Class Associations 1.0 Requirements Specification

1. Scope

1.1 Overview

The Class Associations component provides a way of associating a particular object (the handler) with a specific class or with a class and that class' subtypes (the target). This can be used to associate a formatting object, for example, with the type or types of objects that it can format.

1.2 Logic Requirements

The component must associate objects with either a class or a class hierarchy (a class' subtypes), as specified by the client. The client will specify these associations programmatically, through an API.

1.3 Required Algorithms

When looking up an association for a particular target class, the following algorithm should be used:

- Find handler objects whose associated class is the type of the target, and return it if present.
- If there is no association with the actual type of the target, search for associations (that can handle subtypes) that are associated with supertypes of the target.
- Of these associations, search for one that is associated with a class that is a subtype of all others.

In this way, various handlers can be associated with classes that are related through inheritance or implementation, and given a particular class, the 'closest' handler will be returned.

An example of the algorithm goes as follows:

- Three classes are used as targets: Shape, Rectangle and Square. Square is a subtype of Rectangle, and Rectangle is a subtype of Shape.
- Two classes are used as handlers: ShapeDrawer and RectangleDrawer.
- Two calls are made to the Class Association component to associate the Shape type with an instance of ShapeDrawer, and to associate the Rectangle type with an instance of RectangleDrawer. Both calls indicate that these Drawer instances can handle subtypes of their associated classes.
- A call is made to find a handler for a Square.
- Since no class is directly associated with a Square, we search the associations that are present to find if any of them handle supertypes of Square. As both of the handlers are associated with supertypes of Square, they are both potential handlers.
- Examining the two associations reveals that of the two, Rectangle is closer to a Square than Shape is, because Rectangle is a subtype of Shape. The instance of RectangleHandler that is associated with the Rectangle class and it's subtypes is returned.

1.4 Example of the Software Usage

This component could be used in both the Chart Renderer and Object Formatter, both of which have similar functionality.

1.5 Future Component Direction

It may be useful to add other association algorithms.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.3, Java1.4

2.1.4 Package Structure

com.topcoder.util.classassociations

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?

No.

3.2.2 TopCoder Software Component Dependencies:

None.

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

None.

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000

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
- Test Plan

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.
- Implementations of the design must clearly translate the design documentation into the corresponding language format (i.e. java -> javadoc, .NET -> XML). Source code documentation must be sufficient to enable component usage without additional documentation.

