|  |
| --- |
|  |
| |  |  |  | | --- | --- | --- | | 75 | Date: | $date.get(‘MMMM YYYY’) | |
| Archetype Modeling Language (AML)  #set ($version = “0.5”)  Version: $version  **OMG Document Number: formal/$date.get(‘YYYY-M-d’)**  **Standard document URL: http://www.omg.org/spec/AML/1.0**  Original File: N/A |

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company&apos;s products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE.   
  
IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.  
  
The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 140 Kendrick Street, Needham, MA 02494, U.S.A.

TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group©, OMG© , Unified Modeling Language©, Model Driven Architecture Logo©, Model Driven Architecture Diagram©, CORBA logo©, XMI Logo©, CWM©, CWM Logo©, IIOP© , MOF© , OMG Interface Definition Language (IDL)© , and OMG SysML© are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.  
  
Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

**OMG's Issue Reporting Procedure**

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.)

**Acknowledgements**

The following individuals submitted parts of this specification and/or have assisted the AML team in the development of the specification:

Table of Contents

1 $bookmark.create($obj1.ID, $dp1) 5

2 $dp2 6

2.1 $bookmark.create($obj2.ID, $dp3) 6

2.2 $dp4 6

2.2.1 $bookmark.create($obj3.ID, $dp5) 6

2.2.2 $dp6 6

**Preface**

**OMG**

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.  
  
OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG™s specifications implement the Model Driven Architecture (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG™s specifications include: UML® (Unified Modeling Language); CORBA® (Common Object Request Broker Architecture); CWM (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.  
  
More information on the OMG is available at http://www.omg.org/.

**OMG Specifications**

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A Specifications Catalog is available from the OMG website at:

*http://www.omg.org/technology/documents/spec\_catalog.htm*

Specifications within the Catalog are organized by the following categories:

**OMG Modeling Specifications**

• UML  
• MOF  
• XMI  
• CWM  
• Profile specifications

**OMG Middleware Specifications**

• CORBA/IIOP  
• IDL/Language Mappings  
• Specialized CORBA specifications  
• CORBA Component Model (CCM)

**Platform Specific Model and Interface Specifications**

• CORBAservices  
• CORBAfacilities  
• OMG Domain specifications  
• OMG Embedded Intelligence specifications  
• OMG Security specifications

OMG Headquarters   
 109 Highland Ave,   
 Needham, MA 02494 USA  
 USA   
   
 Tel: +1-781-444-0404   
 Fax: +1-781-444-0320   
 Email: pubs@omg.org  
   
Certain OMG specifications are also available as ISO standards. Please consult http://www.iso.org

**Typographical Conventions**

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

**Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.**

Courier - 10 pt. Bold: Programming language elements.

Helvetica/Arial - 10 pt : Exceptions

NOTE: Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

#import('js', 'com.nomagic.reportwizard.tools.script.JavaScriptTool')

#import("query", "com.nomagic.reportwizard.tools.QueryTool")

#import('text', 'com.nomagic.reportwizard.tools.TextTool')

#foreach ($pkg in $packageScope)

#packageList($pkg, 1)

#end

#macro (writeText $txt)

#set($txt1 = $text.html($txt))

$txt1

#end

#macro (writeBookmark1 $obj1 $dp1 $withNum)

#if($withNum == “true”)

# $bookmark.create($obj1.ID, $dp1)

#else

# $bookmark.create($obj1.ID, $dp1)

#end

#end

#macro (writeHeader1 $dp2 $withNum)

#if($withNum == “true”)

# $dp2

#else

# $dp2

#end

#end

#macro (writeBookmark2 $obj2 $dp3 $withNum)

#if($withNum == “true”)

## $bookmark.create($obj2.ID, $dp3)

#else

## $bookmark.create($obj2.ID, $dp3)

#end

#end

#macro (writeHeader2 $dp4 $withNum)

#if($withNum == “true”)

## $dp4

#else

## $dp4

#end

#end

#macro (writeBookmark3 $obj3 $dp5 $withNum)

#if($withNum == “true”)

### $bookmark.create($obj3.ID, $dp5)

#else

### $bookmark.create($obj3.ID, $dp5)

#end

#end

#macro (writeHeader3 $dp6 $withNum)

#if($withNum == “true”)

### $dp6

#else

### $dp6

#end

#end

#macro (writeBookmark4 $obj4 $dp7 $withNum)

#if($withNum == “true”)

#### $bookmark.create($obj4.ID, $dp7)

#else

#### $bookmark.create($obj4.ID, $dp7)

#end

#end

#macro (writeHeader4 $dp8 $withNum)

#if($withNum == “true”)

#### $dp8

#else

#### $dp8

#end

#end

#macro (writeBookmark $obj4 $dp7 $withNum)

#if($withNum == “true”)

1. **$bookmark.create($obj4.ID, $dp7)**

#else

**$bookmark.create($obj4.ID, $dp7)**

#end

#end

#macro (writeHeader $dp8 $withNum)

#if($withNum == “true”)

1. **$dp8**

#else

**$dp8**

#end

#end

##MACRO writeHeading

#macro (writeHeading $object $disp $isBookmark $headingLevel $headType $withNumbering)

#if($headType != “”) #set($disp = “<$headType> $disp”) #end

#if($headingLevel == 1) #if($isBookmark == “true”) #writeBookmark1($object $disp $withNumbering) #else #writeHeader1($disp $withNumbering) #end

#elseif($headingLevel == 2) #if($isBookmark == “true”) #writeBookmark2($object $disp $withNumbering) #else #writeHeader2($disp $withNumbering) #end

#elseif($headingLevel == 3) #if($isBookmark == “true”) #writeBookmark3($object $disp $withNumbering) #else #writeHeader3($disp $withNumbering) #end

#elseif($headingLevel == 4) #if($isBookmark == “true”) #writeBookmark4($object $disp $withNumbering) #else #writeHeader4($disp $withNumbering) #end

#else #if($isBookmark == “true”) #writeBookmark($object $disp $withNumbering) #else #writeHeader($disp $withNumbering) #end #end

#end

#set($printedInterfaces = $array.createArray())

#set($printedClasses = $array.createArray())

#set($printedEnums = $array.createArray())

#set($printedDataTypes = $array.createArray())

#set($printedStereoTypes = $array.createArray())

#macro(findNestedElement $object)

#set($innerElement = $report.getInnerElement($object))

#foreach($nested in $innerElement)

#if($nested.elementType != “package” && $nested.elementType != “model” && $nested.elementType != “profile” )

#if($nested.elementType == “interface”)

#set($tmp = $nestedInterface.add($nested))

#elseif($nested.elementType == “class”)

#set($tmp = $nestedClass.add($nested))

#elseif($nested.elementType == “enumeration”)

#set($tmp = $nestedEnum.add($nested))

#elseif($nested.elementType == “datatype”)

#set($tmp = $nestedDataTypes.add($nested))

#elseif($nested.elementType == “stereotype”)

#set($tmp = $nestedStereoTypes.add($nested))

#end

#set($in = $report.getInnerElement($nested))

#if($in.size() > 0)

#findNestedElement($nested)

#end

#end

#end

#end

#set($level = 0)

## MACRO packageList

#macro (packageList, $parentPackage, $plevel)

#set($packageInterface = $array.createArray())

#set($packageClass = $array.createArray())

#set($packageEnum = $array.createArray())

#set($packageDataTypes = $array.createArray())

#set($packageStereoTypes = $array.createArray())

#set($nestedInterface = $array.createArray())

#set($nestedClass = $array.createArray())

#set($nestedEnum = $array.createArray())

#set($nestedDataTypes = $array.createArray())

#set($nestedStereoTypes = $array.createArray())

#if(($parentPackage.elementType ==“package”)||($parentPackage.elementType == “profile”))

#foreach($element in $parentPackage.ownedElement)

#if($element.elementType == “interface”)

#set($tmp = $packageInterface.add($element))

#set($inner = $report.getInnerElement($element))

#if($inner.size() > 0)

#findNestedElement($element)

#end

#elseif($element.elementType == “class”)

#set($tmp = $packageClass.add($element))

#set($inner = $report.getInnerElement($element))

#if($inner.size() > 0)

#findNestedElement($element)

#end

#elseif($element.elementType == “enumeration”)

#set($tmp = $packageEnum.add($element))

#set($inner = $report.getInnerElement($element))

#if($inner.size() > 0)

#findNestedElement($element)

#end

#elseif($element.elementType == “datatype”)

#set($tmp = $packageDataTypes.add($element))

#set($inner = $report.getInnerElement($element))

#if($inner.size() > 0)

#findNestedElement($element)

#end

#elseif($element.elementType == “stereotype”)

#set($tmp = $packageStereoTypes.add($element))

#set($inner = $report.getInnerElement($element))

#if($inner.size() > 0)

#findNestedElement($element)

#end

#end

#end

#end

#set($tmp = $array.addCollection($packageInterface, $nestedInterface))

#set($tmp = $array.addCollection($packageClass, $nestedClass))

#set($tmp = $array.addCollection($packageEnum, $nestedEnum))

#set($tmp = $array.addCollection($packageDataTypes, $nestedDataTypes))

#set($tmp = $array.addCollection($packageStereoTypes, $nestedStereoTypes))

#set($diagramList = $array.createArray())

#foreach($d in $sorter.humanSort($parentPackage.ownedDiagram))

#if(($d.diagramType == “Class Diagram”)|| ($d.diagramType == “Profile Diagram”) || ($d.diagramType == “Package Diagram”))

#set($tmp = $diagramList.add($d))

#end

#end

#if($parentPackage != $project.model)

#set($displayTitle = $js.eval(‘pkgname.replace(/[0-9\.]/gi, “”)’, ‘pkgname’, $parentPackage.name))

#if($js.eval(‘(pkgn.charAt(0) > “7”)&&(pkgn.indexOf(“.”) != -1)’, ‘pkgn’, $parentPackage.name)) #writeHeading($displayTitle, $displayTitle, “false”, $plevel, “Package”, “true”) #else #writeHeading($displayTitle, $displayTitle, “false”, $plevel, “”, “true”) #end #if($parentPackage.documentation != “”)

#writeText($parentPackage.documentation)

#end #printDiagrams($diagramList) #end

#set ($subPackages = $parentPackage.nestedPackage)

#foreach ($pkg in $sorter.humanSort($subPackages))

#if ($plevel == 1)

#packageList($pkg, 2)

#elseif($plevel == 2)

#packageList($pkg, 3)

#elseif($plevel == 3)

#packageList($pkg, 4)

#else

#packageList($pkg, 5)

#end

#end

#end

###############################################

#\* MACRO printDiagrams

Prints diagrams for a given element

\*#

###############################################

#macro (printDiagrams $pkgdiagrams)

#if($pkgdiagrams)

#if($pkgdiagrams.size() > 0)

#foreach($diag in $sorter.humanSort($pkgdiagrams))

$image.setWidth($diag.image, -2)

**$diag.name**

#if($diag.documentation != “”)

#writeText($diag.documentation)

#end

#set($diagramElements = $array.createArray())

#set($diagramElements = $report.getDiagramElements($diag))

## Collecting Diagram elements

#set($interfaceLists = $array.createArray())

#set($classLists = $array.createArray())

#set($enumLists = $array.createArray())

#set($dataTypesLists = $array.createArray())

#set($stereoTypesLists = $array.createArray())

#foreach($element in $diagramElements)

#if($element.elementType == “interface”)

#set($tmp = $interfaceLists.add($element))

#set($tmp = $packageInterface.remove($element))

#elseif($element.elementType == “class”)

#set($tmp = $classLists.add($element))

#set($tmp = $packageClass.remove($element))

#elseif($element.elementType == “datatype”)

#set($tmp = $dataTypesLists.add($element))

#set($tmp = $packageDataTypes.remove($element))

#elseif($element.elementType == “stereotype”)

#set($tmp = $stereoTypesLists.add($element))

#set($tmp = $packageStereoTypes.remove($element))

#elseif($element.elementType == “enumeration”)

#set($tmp = $enumLists.add($element))

#set($tmp = $packageEnum.remove($element))

#end

#end

## Print Diagram elements

#if($dataTypesLists.size() > 0)

#set($knownDataTypes = $array.createArray())

#foreach($dtp in $sorter.humanSort($dataTypesLists))

#if(!$printedDataTypes.contains($dtp))

#set($tmp = $printedDataTypes.add($dtp))

#createCommonContent($dtp, “DataType”)

#else

#set($tmp = $knownDataTypes.add($dtp))

#end

#end

#if($knownDataTypes.size() > 0)

#writeHeading($knownDataTypes, “Known other Data Types”, “false”, 5, “”, “false”)

#set($size = $knownDataTypes.size())

#foreach($e in $knownDataTypes)$bookmark.open($e.ID, $e.name)#if($size != $velocityCount), #end#end

#end

#end##endDataTypes

#if($interfaceLists.size() > 0)

#set($knownInterface = $array.createArray())

#foreach($interface in $sorter.humanSort($interfaceLists))

#if(!$printedInterfaces.contains($interface))

#set($tmp = $printedInterfaces.add($interface))

#createCommonContent($interface, “Interface”)

#else

#set($tmp = $knownInterface.add($interface))

#end

#end

#if($knownInterface.size() > 0)

#writeHeading($knownInterface, “Known other interfaces” , “false”, 5, “”, “false”)

#set($size = $knownInterface.size())

#foreach($i in $knownInterface)$bookmark.open($i.ID, $i.name)#if($size != $velocityCount), #end#end

#end

#end##endinterface

#if($classLists.size() > 0)

#set($knownClass = $array.createArray())

#foreach($class in $sorter.humanSort($classLists))

#if(!$printedClasses.contains($class))

#set($tmp = $printedClasses.add($class))

#createCommonContent($class, “Class”)

#else

#set($tmp = $knownClass.add($class))

#end

#end

#if($knownClass.size() > 0)

#writeHeading($knownClass, “Known other classes” , “false”, 5, “”, “false”)

#set($size = $knownClass.size())

#foreach($c in $knownClass)$bookmark.open($c.ID, $c.name)#if($size != $velocityCount), #end#end

#end

#end##endclass

#if($enumLists.size() > 0)

#set($knownEnum = $array.createArray())

#foreach($enum in $sorter.humanSort($enumLists))

#if(!$printedEnums.contains($enum))

#set($tmp = $printedEnums.add($enum))

#createEnumerationContent($enum)

#else

#set($tmp = $knownEnum.add($enum))

#end

#end

#if($knownEnum.size() > 0)

#writeHeading($knownEnum, “Known other enumerations” , “false”, 5, “”, “false”)

#set($size = $knownEnum.size())

#foreach($e in $knownEnum)$bookmark.open($e.ID, $e.name)#if($size != $velocityCount), #end#end

#end

#end##endenumeration

#if($stereoTypesLists.size() > 0)

#set($knownStereoTypes = $array.createArray())

#foreach($stp in $sorter.humanSort($stereoTypesLists))

#if(!$printedStereoTypes.contains($stp))

#set($tmp = $printedStereoTypes.add($stp))

#createCommonContent($stp, “Stereotype”)

#else

#set($tmp = $knownStereoTypes.add($stp))

#end

#end

#if($knownStereoTypes.size() > 0)

#writeHeading($knownStereoTypes, “Known other Stereotypes” , “false”, 5, “”, “false”)

#set($size = $knownStereoTypes.size())

#foreach($e in $knownStereoTypes)$bookmark.open($e.ID, $e.name)#if($size != $velocityCount), #end#end

#end

#end##endStereoTypes

#end

## Print packageDataTypes, packageInterface, packageClass, packageEnum, packageStereotypes

#if($packageDataTypes.size() > 0)

#set($knownDataType = $array.createArray())

#foreach($dtp in $sorter.humanSort($packageDataTypes))

#if(!$printedDataTypes.contains($dtp))

#set($tmp = $printedDataTypes.add($dtp))

#createCommonContent ($dtp, “DataType”)

#else

#set($tmp = $knownDataType.add($dtp))

#end

#end

#if($knownDataType.size() > 0)

#writeHeading($knownDataType, “Known other Data Types” , “false”, 5, “”, “false”)

#set($size = $knownDataType.size())

#foreach($i in $knownDataType)$bookmark.open($i.ID, $i.name)#if($size != $velocityCount), #end#end

#end

#end##endpackageDataTypes

#if($packageInterface.size() > 0)

#set($knownInterface = $array.createArray())

#foreach($interface in $sorter.humanSort($packageInterface))

#if(!$printedInterfaces.contains($interface))

#set($tmp = $printedInterfaces.add($interface))

#createCommonContent($interface, “Interface”)

#else

#set($tmp = $knownInterface.add($interface))

#end

#end

#if($knownInterface.size() > 0)

#writeHeading($knownInterface, “Known other interfaces” , “false”, 5, “”, “false”)

#set($size = $knownInterface.size())

#foreach($i in $knownInterface)$bookmark.open($i.ID, $i.name)#if($size != $velocityCount), #end#end

#end

#end##endpackageInterface

#if($packageClass.size() > 0)

#set($knownClass = $array.createArray())

#foreach($class in $sorter.humanSort($packageClass))

#if(!$printedClasses.contains($class))

#set($tmp = $printedClasses.add($class))

#createCommonContent($class, “Class”)

#else

#set($tmp = $knownClass.add($class))

#end

#end

#if($knownClass.size() > 0)

#writeHeading($knownClass, “Known other classes” , “false”, 5, “”, “false”)

#set($size = $knownClass.size())

#foreach($c in $knownClass)$bookmark.open($c.ID, $c.name)#if($size != $velocityCount), #end#end

#end

#end##endpackageClass

#if($packageEnum.size() > 0)

#set($knownEnum = $array.createArray())

#foreach($enum in $sorter.humanSort($packageEnum))

#if(!$printedEnums.contains($enum))

#set($tmp = $printedEnums.add($enum))

#createEnumerationContent($enum)

#else

#set($tmp = $knownEnum.add($enum))

#end

#end

#if($knownEnum.size() > 0)

#writeHeading($knownEnum, “Known other enumerations” , “false”, 5, “”, “false”)

#set($size = $knownEnum.size())

#foreach($e in $knownEnum)$bookmark.open($e.ID, $e.name)#if($size != $velocityCount), #end#end

#end

#end##endpackageEnum

#if($packageStereoTypes.size() > 0)

#set($knownStereoType = $array.createArray())

#foreach($stp in $sorter.humanSort($packageStereoTypes))

#if(!$printedStereoTypes.contains($stp))

#set($tmp = $printedStereoTypes.add($stp))

#createCommonContent($stp, “Stereotype”)

#else

#set($tmp = $knownStereoType.add($stp))

#end

#end

#if($knownStereoType.size() > 0)

#writeHeading($knownStereoType, “Known other Stereotypes” , “false”, 5, “”, “false”)

#set($size = $knownStereoType.size())

#foreach($i in $knownStereoType)$bookmark.open($i.ID, $i.name)#if($size != $velocityCount), #end#end

#end

#end##endpackageStereoTypes

#end

#end

#end

###############################################

## MACRO printAttr

###############################################

#macro(printAttr $att)

• $att.visibility $att.name#if($att.type) : #if($js.eval(‘(typeQN.indexOf(“UML Standard Profile”) != -1)’, ‘typeQN’, $att.type.qualifiedName))$att.type.name#else$bookmark.open($att.type.ID, $att.type.name)#end #end#if($att.multiplicity != “”) [$att.multiplicity]#end#if($att.defaultValue) = $att.defaultValue.text#end

#end

###############################################

## MACRO printOper

###############################################

#macro(printOper $oper)

#set($paramLists = $oper.ownedParameter)

#set($size = 0)

#foreach($p in $paramLists)

#if($p.direction != “return”)

#set($size = $size +1)

#end

#end

#set($i = 1)

• $oper.visibility $oper.name (#foreach($param in $paramLists)#if($param.direction != “return”)$param.name#if($param.type) : $bookmark.open($param.type.ID, $param.type.name)#end#if($param.multiplicity != “”) [$param.multiplicity]#end#if($param.defaultValue) = $param.defaultValue.text#end#if($size != $i), #end#set($i = $i + 1)#end#end)#if($oper.type) : $bookmark.open($oper.type.ID, $oper.type.name)#end#if($oper.hasTypeModifier() && $oper.typeModifier != “”)$oper.typeModifier#end

#end

###############################################

##MACRO printAsso

###############################################

#macro(printAsso $attribute $association $object)

#foreach($member in $association.memberEnd)

#if($member.type != $object)

#set($memberEnd = $member)

#end

#end

• $memberEnd.visibility#if($attribute.name != “”) $attribute.name#end#if($memberEnd.type) : $bookmark.open($memberEnd.type.ID, $memberEnd.type.name)#end#if($memberEnd.multiplicity != “”) [$memberEnd.multiplicity]#end #if($memberEnd.defaultValue) = $memberEnd.defaultValue.text#end

#end

###############################################

## MACRO createDataTypeContent

###############################################

#macro(createDataTypeContent $dataTp)

#writeHeading($dataTp, $dataTp.name, “true”, 3, “Data Type”, “true”)

#if($dataTp.documentation != “”)

#writeHeading($dataTp, “Description”, “false”, 5, “”, “false”)

#writeText($dataTp.documentation)

#end

#end

###############################################

## MACRO createClassContent

###############################################

#macro(createClassContent $class)

#writeHeading($class, $class.name, “true”, 3, “Class”, “true”)

#if($class.documentation != “”)

#writeHeading($class, “Description”, “false”, 5, “”, “false”)

#writeText($class.documentation)

#end

#set($implementInterface = $class.realizedInterface)

#set($size = $implementInterface.size())

#if($implementInterface.size() > 0)

#writeHeading($implementInterface “Implemented Interface”, “false”, 5, “”, “false”)

#foreach($interface in $sorter.humanSort($implementInterface))$bookmark.open($interface.ID, $interface.name)#if($size != $velocityCount), #end#end#end

#set($baseClassifier = $class.baseClassifier)

#set($size = $baseClassifier.size())

#if($baseClassifier.size() > 0)

#writeHeading($baseClassifier, “Direct Known Superclasses (Generalization)”, “false”, 5, “”, “false”)

#foreach($class in $sorter.humanSort($baseClassifier))$bookmark.open($class.ID, $class.name)#if($size != $velocityCount), #end#end#end

#set($specClassifier = $class.specificClassifier)

#set($size = $specClassifier.size())

#if($specClassifier.size() > 0)

#writeHeading($specClassifier, “Direct Known Subclasses (Specialization)”, “false”, 5, “”, “false”)

#foreach($class in $sorter.humanSort($specClassifier))$bookmark.open($class.ID, $class.name)#if($size != $velocityCount), #end#end#end

#set($classAtt = $array.createArray())

#foreach($a in $class.ownedAttribute)

#if(!$a.association) #set($tmp = $classAtt.add($a)) #end #end

#if($classAtt.size() > 0)

#writeHeading($classAtt, “Attributes”, “false”, 5, “”, “false”)

#foreach($att in $classAtt) #if(!$att.association) #printAttr($att) #if($att.documentation != “”) #writeText($att.documentation)#end#end#end#end

#set($classOper = $class.ownedOperation)

#if($classOper.size() > 0)

#writeHeading($classOper, “Operations”, “false”, 5, “”, “false”)

#foreach($oper in $classOper) #printOper($oper)

#if($oper.documentation != “”) #writeText($oper.documentation) #end #end #end

#set($associationLists = $array.createArray())

#foreach($attribute in $class.ownedAttribute)

#if($attribute.association) #set($tmp = $associationLists.add($attribute.association)) #end #end

#if($associationLists.size() > 0)

#writeHeading($associationLists, “Associations”, “false”, 5, “”, “false”)

#foreach($attribute in $class.ownedAttribute) #if($attribute.association) #set($association = $attribute.association) #printAsso($attribute, $association, $class) #if($memberEnd.type.documentation != “”) #writeText($memberEnd.type.documentation) #end #end #end #end #end

###############################################

##MACRO createEnumerationContent

###############################################

#macro(createEnumerationContent $enum)

#writeHeading($enum, $enum.name, “true”, 3, “Enumeration”, “true”)

#if($enum.documentation != “”)

#writeHeading($enum, “Description”, “false”, 5, “”, “false”)

#writeText($enum.documentation) #end

#set($enumLiterals = $enum.ownedLiteral)

#if($enumLiterals.size() > 0) #writeHeading($implementInterface “Enumeration Literals”, “false”, 5, “”, “false”) #foreach($enumLit in $sorter.humanSort($enumLiterals))

* **$enumLit.name**

#writeText($enumLit.documentation) #end #end #end

###############################################

## MACRO createCommonContent

###############################################

#macro(createCommonContent $umlType $typeName)

#writeHeading($umlType, $umlType.name, “true”, 3, $typeName, “true”)

#if($umlType.documentation != “”)

#writeHeading($umlType, “Description”, “false”, 5, “”, “false”)

#writeText($umlType.documentation)

#end

#if (($typeName != “DataType”)&&($typeName != “Interface”))

#set($implementInterface = $umlType.realizedInterface)

#set($size = $implementInterface.size())

#if($implementInterface.size() > 0)

#writeHeading($implementInterface “Implemented Interface”, “false”, 5, “”, “false”)

#foreach($interface in $sorter.humanSort($implementInterface))$bookmark.open($interface.ID, $interface.name)#if($size != $velocityCount), #end#end#end

#end

#set($baseClassifier = $umlType.baseClassifier)

#set($size = $baseClassifier.size())

#if($baseClassifier.size() > 0)

#writeHeading($baseClassifier, “Direct Known Superclasses (Generalization)”, “false”, 5, “”, “false”)

#foreach($bclass in $sorter.humanSort($baseClassifier))$bookmark.open($bclass.ID, $bclass.name)#if($size != $velocityCount), #end#end#end

#set($specClassifier = $umlType.specificClassifier)

#set($size = $specClassifier.size())

#if($specClassifier.size() > 0)

#writeHeading($specClassifier, “Direct Known Subclasses (Specialization)”, “false”, 5, “”, “false”)

#foreach($sclass in $sorter.humanSort($specClassifier))$bookmark.open($sclass.ID, $sclass.name)#if($size != $velocityCount), #end#end#end

#set($allAtt= $array.createArray())

#foreach($a in $umlType.ownedAttribute)

#if(!$a.association) #set($tmp = $allAtt.add($a)) #end #end

#if($allAtt.size() > 0)

#writeHeading($allAtt, “Attributes”, “false”, 5, “”, “false”)

#foreach($att in $allAtt) #if(!$att.association) #printAttr($att) #if($att.documentation != “”) #writeText($att.documentation)#end#end#end#end

#set($allOper = $umlType.ownedOperation)

#if($allOper.size() > 0)

#writeHeading($allOper, “Operations”, “false”, 5, “”, “false”)

#foreach($oper in $allOper) #printOper($oper)

#if($oper.documentation != “”) #writeText($oper.documentation) #end #end #end

#set($associationLists = $array.createArray())

#foreach($attribute in $umlType.ownedAttribute)

#if($attribute.association) #set($tmp = $associationLists.add($attribute.association)) #end #end

#if($associationLists.size() > 0)

#writeHeading($associationLists, “Associations”, “false”, 5, “”, “false”)

#foreach($attribute in $umlType.ownedAttribute) #if($attribute.association) #set($association = $attribute.association) #printAsso($attribute, $association, $umlType) #if($memberEnd.type.documentation != “”) #writeText($memberEnd.type.documentation) #end #end #end #end

#set($rules = $array.createArray())

#foreach($rule in $umlType.ownedRule) #set($tmp = $rules.add($rule)) #end

#if($rules.size() > 0)

#writeHeading($rules, “Constraints”, “false”, 5, “”, “false”)

#foreach ($rulei in $rules)

• $rulei.name #if ($rulei.specification) #if($rulei.documentation != “”)#writeText($rulei.documentation)#end #if($report.hasProperty($rulei.specification, “language”)) $rulei.specification.language#end

$rulei.specification.text #end#end#end#end