[TopCoder]

Software Documentation : Java Generic UML Tool JAR Importer

This page last changed on Jul 22, 2008 by ghostar.

1. Scope

1.1 Overview

The UML Tool currently has no way of importing data into the tool an external source. This component will allow for the import of Java class data into the tool, into an existing diagram. This will enable users to write test Java code and just import it into the diagram. This component will import the data directly into the UML Tool project model.

1.1.1 Version

1.0

1.2 Logic Requirements

1.2.1 Input

The main input to the component will be a path to a JAR, and the UMLModelManager instance that represents the model manager for the current UML Tool project. The component must support the path specification as both a string path to a local file, as well as a URL to a remote file.

1.2.2 JAR Processing

The given JAR will be processed to pull all interface, class, operation, and attribute information, as well as the corresponding visibility, final, static, and abstract modifiers into the project model housed in the UMLModelManager instance provided.

1.2.3 Diagram selection

The component API must support importing into a provided class diagram, specified by diagram name. The diagram can be selected from the UMLModelManager instance provided.

1.2.4 Import associations

When importing, care must be taken to properly associate elements with each other. This means that if a class defines an attribute or operation parameter of a type that is also defined in the JAR, the association should be to a single instance of the associated type, instead of having redundant model elements. For instance, if we have a class named "Type1", and class "Type2" has an attribute of type "Type1" and two operation parameters of "Type1", "Type1" should only be placed into the model once, and the associations in "Type2" should reference that single model element.

1.2.5 External entities

When importing, if a class extends from an external class, if an interface extends from another interface, or if a class implements an external interface, the relationships should be added as well. The external class or element should be added to the diagram, and should be marked "External", by setting the "isExternal" property of the class node to true. For example, if a class implements the Serializable interface, the interface should be shown on the digram, with the proper realization relation defined from the class to the interface.

1.2.6 Inner classes

Inner classes should be imported, with a stereotype of "inner". A dependency relationship should also be added from the parent class to its inner class.

1.3 Required Algorithms

How the JAR is processed and the model is created needs to be described in detail.

1.4 Example of the Software Usage

This component will provide an easy way for the user to import code into the UML tool, speeding up design for users.

1.5 Future Component Direction

Better support for various Java properties will be added in the future, for things like attributes applied to classes and operations to be shown as stereotypes, as well as direct Java source input, and possibly documentation as well.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

None

2.1.3 Environment Requirements

- Development language: Java 1.5
- Compile target: Java 1.5

2.1.4 Package Structure

com.topcoder.uml.importer.jarimporter

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

- JAR

3.2.2 TopCoder Software Component Dependencies:

- UML Model Manager http://software.topcoder.com/catalog/c_component.jsp?comp=24458909
- UML Model Management http://software.topcoder.com/catalog/c_component.jsp?comp=24249402
- UML Model Core Relationships http://software.topcoder.com/catalog/c_component.jsp?comp=24211040
- UML Model Core Classifiers http://software.topcoder.com/catalog/c_component.jsp?comp=24210768

3.2.3 Third Party Component, Library, or Product Dependencies:

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of the TopCoder UML Tool.