Type Safe Enum 1.1 Requirements Specification

1. Required Changes

1.1 Add Constructor Enum(Class className)

A new constructor should be added to the Enum class to accept a Class instance to be used as the key into the enumsByClass map. This will allow subclasses to use internal sub-classing to differentiate behavior based on enumeration type while maintaining proper ordinal counts.

1.1.1 Update Existing Constructor to Accept Class Argument

Change the existing constructor to accept a Class className argument. Use this as the key into the Map instead of this.getClass()

1.1.2 Add New Default Constructor to Use this.getClass() by Default

Add a new default constructor that calls the other constructor with this(this.getClass()).

1.1.3 Update Documentation

Update documentation to reflect the new constructor.

1.1.4 Update Tests as Needed

Update test code to test the new API as needed.

1.2 Remove Use of ‘enum’ Keyword

The Type Safe Enum component currently uses the ‘enum’ keyword in its tests. These variable names should be changed to use a non-reserved name.

1.3 Verify Compilation and Execution in Both Java 1.4 and 1.5

The component should be compiled and tested in both Java 1.4 and 1.5 to verify the changes were successful.

1.4 Update/Redo All Documentation for Current TopCoder Standards

The Type Safe Enum component was designed and developed some time ago, so the documentation and diagrams are not up to the current TopCoder standards. All of the documentation and Poseidon files and diagrams should be updated to meet the current standards. This should be fairly simple as the component only contains one class.

1.5 Maintain 100% Backwards Compatibility

The above changes are all that is required. The component must maintain backwards compatibility.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

Not applicable.

2.1.2 External Interfaces

Not applicable.

2.1.3 Environment Requirements

- o Development language: Java 1.4, Java 5.0
- o Compile target: Java 1.4, Java 5.0

2.1.4 Package Structure

com.topcoder.util.collection.typesafeenum

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

Not applicable.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None required.

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:

Not applicable.

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 Poseidon.