there is a model tree with three folders: 'header', 'body', and 'footer'. The 'header' folder is selected. A callout box points to this tree with the text: 'All objects that are on the model set' followed by a bulleted list: '• simple object (none for now)' and '• container that will be used to organise layout'. Below the model tree is a table with two columns: 'Attribute' and 'Value'. A callout box points to this table with the text: 'Attributes of selected object will show here'. In the center of the main design area, there is a 'Note' box containing the text: 'Note : The data in the "header" or "footer" is static : objects located in the Header or Footer part will only be displayed once on the XML data document and the data in the header and footer never changes in the document.' At the bottom of the window, there are three tabs: 'Model', 'Data', and 'Custom', with 'Model' currently selected.

File Edit Genrpt

Untitled1

table line arrayline cont image stroke

- header
- body
- footer

All objects that are on the model set

- simple object (none for now)
- container that will be used to organise layout

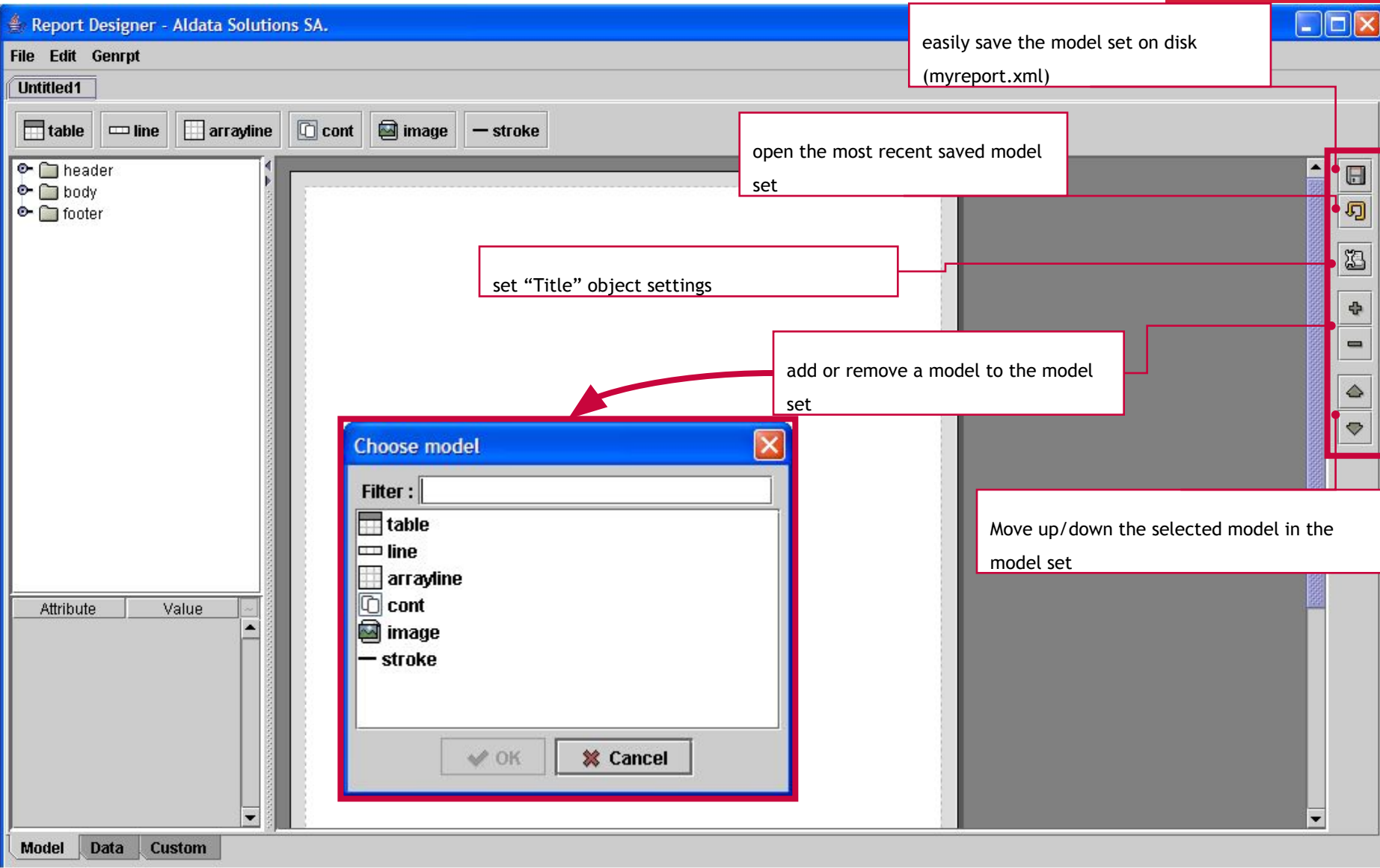
Note :

The data in the "header" or "footer" is static :
objects located in the Header or Footer part will only be displayed once on the XML data document and the data in the header and footer never changes in the document.

Attribute	Value
-----------	-------

Attributes of selected object will show here

Model Data Custom



The screenshot shows the 'Report Designer' application window. The title bar reads 'Report Designer - Aldata Solutions SA.'. The menu bar includes 'File', 'Edit', and 'Genrpt'. The main workspace is titled 'Untitled1'. A toolbar at the top contains icons for 'table', 'line', 'arrayline', 'cont', 'image', and 'stroke'. On the left, a tree view shows a folder structure with 'header', 'body', and 'footer'. At the bottom, there are tabs for 'Model', 'Data', and 'Custom'. A 'Choose model' dialog box is open in the center, listing the same toolbar items. A vertical toolbar on the right side of the workspace contains icons for saving, opening, and moving items. Red callout boxes with arrows point to these elements, providing instructions.

File Edit Genrpt

Untitled1

table line arrayline cont image stroke

header
body
footer

Attribute Value

Model Data Custom

Choose model

Filter :

table
line
arrayline
cont
image
stroke

OK Cancel

easily save the model set on disk
(myreport.xml)

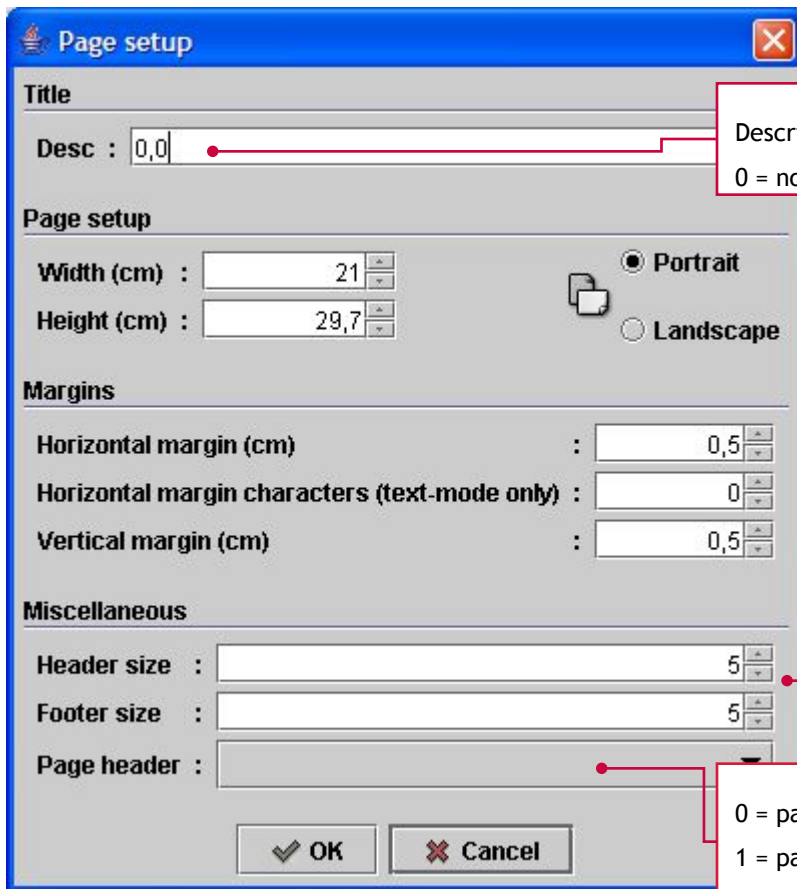
open the most recent saved model
set

set "Title" object settings

add or remove a model to the model
set

Move up/down the selected model in the
model set

- This object must absolutely be present in the model and set up first.
- It specifies the report environment.



Description number.
0 = no title bar displayed

In % of the page

0 = page header and footer not displayed
1 = page header and footer are displayed.
By default : standard page header and footer are displayed



Set Genrpt properties



complete path to the **genrpt.exe** program

PDF, HTML or TXT

User/Password oracle for the connexion to the dictionary

Number of lines per page for the HTML editions (0 for PDF)

Type of the template. (Automatic)

Set to 1 if use of extended ASCII characters

language code for searching in the dictionary

Number of characters per line (for txt output)

Set environment variables

Preferences

Genrpt parameters and environment

Genrpt.exe path : genrpt.exe

Oracle connection : stkdew/stkdev@devstock

Application name : ESTOCK

Output type : HTML

Use Custom model :

Number of lines : 10

Language : GB

Number of char : 132

Extended : 1

Decimal separator ('.' or ';')

Environment variables

GOLDNUMBERSEP : .

GOLDDATEMASK : DD/MM/RR

GOLDTIMEMASK : HH24

GOLDENCODING : ISO-8859-1

REPORT_CFG : .

GENRPT_DEBUG :

OK Cancel

encoding type for the XML generated file (ex : ISO-8859-1)

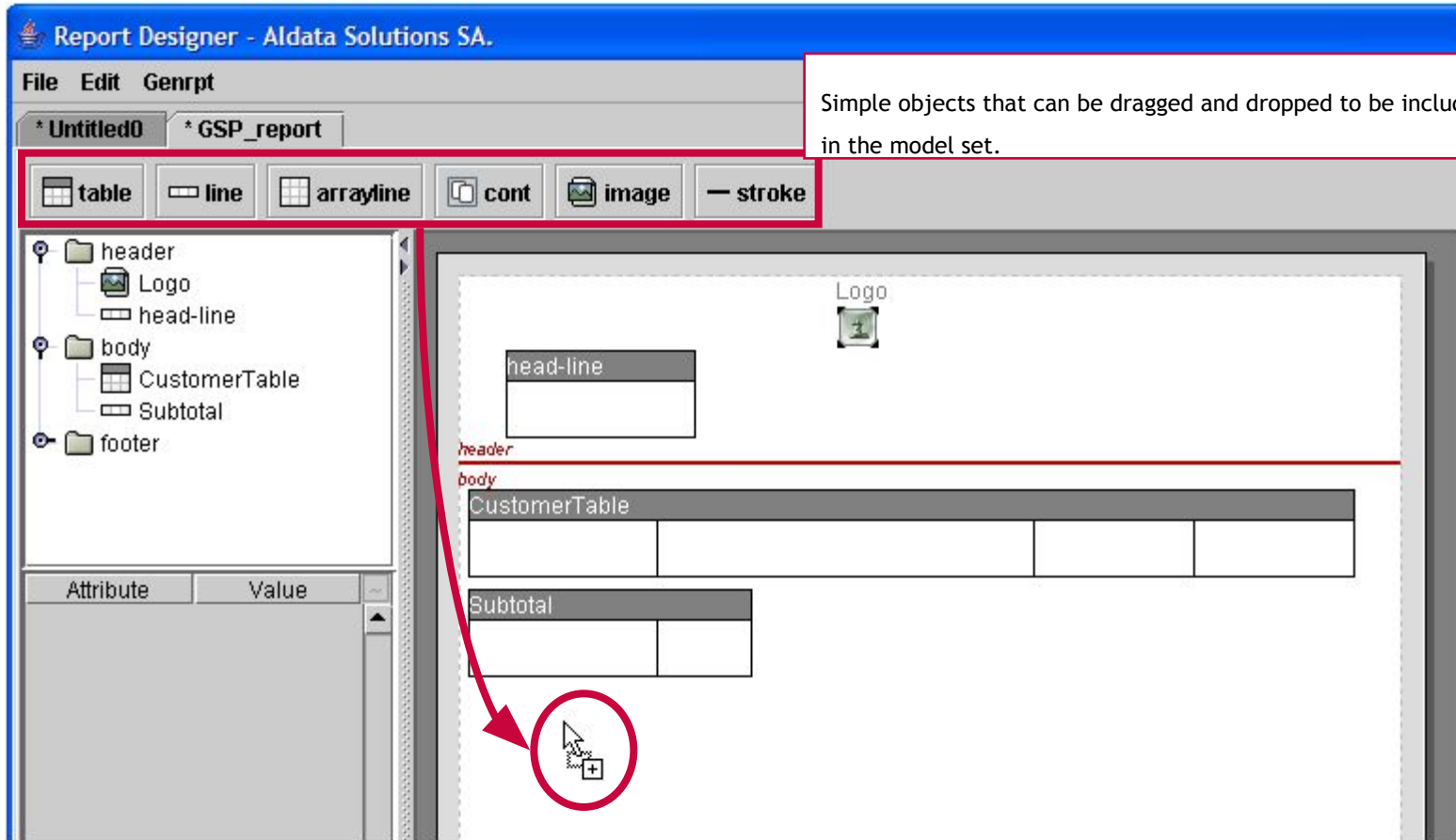
Decimal separator ('.' or ';')

Date format : use same codes as Oracle

Time format : HH24 or HH12

Report_cfg folder if used

Add objects to model



Report Designer - Aldata Solutions SA.

File Edit Genrpt

*Untitled0 *GSP_report

table line arrayline cont image stroke

header
 Logo
 head-line
body
 CustomerTable
 Subtotal
footer

Attribute Value

Simple objects that can be dragged and dropped to be included in the model set.

header

body

CustomerTable

Subtotal

Properties common to all objects.



3



Add new model "table"

Name :

Type : table

Location :

modelRef :

Left : 0

LeftChar : 0

Top : 0

PageAfter :

Keep :

Desc :

Size :

SizeChar :

Align :

Aspect :

Font : 1

LastBorder :

Updesc :

Upstart :

Upsize :

OK Cancel

Name of the template (50 characters max - no spaces)

Type of the template. (Automatic)

Section in which the object will show (header, body, footer)

Position : left margin, ... , distance between object and previous one.

Insert a page break after the object.

Mandatory fields are shown in blue

Indicate if Generator should keep this block and the next one on the same page.

Table Object



Add new model "table"

Name :

Type : table

Location :

modelRef :

Left :

LeftChar :

Top :

PageAfter :

Customer code	Customer	Customer type code	Creation date
---------------	----------	--------------------	---------------

Keep :

Desc :

Size :

SizeChar :

Align :

Aspect :

Font : 1

LastBorder :

Updesc :

Upstart :

Upsize :

OK Cancel

Name of the container (if any)

Description numbers for the header.
i.e. 172,223,1696,460

Size of each column.

Alignment for each column. (l, c or r)

Note:

- by default, columns are left aligned.
- only specifies the alignment of values in columns (no impact on headers that are always centered)

Table Object



Add new model "table"

Name :

Type : table

Location :

modelRef :

Left :

LeftChar :

Top :

PageAfter :

Keep :

Desc :

Size :

SizeChar :

Align :

Aspect :

Font :

LastBorder :

Updesc :

Upstart :

Upsize :

Font size for characters. Possible values: 1, 2, 3, 4...11 (from the smallest to the largest font).

Only used for TXT generation to fuse last border of a table to the next one

Used for complex tables to group headers

DESIGNATION ARTICLE			FOURNISSEUR		
Article	VA	VL	No ligne	Code	Fillere
colonne 1	colonne 2	colonne 3	colonne 4	colonne 5	colonne 6



Add new model "table"

Name :

Type :

Location :

modelRef :

Left :

LeftChar :

Top :

PageAfter :

Keep :

Desc :

Size :

SizeChar :

Align :

Aspect :

- box
- gray
- grid
- Ngrid
- Ugrid

Font :

LastBorder :

Updesc :

Upstart :

Upsize :

- Graphic aspect of the table.
 - "gray" : highlight a line out of two
 - "grid" : border around the table
 - "Ugrid" : border around the table except from top first line
 - "Ngrid" : border around the table except bottom last line
 - "box" : specific border of box type

Table Aspect : examples

- Graphic aspect of the table.
 - “gray”

Customer code
xxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxx

- “gray” + “grid”

Customer code
xxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxx

- Nothing

Customer type code
xxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxx

- “box”
 - “box” attribute is only valid for a table with one column.
 - “box” attribute is not compatible with “gray” and “grid” attribute

Total number of lines
xxxxxxxxxxxxxxxxxx

Line Object

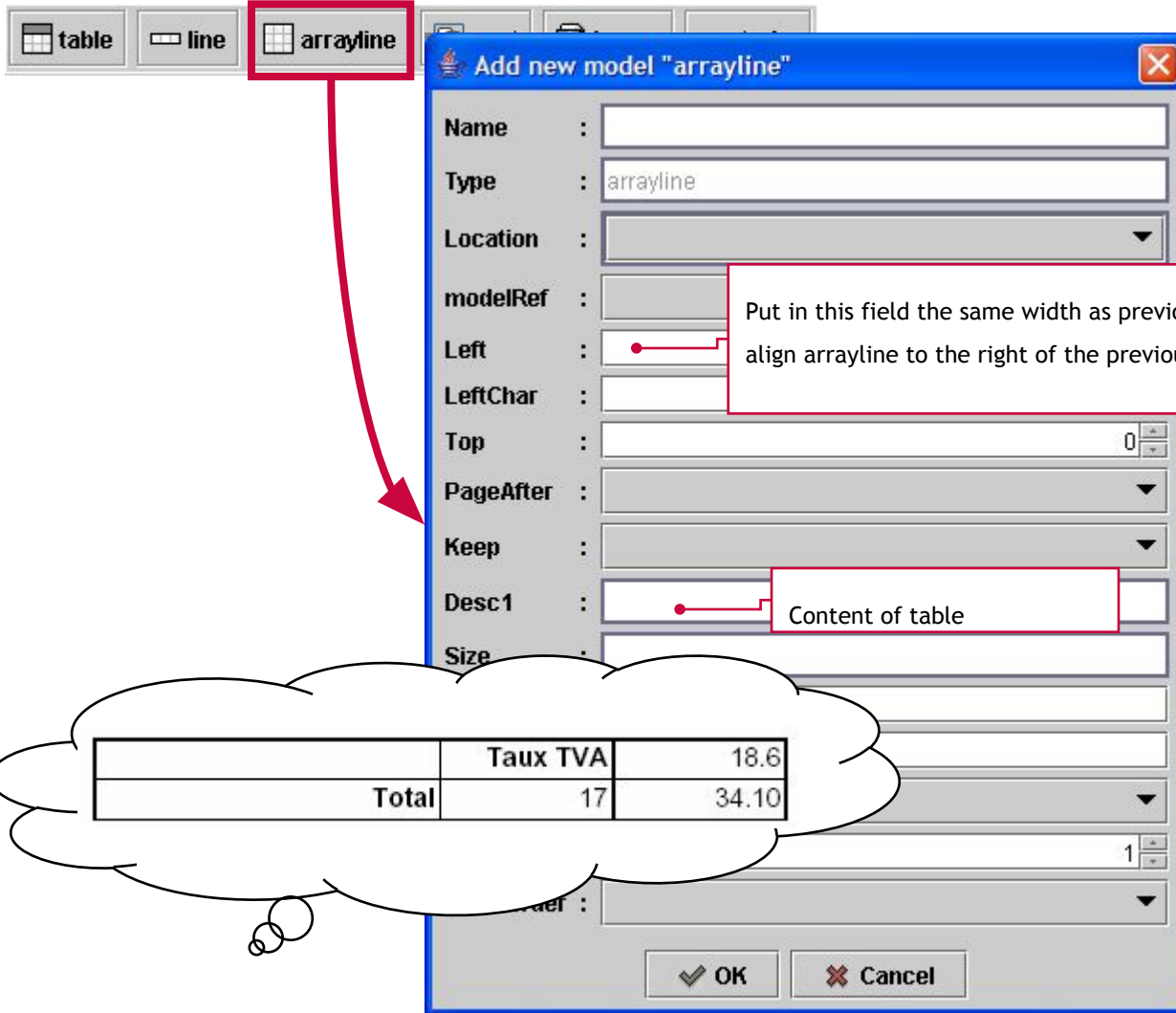
- Object used to insert a line of text into a report and does not retrieve any information from the XML data file.

Content of the line (description or variant):

- a description number
- or the ' v ' character to indicate the presence of a variable in the XML data file.

Size of each field in the line

Nothing or "gray"



Put in this field the same width as previous table to align arrayline to the right of the previous table

Content of table

	Taux TVA	18.6
Total	17	34.10

- Arrayline objects are tables with non-homogeneous data.
 - Some columns of the table may not contain any data.



- Objects cannot be aligned horizontally when using simple objects : you must use a container.

Add new model "cont"

Name :

Type :

Location :

modelRef :

Left :

LeftChar :

Top :

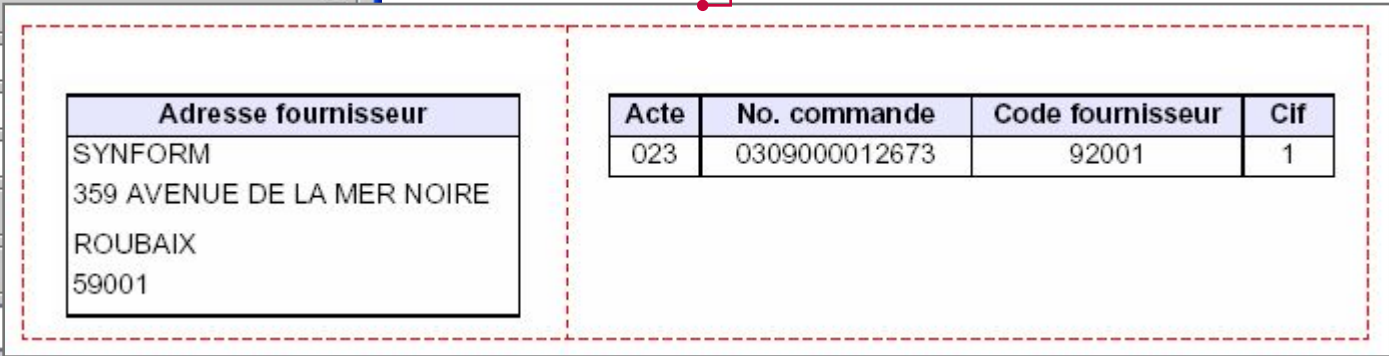
PageAfter :

Keep :

Size :

SizeChar :

Example :
Container with two columns



```
<Modele type="cont" Name="commande" size="40,60"/>
<Modele Type="table" Name="Adresse" modelRef="commande" .... />
<Modele Type="table" Name="InfoCommande" modelRef="commande" .... />
```



Add new model "image"

Name :

Type :

Location :

modelRef :

Left :

LeftChar :

Top :

PageAfter :

Keep :

Url :

URL address where image can be found.

- Object used to insert an image into a report and does not retrieve any information from the XML data file.
- Compatible image format:
 - Gif
 - Jpeg





Add new model "stroke"

Name :

Type : stroke

Location :

modelRef :

Left :

LeftChar :

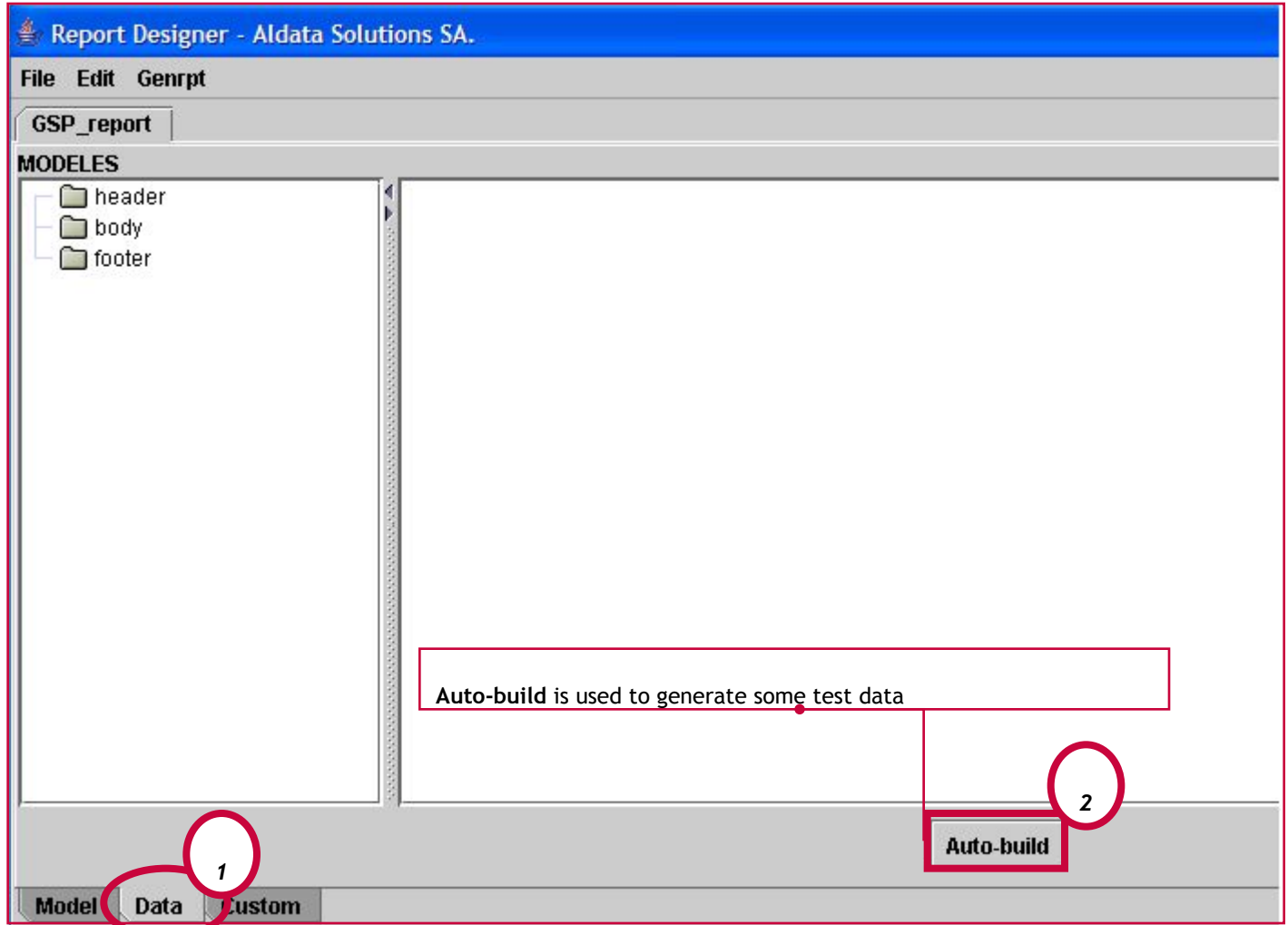
Top :

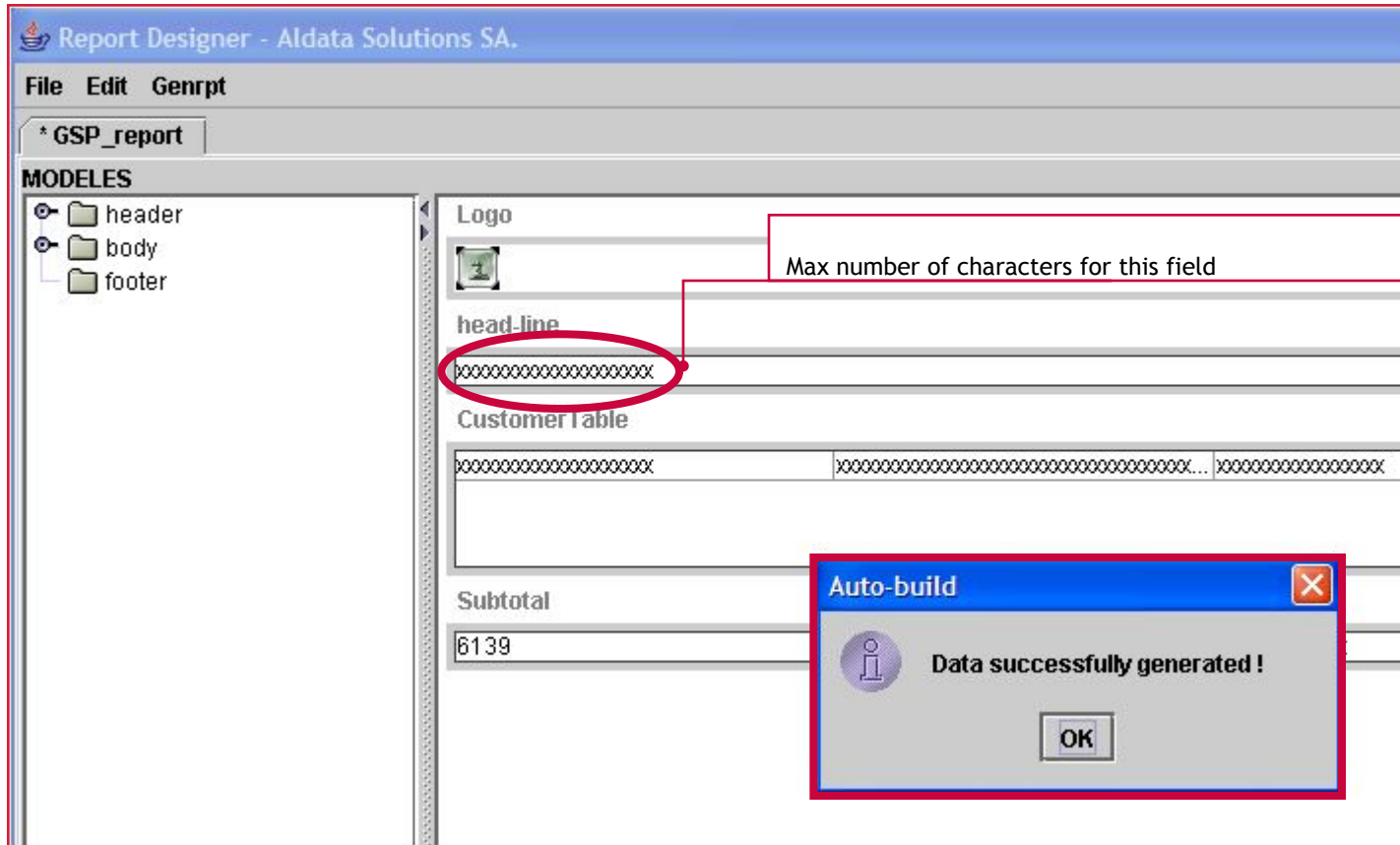
PageAfter :

Keep :

- Object used to display an horizontal line in the document and does not retrieve any information from the XML data file.

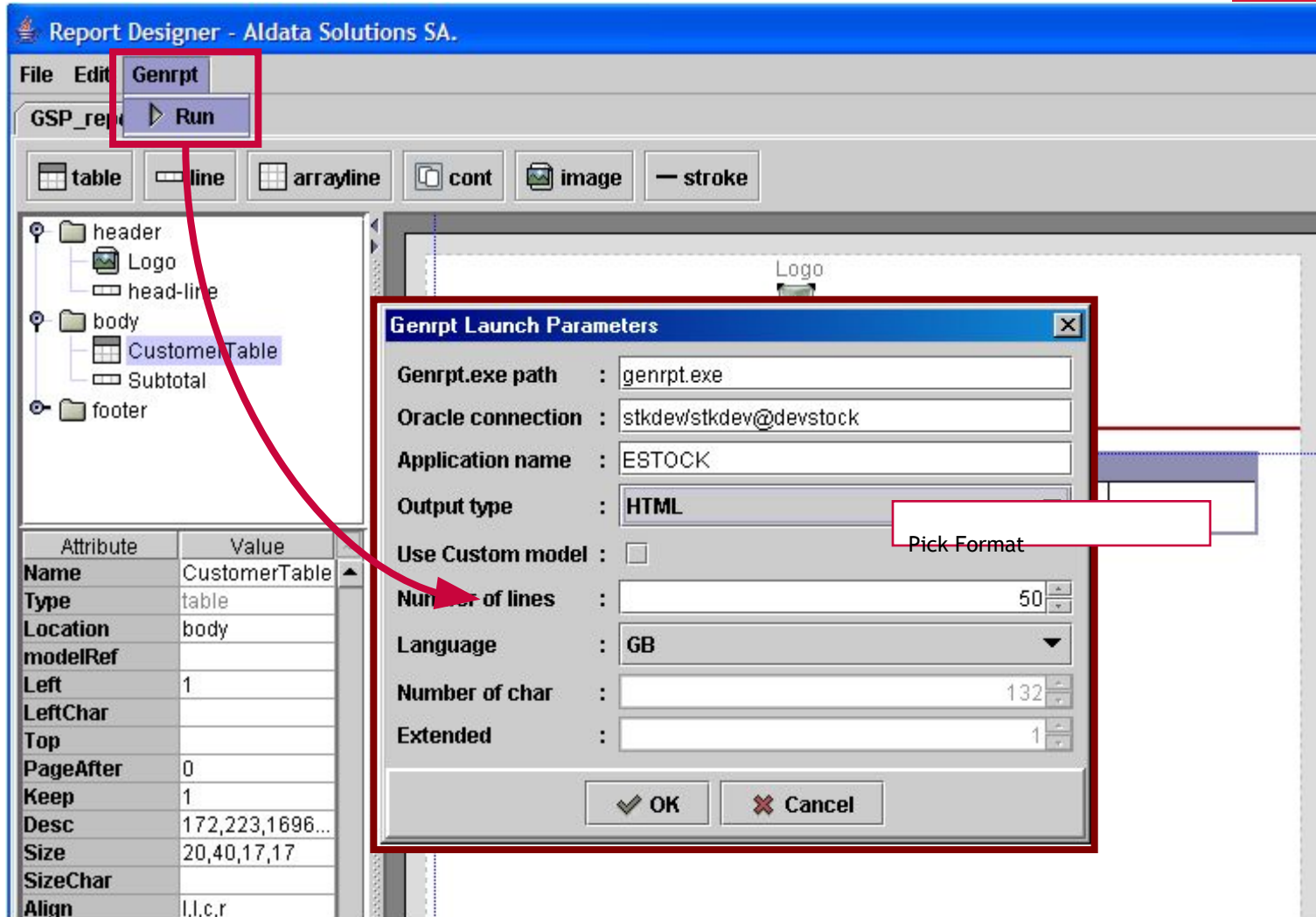
- Creating the data.xml file now will be useful for tests or previews.





The screenshot shows the 'Report Designer - Aldata Solutions SA.' interface. On the left, a 'MODELES' tree contains 'header', 'body', and 'footer'. The main workspace displays a report layout with sections: 'Logo', 'head-line', 'Customer Table', and 'Subtotal'. The 'head-line' section contains a field with a red circle around it, and a callout box points to it with the text 'Max number of characters for this field'. The 'Customer Table' section shows a table with placeholder text. The 'Subtotal' section shows the value '6139'. An 'Auto-build' dialog box is open in the foreground, displaying an information icon and the message 'Data successfully generated!' with an 'OK' button.

Generate Report



The screenshot shows the 'Report Designer - Aldata Solutions SA.' application. The 'Genrpt' menu is highlighted, and the 'Run' option is selected. A 'Genrpt Launch Parameters' dialog box is open, displaying the following configuration:

- Genrpt.exe path : genrpt.exe
- Oracle connection : stkdev/stkdev@devstock
- Application name : ESTOCK
- Output type : HTML
- Use Custom model :
- Number of lines : 50
- Language : GB
- Number of char : 132
- Extended : 1

Buttons for 'OK' and 'Cancel' are visible at the bottom of the dialog. A callout box labeled 'Pick Format' points to the 'Output type' field.

The background shows a report design tree with 'CustomerTable' selected in the 'body' section. Below the tree is a table with the following data:

Attribute	Value
Name	CustomerTable
Type	table
Location	body
modelRef	
Left	1
LeftChar	
Top	
PageAfter	0
Keep	1
Desc	172,223,1696...
Size	20,40,17,17
SizeChar	
Align	l,l,c,r


Preview Report Layout

Modeles

G.O.L.D. Report

Page 1
07 15:37:31
GSP_report

Print report



XXXXXXXXXXXXXXXXXX

Customer code	Customer	Customer type code	Creation date
XXXXXXXXXXXXXXXXXX	XX	XXXXXXXXXXXX	XXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX	XX	X	
XXXXXXXXXXXXXXXXXX	XX	X	
Total number of lines			
Total number of lines	XXXXXX		

Modele Name="title"

Modele Name="Logo"

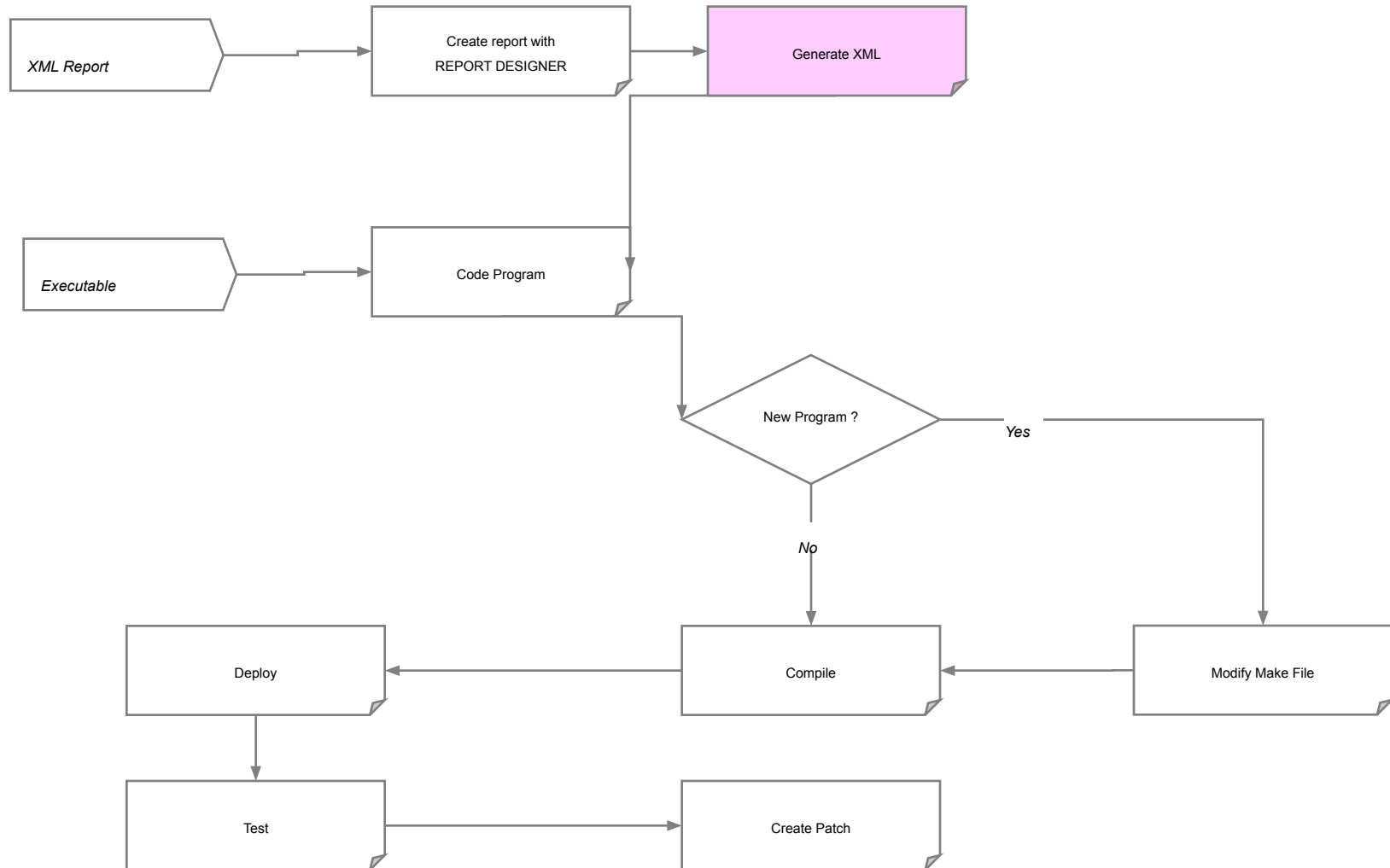
Modele Name="head-line"

Modele Name="CustomerTable"

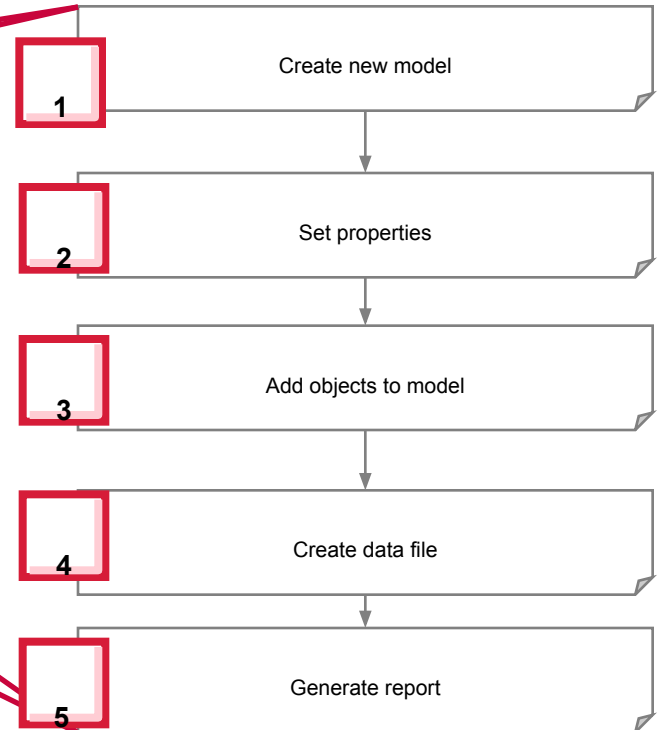
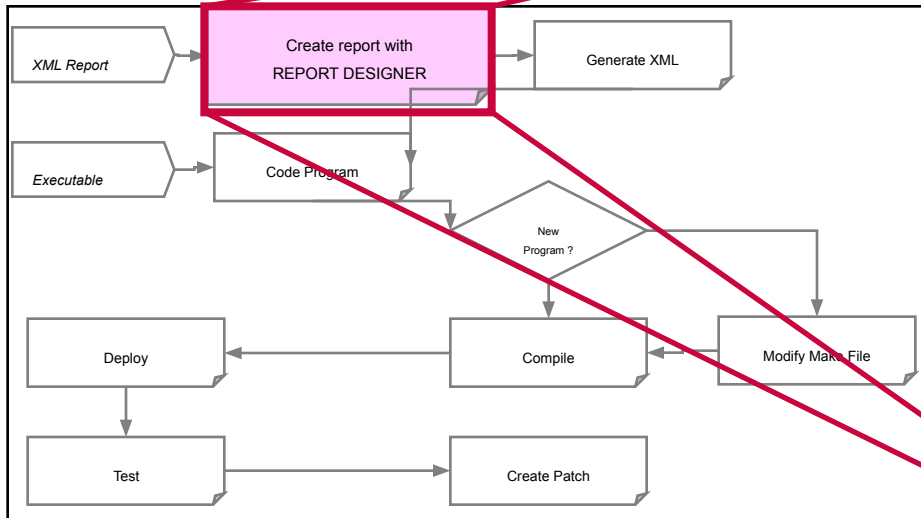
Modele Name="Subtotal"

```
<Modeles>
<Modele Name="title" Type="" Location="" Desc="5976,0" Height="29.700000762939453" Width="21.0" Hmargin="0.5" HmarginChar="0" Vmargin="0.5" Sizeheader="5" Sizefooter="5" />
<Modele Name="Logo" Type="image" Location="header" Left="40" Top="0" PageAfter="0" Keep="0" Url="http://odeon/GSP/LOGOGSP.png" />
<Modele Name="head-line" Type="line" Location="header" Left="5" Top="0" PageAfter="0" Keep="0" Desc="v" Size="20" Font="5" />
<Modele Name="CustomerTable" Type="table" Location="body" Left="1" PageAfter="0" Keep="1" Desc="172,223,1696,460" Size="20,40,17,17" Align="l,l,c,r" Aspect="gray" />
<Modele Name="Subtotal" Type="line" Location="body" Left="1" Top="1" PageAfter="0" Keep="0" Desc="6139,v" Size="20,10" />
</Modeles>
```

Create a program process



Report Designer





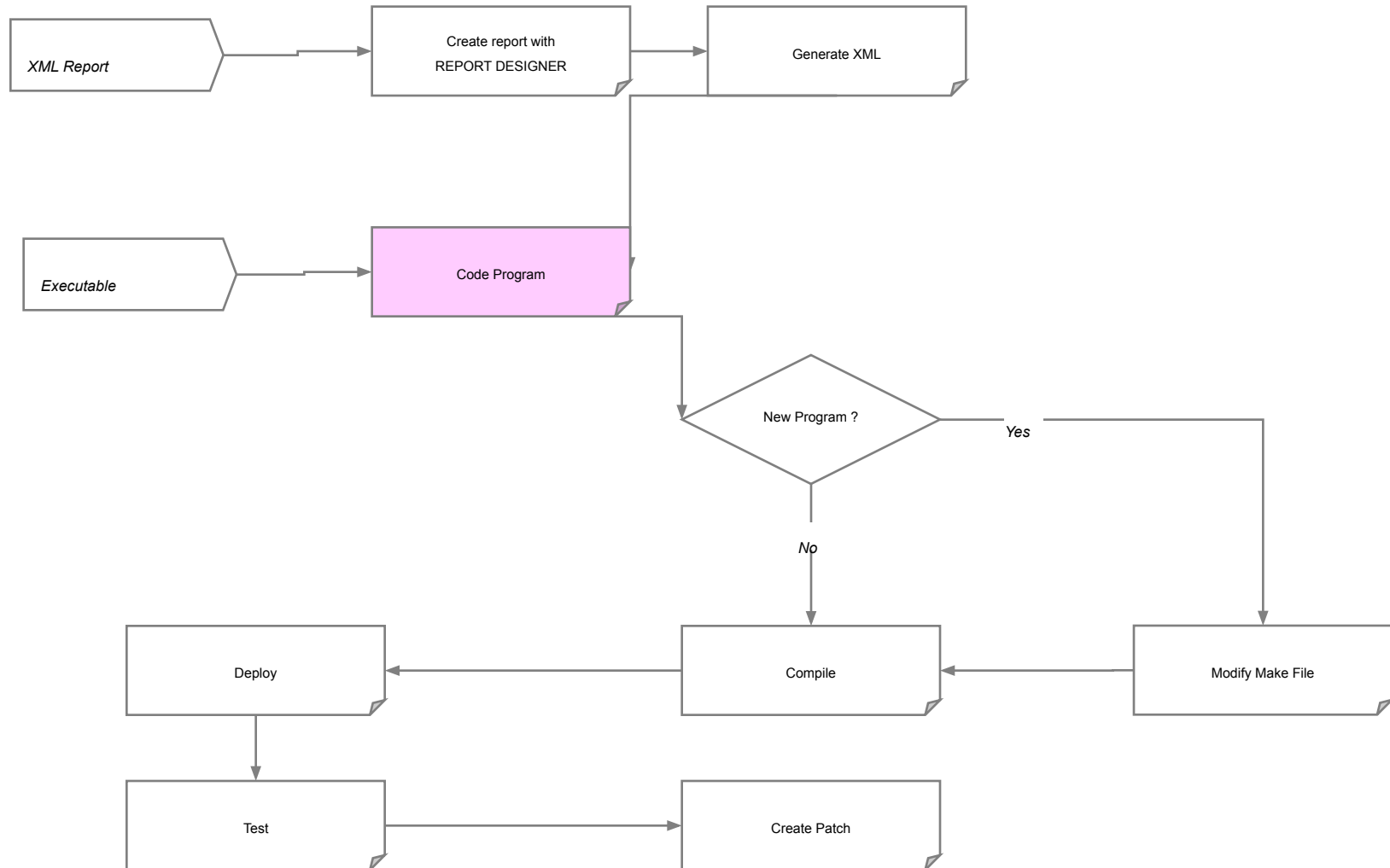
EXERCISE: Report Designer



- Open the following file for more details on this exercise:
 - TS200_GOLDStockDevelopment_Exercises
- 5. Report Designer
 - 1.1. Set preferences
 - 1.2. Open existing report
 - 1.3. Create data file
 - 1.4. Generate report
- 6. *Review sample reports*
 - Launch report from command line
 - Generate report from XML model and data file: genrpt, FOP



Create a program process



Code report program



- In the program is implemented logic for report, report XML data generation
- Pro*C program use standard functions of the different GOLD libraries
- Functions for creating report data
 - **InitReport(flleName)**
InitReportWithAttributes(flleName,Attrib,nbAttrib)
 - **CloseReport**
 - **InitModele (templateName)**
InitModeleWithAttributes(templateName,Attrib,nbAttrib)
Attrib[]={ "Label", "3,toto,1,tutu", " UpLabel", "1,up1,2,up2"};
 - **WriteRow(n, arg1, arg2, ..., argn)**
The first argument always gives the number of the next argument in the function
arg1 is the name of an element, arg2 is the value of this element,
- It is very important that the call order in the data XML file starts with the HEADER data then the FOOTER data, and finally the data to be displayed in the BODY.

Code report program



Utility functions for WriteRow:

- `char * DoubleToChar (arg1, arron)`
This function changes the `arg1` argument with character-string format, rounded to the decimal number indicated in the rounding argument.

```
WriteRow(2, « SiteNumber », DoubleToChar ( (double) site , 0)) ;
```

`GOLDNUMBERSEP`: indicates the decimal separator ('.' or ',').

`GOLDGROUPNUMBERSEP`: indicates the separator for thousands.

Special elements to the data:

- Images: `REFIMG(url_image)` ;
`WriteRow(2, « Img », « REFIMG(http://127.0.0.1/img/hello01.jpg) »);`
- bar codes: `REFBARCODE (bar_code, type, height, numeric_display)` ;
`WriteRow(2, «CodeBarre », « REFBARCODE (9780444505156, EAN13, 23.1, TRUE) ») ;`



EXERCISE: reports



- Open the following file for more details on this exercise:
 - TS200_GOLDStockDevelopment_Exercises - 5, 6

1. Report designer

2. *Review sample reports*





EXERCISE: reports

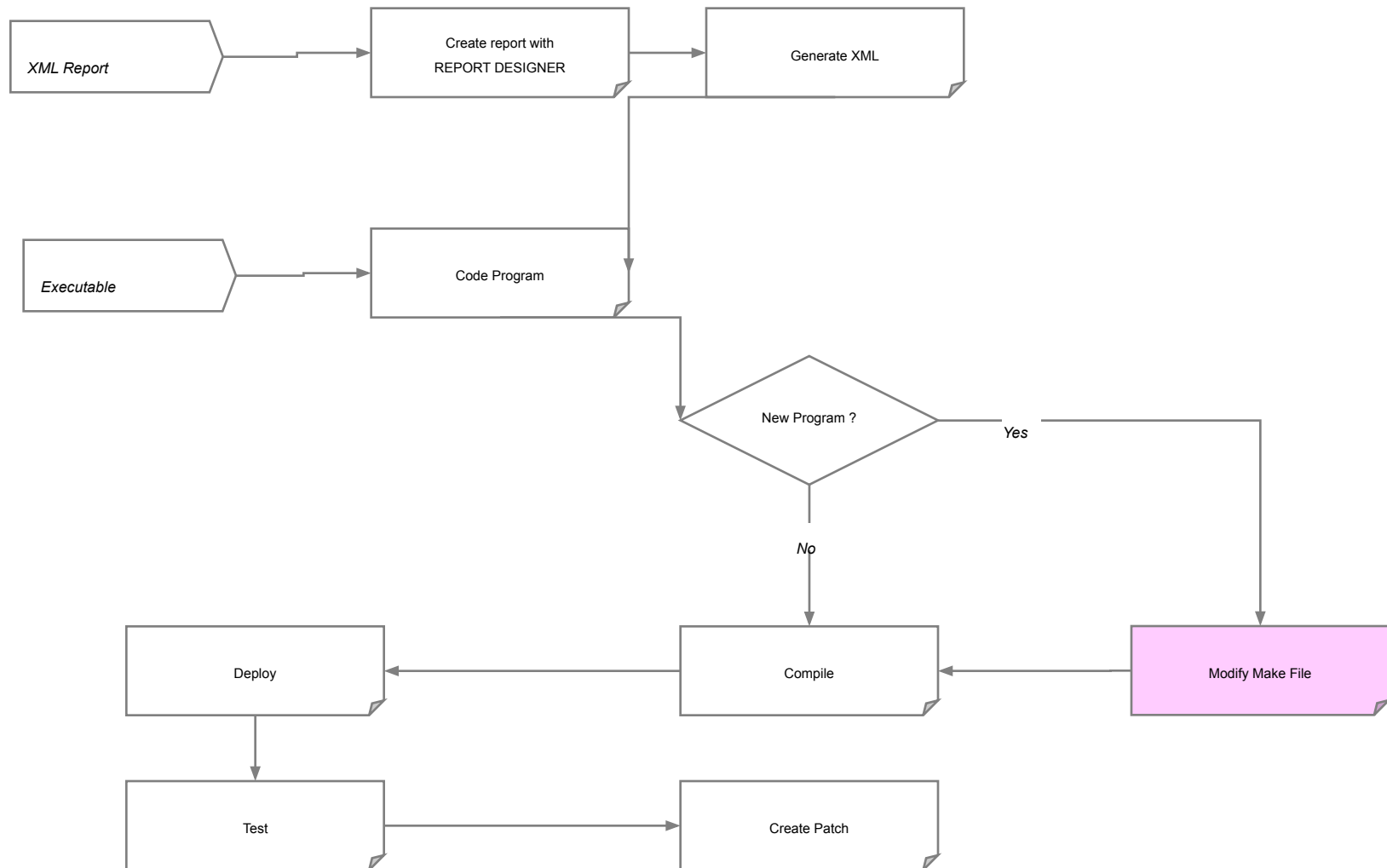


- Open the following file for more details on this exercise:
 - TS200_GOLDStockDevelopment_Exercises - 7

1. Review source code of sample reports



Create a program process



Modify Make File



- The makefile file is used to get a binary file from sources

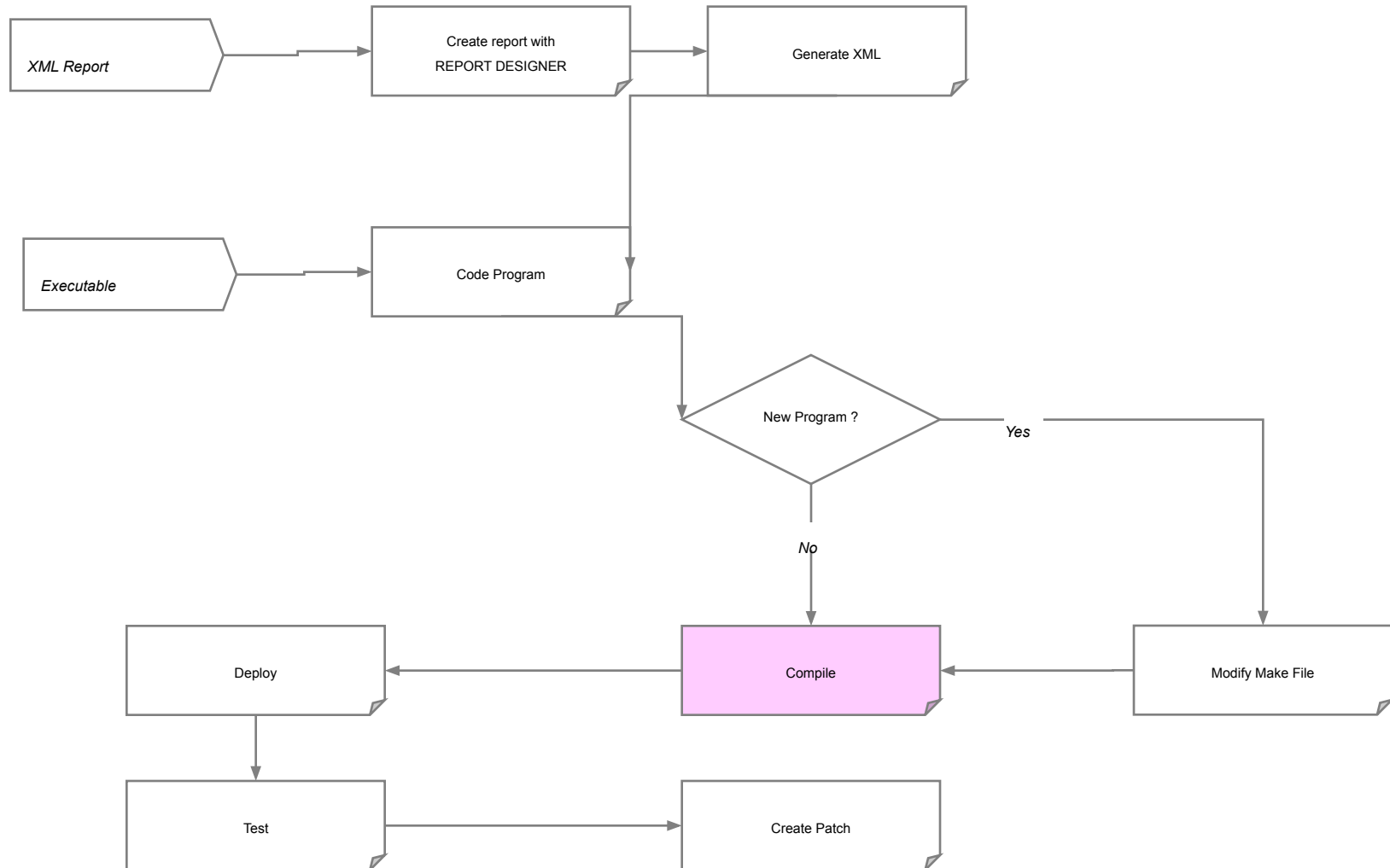
```
#####
@if [ -f $(TEMP)/$(@F).c ] ; then rm $(TEMP)/$(@F).c; fi
@if [ -f $(DESTEXE)/$(@F) ] ; then $(STRIP) $(DESTEXE)/$(@F); fi

#-----#
# Programmes linkés avec LIBEDT, LIBETI, LIBGEN1, LIBSTD
#=====
PRG_EDT =      $(DESTEXE)/ba_rec01 \
              $(DESTEXE)/pr_gdb01 \
..
              $(DESTEXE)/pr_rem21 \
              $(DESTEXE)/pr_tra11 \
              $(DESTEXE)/pr_tra12 \
              $(DESTEXE)/st_spe01

PRG_EDT :    $(PRG_EDT)

$(PRG_EDT) :  $(SRC_C)/$$(@F).pc $(DEPBASE) $(LIBEDT) $(LIBETI) $(LIBGEN1) $(LIBSTD) $(LIBEDT)
@echo "pc -> exe" $(@F)
@if [ -f $(DESTEXE)/$(@F).o ] ; then rm $(DESTEXE)/$(@F).o; fi
$(PCC) $(PROC_FLG) iname=$(SRC_C)/$(@F).pc oname=$(TEMP)/$(@F).c lname=$(TEMP)/$(@F).lis
$(CC_OBJECT) $(CC_FLG) $(TEMP)/$(@F).c -o $(DESTEXE)/$(@F)
@if [ -f $(DESTEXE)/$(@F) ] ; then mv $(DESTEXE)/$(@F) $(DESTEXE)/$(@F).o; fi
$(CC) $(CC_FLG) $(DESTEXE)/$(@F).o -o $(DESTEXE)/$(@F) \
#####
```

Create a program process



Compilation

```
[egold@DevLinux ~]$ cd $pc
[egold@DevLinux pc]$ make $bin/pr_par01
pc -> exe pr_par01
proc sqlcheck=full userid=STK507/STK507 errors=yes dbms=v8 char_map=VARC
HAR2 unsafe_null=yes select_error=no ltype=short ireclen=256 oreclen=256
  parse=partial define=GOLDDOUBLEBYTES define=STOCK define=UNIX_SYSTEM_V
CODE=ANSI_C hold_cursor=yes release_cursor=no lines=yes mode=oracle xref
=no maxopencursors=5000 varchar=yes include=/opt/GOLD/stk507/src/inc inc
lude=/opt/oracle/product/10g/sqllib/public include=/usr/include include=
/opt/oracle/product/10g/xdk/include iname=/opt/GOLD/stk507/src/pc/pr_par
01.pc oname=./tmp/pr_par01.c lname=./tmp/pr_par01.lis

Pro*C/C++: Release 10.2.0.1.0 - Production on Mon Jun 15 23:10:26 2009

Copyright (c) 1982, 2005, Oracle. All rights reserved.

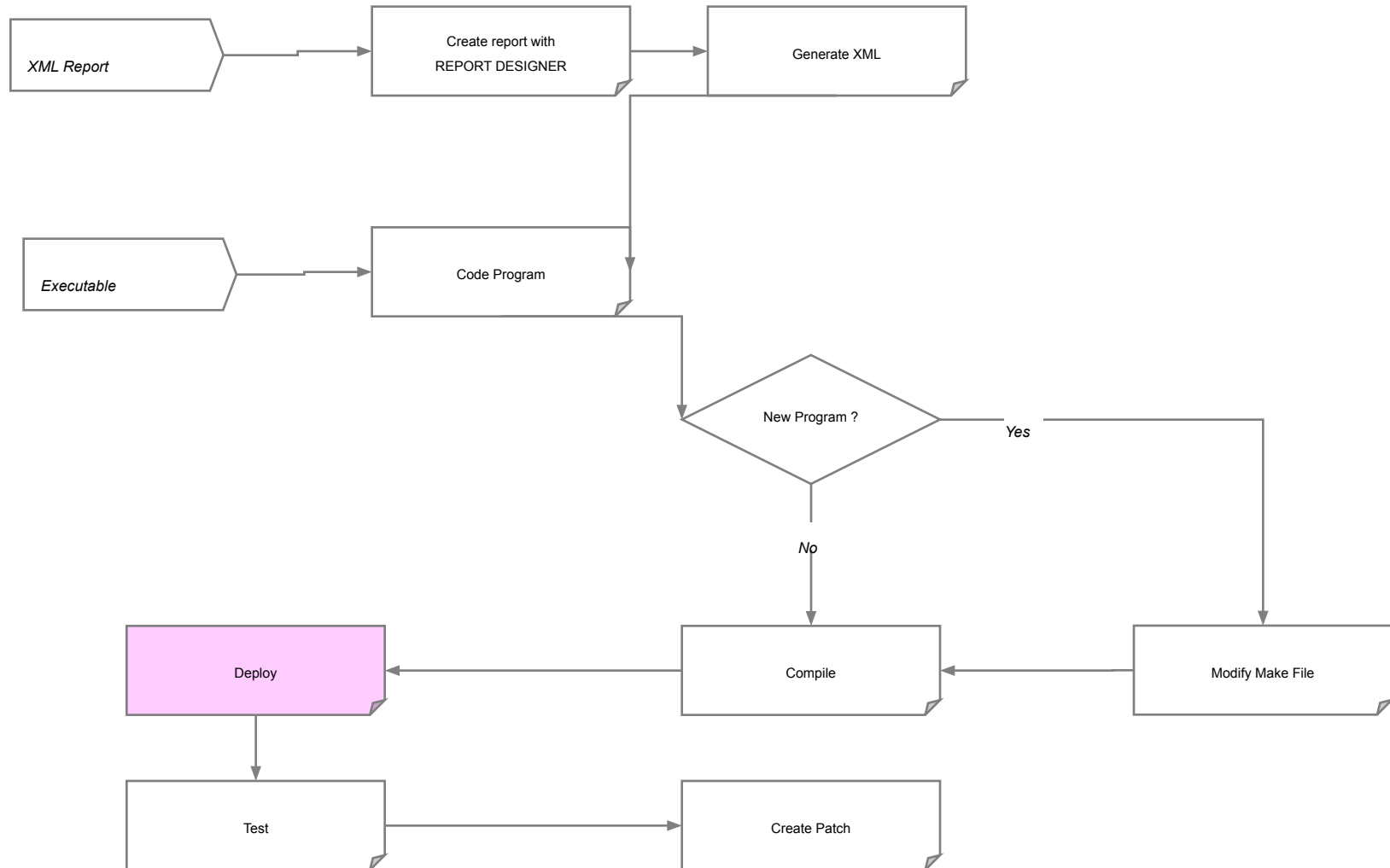
System default option values taken from: /opt/oracle/product/10g/precomp
/admin/pcscfg.cfg

cc -w -c -O2 -fPIC -DPRECOMP -DLINUX -D_GNU_SOURCE -D_LARGEFILE64_SO
URCE=1 -D_LARGEFILE_SOURCE=1 -DSLTS_ENABLE -DSLMMX_ENABLE -D_REENTRANT
-DNS_THREADS -I/opt/GOLD/stk507/src/inc -D "UNIX_SYSTEM_V" -D "STOCK"
-D "GOLDDOUBLEBYTES" -g ./tmp/pr_par01.c -o /opt/GOLD/stk507/bin/pr_p
ar01
cc -w -O2 -fPIC -DPRECOMP -DLINUX -D_GNU_SOURCE -D_LARGEFILE64_SOURC
E=1 -D_LARGEFILE_SOURCE=1 -DSLTS_ENABLE -DSLMMX_ENABLE -D_REENTRANT -DN
S_THREADS -I/opt/GOLD/stk507/src/inc -D "UNIX_SYSTEM_V" -D "STOCK" -D
"GOLDDOUBLEBYTES" -g /opt/GOLD/stk507/bin/pr_par01.o -o /opt/GOLD/stk
507/bin/pr_par01 \
-o /opt/GOLD/stk507/bin/pr_par01 -L/opt/oracle/product/10g/precomp
/lib/ -L/opt/oracle/product/10g/lib/ -L/opt/oracle/product/10g/lib/stu
```

The compilation, though one command is used, two compiles are done :

- Oracle Proc
- CC compiler

Create a program process



Deploy and test



- Program/Batch
 - \$BIN (\$GOLD_HOME\bin)
- Report XML Model
 - \$XML (\$GOLD_HOME\gaia\deploy\DEFAULT\web\estock\xml)
- Test compiled binary from command line
- Review generated report and report data in \$LST directory
- Test report from Gold Stock application



EXERCISE: Customize report



- Open the following file for more details on this exercise:
 - TS200_GOLDStockDevelopment_Exercises - 8

1. Customize report
2. Alter source code
3. Modify makefile
4. Compile
5. Deploy and test





EXERCISE: Create new report

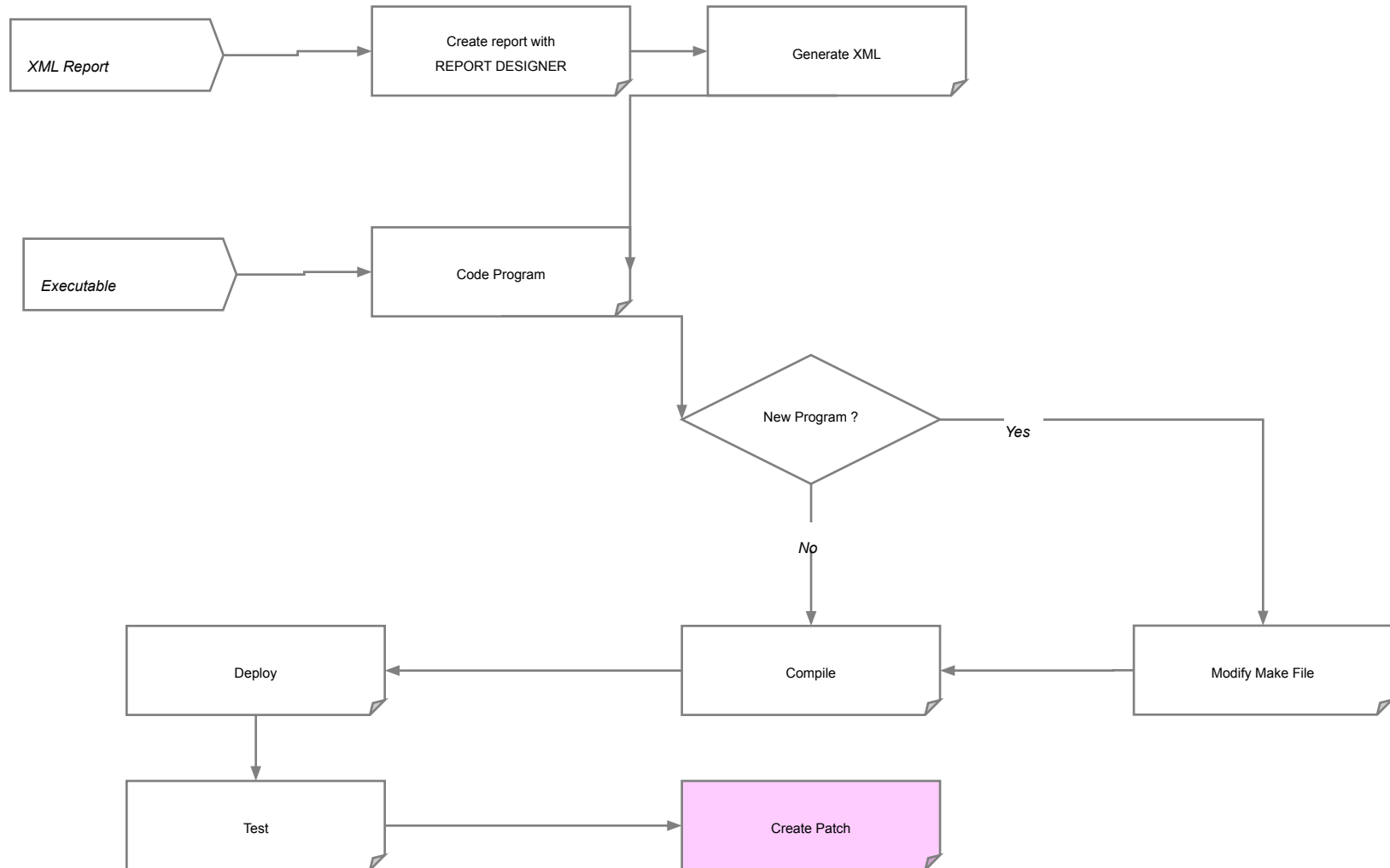


- Open the following file for more details on this exercise:
 - TS200_GOLDStockDevelopment_Exercises - 9

1. Design report template
2. Report source code
3. Modify makefile
4. Compile
5. Report parameterization
6. Deploy and test



Create a program process



Create Patch



- Files you **MUST** add to patch after a program was created :
 - Binary files
 - XML Model (if it is a report)
- Files you may add to patch after a program was created :
 - Script with “Descriptions/Labels” if new ones are used in report
 - Script with Message Usage
 - Script with Message Errors
 - Script to insert menu entry in the ESTOCK application



EXERCISE: Patch Content



- TS200_GOLDStockDevelopment_Exercises - 10
- If you create a new screen + report for your client, what will you put in the patch ?



- (5 files are mandatory)

- _____
- _____
- _____
- _____
- _____

• *Answer on next screen*

Patch Content



- Previous' page answers :
 1. xxx.server.yyy.jar
 2. xxx.client.yyy.jar
 3. estock.html
 4. prXXX
 5. modele.xml

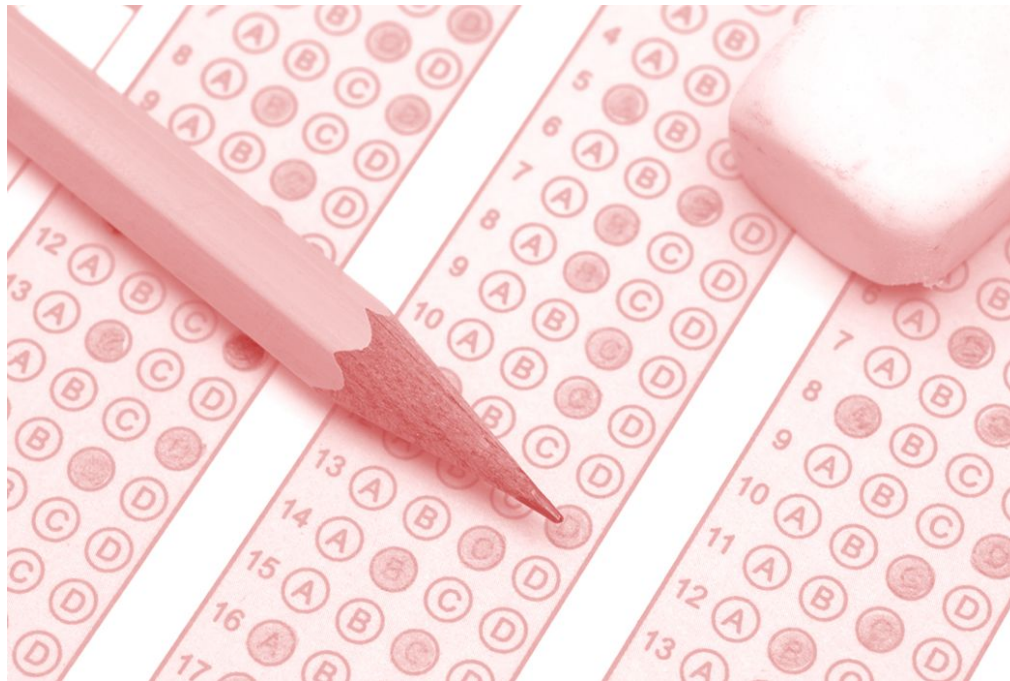
- You could also need to add the makefile and prXXX.o files in order to re-link with specific Oracle version on your client's server



Questions?



- Please complete the following quiz on the intranet:
 - TS200_GOLDStockDevelopment_Quiz





Additional Documentation

- Ergonomy of JAVA Screens for G.O.L.D.
 - Ref. GB-GSP-505-EXP-ERGO-241-1.pdf
- A.D.E.R. 2.0
 - Ref. GB-GSP-505-EXP-ADER-240-1
- Reporting engine
 - Ref. GB-GSP-505-EXP-MOREP-242-1
- Graphic Framework Documentation
 - Ref. GB-GSP-505-EXP-GFWK-243-1

