**This template navigates the UML model package by package.**  <drop/>

**This template is Letter 8.5”x 11” (21.59 cm \* 27.94cm), Landscape Orientation** <drop/>

**This template will display only the model elements that are packages or that are within packages**. <drop/>

All the text in red is comments and won't appear in the output. Do not remove the drop end-of-line. <drop/>

Three parameters need to be set to use this template: <drop/>

* **model\_name**: change the name to that of the papyrus model file <drop/>
  + **e**.g., Common, Infrastructure, Party, Resource, Service, Vnf <drop/>
* **output\_doc\_name**: change if so desired <drop/>
* **output\_folder**: set to desired value <drop/>

Only change text in **brown** below. <drop/>

The generated documentation will be in the doc folder in the eclipse workspace project folder <drop/>

Within Eclipse, if necessary, refresh (F5 key) the project folder after generation of the document from the template. <drop/>

Note that none of the red text in this document will appear in the output <drop/>

Note that commands that do not result in text being printed are in purple. Text and commands that cause print are in black <drop/>

<config services='TagFileBuffer'><drop/>

<param key='model\_name' value='Common'/><drop/>

<param key='output\_doc\_name' value='${model\_name}Model-${date}.docx'/><drop/>

<param key='output\_folder' value=’c:\users\ks0567\gendoc'/><drop/>

<output path='${output\_folder}/${output\_doc\_name}'/><drop/>

</config><drop/>

<context model='${project\_loc}/${model\_name}.uml' element='{0}' importedBundles='commons;gmf;papyrus' searchMetamodels='true'/><drop/>

<gendoc><drop/>

ONAP Information Model  
[gGet('model\_name')/]

This document was generated on [gGet('date').oclAsType(String).substring(1,10)/] by “ONAP Model GenDoc Template” version 3.01

# Introduction

This document contains the Word presentation of the model generated from the ONAP Eclipse Papyrus UML Information model using gendoc. This format is provided to assist the reader that does not use UML or has no access to UML tools.

Add specific text related to your contribution here that you would like to see in the output document <drop/>

</gendoc><drop/>

# Main display routine <drop/>

<gendoc><drop/>

[for (p:Package | self.ownedElement->filter(Package)->sortedBy(name))]<drop/>

# [p.name/]

[p.displayInfo(3)/]<drop/>

[for (p2:Package | p.ownedElement->filter(Package)->sortedBy(name))]<drop/>

## [p2.name/]

[p2.displayInfo(4)/]<drop/>

[for (p3:Package | p2.ownedElement->filter(Package)->sortedBy(name))]<drop/>

### [p3.name/]

[p3.displayInfo(5)/]<drop/>

[for (p4:Package | p3.ownedElement->filter(Package)->sortedBy(name))]<drop/>

#### [p4.name/]

[p4.displayInfo(6)/]<drop/>

[for (p5:Package | p4.ownedElement->filter(Package)->sortedBy(name))]<drop/>

##### [p5.name/]

[p5.displayInfo(7)/]<drop/>

[for (p6:Package | p5.ownedElement->filter(Package)->sortedBy(name))]<drop/>

###### [p6.name/]

[p6.displayInfo(8)/]<drop/>

[for (p7:Package | p6.ownedElement->filter(Package)->sortedBy(name))]<drop/>

[p7.name/]

[p7.displayInfo(9)/]<drop/>

#End level 7 header loop<drop/>

[/for]<drop/>

#End level 6 header loop<drop/>

[/for]<drop/>

#End level 5 header loop<drop/>

[/for]<drop/>

#End level 4 header loop<drop/>

[/for]<drop/>

#End level 3 header loop<drop/>

[/for]<drop/>

#End level 2 header loop<drop/>

[/for]<drop/>

#End level 1 header loop<drop/>

[/for]<drop/>

</gendoc><drop/>

# End main display routine <drop/>

# Display text with headers fragment <drop/>

Display text using the header of the level provided <drop/>

<fragment name='displayText' importedBundles='commons;gmf;papyrus'><drop/>

<arg name='element' type='uml::Element'/><drop/>

<arg name='text1' type='String'/><drop/>

<arg name='text2' type='String'/><drop/>

<arg name='level' type='Integer'/><drop/>

[if (level =2)]<drop/>

## [text1/] [text2/]

[/if]<drop/>

[if (level =3)]<drop/>

### [text1/] [text2/]

[/if]<drop/>

[if (level =4)]<drop/>

#### [text1/] [text2/]

[/if]<drop/>

[if (level =5)]<drop/>

##### [text1/] [text2/]

[/if]<drop/>

[if (level =6)]<drop/>

###### [text1/] [text2/]

[/if]<drop/>

[if (level =7)]<drop/>

[text1/] [text2/]

[/if]<drop/>

[if (level >7)]<drop/>

**[text1/] [text2/]**

[/if]<drop/>

</fragment><drop/>

# Display single element attributes fragment <drop/>

Display the attributes of a single element (class, notification, datatype). <drop/>

Element is the element for which attributes will be displayed. <drop/>

Base is the original class. If B inherits from A, to display all attributes of B, <drop/>

the fragment will be called twice: first element=base=B, element=A, base=B <drop/>

<fragment name='displaySingleElementAttributes' importedBundles='commons;gmf;papyrus'><drop/>

<arg name='element' type='uml::Element'/><drop/>

<arg name='base' type='uml::Element'/><drop/>

[for (p:uml::Property|element.eContents()->filter(uml::Property))]<drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [p.name/] | [if (not (p.type.name.oclIsUndefined()))]<drop/>  [p.type.name/]  [/if] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if (p.ownedComment->notEmpty())]<drop/>  [for (c:Comment | p.ownedComment)]<drop/>  [c.\_body.cleanAndFormat()/]  [/for]  [/if] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('attribute')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/] [else]<drop/> [/if] * [if oa.name.contains(‘isInvariant’)]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/][else]<drop/> [/if] * [if oa.name.contains('value')]valueRange: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/][else] no range constraint [/if] [else]<drop/> [/if] * [if oa.name.contains('support')]support: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/] [else]<drop/> [/if] * [if oa.name.contains('condition')][if (not p.getValue(st, oa.name).oclIsUndefined())]condition: [p.getValue(st, oa.name).oclAsType(String).clean()/] [else] <drop/>[/if] [else]<drop/> [/if] * [if oa.name.contains('state')]LifecycleState: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][else]<drop/> [/if] * [if oa.name.contains('passedByRef')] [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(Boolean)/][else] undefined [/if][else]<drop/> [/if] * [if oa.name.contains('reference')][if (not p.getValue(st, oa.name).oclIsUndefined())]reference: [p.getValue(st, oa.name).oclAsType(String).clean()/][else]<drop/>[/if][else]<drop/> [/if]   [/for]  [/for] |

[/for]<drop/>

</fragment><drop/>

# Display comments fragment <drop/>

Display all the comments of the element <drop/>

<fragment name='displayComments' importedBundles='commons;gmf;papyrus'><drop/>

<arg name='element' type='uml::Element'/><drop/>

[if (element.ownedComment->notEmpty())]<drop/>

**Description:**

[for (description : String | element.eContents()->filter(uml::Comment)->select(c | c.annotatedElement->includes(element)).\_body.splitNewLine())]<drop/>

[description.cleanAndFormat()/]

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display stereotypes fragment <drop/>

Display a list of stereotypes <drop/>

Used for: Package, DataType, Association <drop/>

Not used for: Class, Notification <drop/>

<fragment name='displayStereotypes' importedBundles='commons;gmf;papyrus'><drop/>

<arg name='element' type='uml::Element'/><drop/>

[if (element.getAppliedStereotypes()->notEmpty())]<drop/>

**Applied Stereotypes:**

[for (st:Stereotype | element.getAppliedStereotypes()->sortedBy(name))]<drop/>

* [st.name/]

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display all attributes fragment <drop/>

Display all the attribute information in a class, a notification or a datatype.<drop/>

It includes attributes from parents. Only two levels of parent are supported .<drop/>

<fragment name='displayAllAttributes' importedBundles='commons;gmf;papyrus' importedFragments='displayComments;displaySingleElementAttributes'><drop/>

<arg name='element' type='uml::Element'/><drop/>

[if element.eContents()->filter(uml::Property)->notEmpty()]<drop/>

**Table 7‑1 Attributes for [element.getText()/]**

<table><drop/>

| **Attribute Name** | **Type** | **Mult.** | **Description** | **Stereotypes** |
| --- | --- | --- | --- | --- |

[element.displaySingleElementAttributes(element) /]<drop/>

[if (element.oclAsType(uml::Classifier).general->notEmpty())]<drop/>

[for (parent:uml::Classifier | element.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent.oclIsUndefined()))]<drop/>

[parent.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

[if (parent.oclAsType(uml::Classifier).general->notEmpty())]<drop/>

[for (parent2:uml::Classifier | parent.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent2.oclIsUndefined()))]<drop/>

[parent2.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

[/for] <drop/>

[/if]<drop/>

[/for] <drop/>

[/if]<drop/>

</table><drop/>

[else]

[if (element.oclAsType(uml::Classifier).general->notEmpty())]<drop/>

**Table 7‑2 Attributes for [element.getText()/]**

<table><drop/>

| **Attribute Name** | **Type** | **Mult.** | **Description** | **Stereotypes** |
| --- | --- | --- | --- | --- |

[for (parent:uml::Classifier | element.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent.oclIsUndefined()))]<drop/>

[parent.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

[if (parent.oclAsType(uml::Classifier).general->notEmpty())]

[for (parent2:uml::Classifier | parent.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent2.oclIsUndefined()))]<drop/>

[parent2.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

[/for] <drop/>

[/if]<drop/>

[/for] <drop/>

</table><drop/>

[/if]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display package fragment <drop/>

Display all the information of a package: qualififedName, comments, stereotypes <drop/>

<fragment name='displayPackage' importedBundles='commons;gmf;papyrus' importedFragments='displayText;displayComments;displayStereotypes'><drop/>

<arg name='element' type='uml::Element'/><drop/>

<arg name='level' type='Integer'/><drop/>

[element.displayText('Overview', '',level-1)/]<drop/>

**Qualified Name:** [element.oclAsType(Package).qualifiedName/]

[element.displayComments()/]<drop/>

[element.displayStereotypes()/]<drop/>

</fragment><drop/>

# Display classes fragment <drop/>

Display all the information in a class: comments, stereotypes and attributes <drop/>

<fragment name='displayClasses' importedBundles='commons;gmf;papyrus' importedFragments='displayComments;displayAllAttributes;displayText'><drop/>

<arg name='element' type='uml::Element'/><drop/>

<arg name='level' type='Integer'/><drop/>

[if element.ownedElement->filter(Class)->notEmpty()]<drop/>

[element.displayText('Classes', '',level)/]<drop/>

[for (cl:uml::Class | element.ownedElement->filter(Class) ->sortedBy(name))]<drop/>

[element.displayText(cl.name, 'class', level+1)/]<drop/>

**Qualified Name:** [cl.qualifiedName/]

[cl.displayComments()/]<drop/>

[if (cl.isAbstract)]<drop/>

This class is abstract.

[/if]<drop/>

[if (cl.oclAsType(uml::Classifier).general->notEmpty())]<drop/>

**Parent class:** [cl.oclAsType(uml::Classifier).general ->asSequence()->first().name/]

[/if]<drop/>

**Applied Stereotypes:**

[for (st:uml::Stereotype | cl.getAppliedStereotypes()->sortedBy(name))]<drop/>

* [st.name/]

[for (oa:uml::Property|st.ownedAttribute)]<drop/>

* + [if oa.name.contains('attribute')]AVC: [cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/] [else]<drop/> [/if]
  + [if oa.name.contains(‘isInvariant’)]isInvariant: [cl.getValue(st, oa.name).oclAsType(Boolean)/][else]<drop/> [/if]
  + [if oa.name.contains('value')]valueRange: [if (not cl.getValue(st, oa.name).oclIsUndefined())][cl.getValue(st, oa.name).oclAsType(String).clean()/][else] no range constraint [/if] [else]<drop/>[/if]
  + [if oa.name.contains('support')]support: [cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/] [else]<drop/>[/if]
  + [if oa.name.contains('condition')][if (not cl.getValue(st, oa.name).oclIsUndefined())]condition: [cl.getValue(st, oa.name).oclAsType(String).clean()/] [else] <drop/>[/if] [else]<drop/>[/if]
  + [if oa.name.contains('state')]LifecycleState: [cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][else]<drop/>[/if]
  + [if oa.name.contains('passedByRef')] [if (not cl.getValue(st, oa.name).oclIsUndefined())][cl.getValue(st, oa.name).oclAsType(Boolean)/][else] undefined [/if][else]<drop/>[/if]
  + [if oa.name.contains('reference')][if (not cl.getValue(st, oa.name).oclIsUndefined())]reference: [cl.getValue(st, oa.name).oclAsType(String).clean()/][else]<drop/>[/if][else]<drop/>[/if]

[/for]<drop/>

[/for]

[cl.displayAllAttributes()/]

[/for] <drop/>

[/if]<drop/>

</fragment><drop/>

# Display notifications fragment <drop/>

Display all the information in a notification: comments, stereotypes and attributes. <drop/>

Notification = UML Signal – notification should be amended to signal.<drop/>

<fragment name='displayNotifications' importedBundles='commons;gmf;papyrus' importedFragments='displayComments;displayAllAttributes;displayText'><drop/>

<arg name='element' type='uml::Element'/><drop/>

<arg name='level' type='Integer'/><drop/>

[if element.ownedElement->filter(Signal)->notEmpty()]<drop/>

[element.displayText('Notifications', '',level)/]<drop/>

[for (cl:uml::Signal | element.ownedElement->filter(Signal) ->sortedBy(name))]<drop/>

[element.displayText(cl.name, 'signal', level+1)/]<drop/>

**Qualified Name:** [cl.qualifiedName/]

[cl.displayComments()/]<drop/>

**Applied Stereotypes:**

[for (st:uml::Stereotype | cl.getAppliedStereotypes()->sortedBy(name))]<drop/>

* [st.name/]   
  [for (oa:uml::Property|st.ownedAttribute)]<drop/>
  + [if oa.name.contains('attribute')]AVC: [cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/] [else]<drop/> [/if]
  + [if oa.name.contains(‘isInvariant’)]isInvariant: [cl.getValue(st, oa.name).oclAsType(Boolean)/][else]<drop/> [/if]
  + [if oa.name.contains('value')]valueRange: [if (not cl.getValue(st, oa.name).oclIsUndefined())][cl.getValue(st, oa.name).oclAsType(String).clean()/][else] no range constraint [/if] [else]<drop/>[/if]
  + [if oa.name.contains('support')]support: [cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/] [else]<drop/>[/if]
  + [if oa.name.contains('condition')][if (not cl.getValue(st, oa.name).oclIsUndefined())]condition: [cl.getValue(st, oa.name).oclAsType(String).clean()/] [else] <drop/>[/if] [else]<drop/>[/if]
  + [if oa.name.contains('state')]LifecycleState: [cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][else]<drop/>[/if]
  + [if oa.name.contains('passedByRef')] [if (not cl.getValue(st, oa.name).oclIsUndefined())][cl.getValue(st, oa.name).oclAsType(Boolean)/][else] undefined [/if][else]<drop/>[/if]
  + [if oa.name.contains('reference')][if (not cl.getValue(st, oa.name).oclIsUndefined())]reference: [cl.getValue(st, oa.name).oclAsType(String).clean()/][else]<drop/>[/if][else]<drop/>[/if]

[/for]<drop/>

[/for]

[cl.displayAllAttributes()/]

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display datatypes fragment <drop/>

Display all the information in datatypes: comments, stereotypes and attributes <drop/>

Valid for primitive & standard datatypes <drop/>

<fragment name='displayDatatypes' importedBundles='commons;gmf;papyrus' importedFragments='displayComments;displayAllAttributes;displayText;displayStereotypes'><drop/>

<arg name='element' type='uml::Element'/><drop/>

<arg name='level' type='Integer'/><drop/>

[if element.ownedElement->filter(DataType)->notEmpty()]<drop/>

[element.displayText('Datatypes', '',level)/]<drop/>

[for (dt:uml::DataType | element.ownedElement->filter(DataType) ->sortedBy(name))]<drop/>

[if dt.oclIsTypeOf(DataType)]<drop/>

[element.displayText(dt.name, 'datatype', level+1)/]<drop/>

**Qualified Name:** [dt.qualifiedName/]

[dt.displayComments()/]<drop/>

[dt.displayStereotypes()/]<drop/>

[dt.displayAllAttributes()/]

[/if] <drop/>

[if dt.oclIsTypeOf(PrimitiveType)]<drop/>

[element.displayText(dt.name, 'primitive type', level+1)/]<drop/>

**Qualified Name:** [dt.qualifiedName/]

[dt.displayComments()/]<drop/>

[dt.displayStereotypes()/]<drop/>

[/if]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display associations fragment <drop/>

Display all the information in an association: comments, stereotypes and attributes <drop/>

<fragment name='displayAssociations' importedBundles='commons;gmf;papyrus' importedFragments='displayComments;displayText;displayStereotypes'><drop/>

<arg name='element' type='uml::Element'/><drop/>

<arg name='level' type='Integer'/><drop/>

[if (element.ownedElement->filter(Association)->notEmpty())]<drop/>

[element.displayText('Associations', '',level)/]<drop/>

[for (as:uml::Association | element.ownedElement->filter(Association) ->sortedBy(name))]<drop/>

[element.displayText(as.name, 'association', level+1)/]<drop/>

**Qualified Name:** [as.qualifiedName/]

[as.displayComments()/]<drop/>

[as.displayStereotypes()/]<drop/>

[if (as.memberEnd->notEmpty())]<drop/>

**Table 12‑1 Member ends for [as.getText()/]**

<table><drop/>

| **Attribute Name** | **Aggreg.** | **Navig.** | **Mult.** | **Type** | **Description** | **Stereotypes** |
| --- | --- | --- | --- | --- | --- | --- |

[for (p:uml::Property|as.memberEnd)]<drop/>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [p.name/] | [p.aggregation/] | [if (p.isNavigable())]<drop/>  Navig.[else]<drop/>  Not navig. [/if] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if (not (p.type.name.oclIsUndefined()))]<drop/>  [p.type.name/]  [/if] | [if (p.ownedComment->notEmpty())]<drop/>  [for (c:Comment | p.ownedComment)]<drop/>  [c.\_body.cleanAndFormat()/]  [/for] <drop/>  [/if] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('attribute')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/] [else]<drop/> [/if] * [if oa.name.contains(‘isInvariant’)]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/][else]<drop/> [/if] * [if oa.name.contains('value')]valueRange: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/][else] no range constraint [/if] [else]<drop/> [/if] * [if oa.name.contains('support')]support: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/] [else]<drop/> [/if] * [if oa.name.contains('condition')][if (not p.getValue(st, oa.name).oclIsUndefined())]condition: [p.getValue(st, oa.name).oclAsType(String).clean()/] [else] <drop/>[/if] [else]<drop/> [/if] * [if oa.name.contains('state')]LifecycleState: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][else]<drop/> [/if] * [if oa.name.contains('passedByRef')] [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(Boolean)/][else] undefined [/if][else]<drop/> [/if] * [if oa.name.contains('reference')][if (not p.getValue(st, oa.name).oclIsUndefined())]reference: [p.getValue(st, oa.name).oclAsType(String).clean()/][else]<drop/>[/if][else]<drop/> [/if]   [/for]  [/for] |

[/for]<drop/>

</table><drop/>

[/if]

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display diagrams fragment <drop/>

Display all the diagrams in a package<drop/>

<fragment name='displayDiagrams' importedBundles='commons;gmf;papyrus' importedFragments='displayText'><drop/>

<arg name='element' type='uml::Element'/><drop/>

<arg name='level' type='Integer'/><drop/>

[if element.getPapyrusDiagrams()->notEmpty()]<drop/>

[element.displayText('Diagrams', '',level)/]<drop/>

[for (d:Diagram|element.getPapyrusDiagrams())]<drop/>

<image object='[d.getDiagram()/]' maxW='true'><drop/>

</image><drop/>

**Figure 13‑1: Diagram [d.name.cleanAndFormat()/]**

<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display enumerations fragment <drop/>

Display all the information in enumerations: comments, stereotypes and attributes <drop/>

Valid for primitive & standard datatypes <drop/>

<fragment name='displayEnumerations' importedBundles='commons;gmf;papyrus' importedFragments='displayComments;displayAllAttributes;displayText;displayStereotypes'><drop/>

<arg name='element' type='uml::Element'/><drop/>

<arg name='level' type='Integer'/><drop/>

[if element.ownedElement->filter(Enumeration)->notEmpty()]<drop/>

[element.displayText('Enumerations', '',level)/]<drop/>

[for (dt:uml::DataType | element.ownedElement->filter(Enumeration) ->sortedBy(name))]<drop/>

[if dt.oclIsTypeOf(Enumeration)]<drop/>

[element.displayText(dt.name, 'enumeration', level+1)/]<drop/>

**Qualified Name:** [dt.qualifiedName/]

[dt.displayComments()/]<drop/>

[dt.displayStereotypes()/]<drop/>

[if (dt.oclAsType(Enumeration).ownedLiteral)->notEmpty()]<drop/>

**Contains Enumeration Literals:**

[for (e:EnumerationLiteral|dt.oclAsType(Enumeration).ownedLiteral)]<drop/>

* [e.name/]

[for (co:Comment | e.ownedComment)]<drop/>

* + <dropEmpty> [co.\_body.cleanAndFormat()/]</dropEmpty>

[/for]<drop/>

[/for]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display all package information fragment <drop/>

Display all the information in a package: comments, classes, notifications, datatypes and diagrams <drop/>

<fragment name='displayInfo' importedBundles='commons;gmf;papyrus' importedFragments='displayComments;displayClasses;displayNotifications;displayDatatypes;displayAssociations;displayDiagrams;displayPackage;displayEnumerations'><drop/>

<arg name='element' type='uml::Element'/><drop/>

<arg name='level' type='Integer'/><drop/>

[element.displayPackage(level)/]<drop/>

[element.displayAssociations(level)/]<drop/>

[element.displayDiagrams(level)/]<drop/>

[element.displayClasses(level)/]<drop/>

[element.displayNotifications(level)/]<drop/>

[element.displayDatatypes(level)/]<drop/>

[element.displayEnumerations(level)/]<drop/>

</fragment><drop/>

# [element.displayAssociations(level)/]<drop/>

# [element.displayDiagrams(level)/]<drop/>

# [element.displayClasses(level)/]<drop/>

# [element.displayNotifications(level)/]<drop/>

# [element.displayDatatypes(level)/]<drop/>

# [element.displayEnumerations(level)/]<drop/>