Requirements Specification

1. Scope

1.1 Overview

The rounding component provides numerous rounding techniques through a standard interface. A strategy pattern is used to load the proper rounding algorithm. Rounding algorithms included in this release are Banker's Rounding, Random rounding and standard rounding. Additional algorithms are easily added to this component by implementing a specific interface.

1.2 Logic Requirements

1.2.1 Precision

- Provide a function to set the number of digits to the right of the decimal point for accuracy. The default will be 2.
- Comparison digit – default will be 5 but may be set to another value. This will be used for rounding.

1.2.2 Rounding strategy

- No Rounding
  - Ignore accuracy digit and return the whole number.
  - Example 3.4563 returns 3.4563
- Always Round Up (Symmetric)
  - Truncate everything after the accuracy digit and add one to the accuracy digit.
  - Example 3.21234 returns 4 if the accuracy is 0.
  - Example 3.21234 returns 3.22 if the accuracy is 2.
  - Example -2.61234 returns -2 if the accuracy is 0.
  - Example -2.61234 returns -2.61 if the accuracy is 2.
- Always Round Up (Asymmetric)
  - Truncate everything after the accuracy digit and add one to the accuracy digit
  - Example 3.21234 returns 4 if the accuracy is 0.
  - Example 3.21234 returns 3.22 if the accuracy is 2.
  - Example -2.61234 returns -3 if the accuracy is 0.
  - Example -2.61234 returns -2.62 if the accuracy is 2.
- Always Round Down (Symmetric) – All numbers move to zero
  - Truncate everything after the accuracy digit
  - Example 3.21234 returns 3 if the accuracy is 0.
  - Example 3.21234 returns 3.21 if the accuracy is 2.
  - Example -2.61234 returns -2 if the accuracy is 0.
  - Example -2.61834 returns -2.61 if the accuracy is 2.
- Always Round Down (Asymmetric) – All numbers round down
  - Truncate everything after the accuracy digit
  - Example 3.21234 returns 3 if the accuracy is 0.
  - Example 3.21234 returns 3.21 if the accuracy is 2.
  - Example -2.61234 returns -3 if the accuracy is 0.
  - Example -2.61834 returns -2.62 if the accuracy is 2.
- Symmetric Rounding
  - Round anything > than the comparison digit up
  - Round anything < than the comparison digit down
  - Round anything = to the comparison digit down
  - Example -.5 is rounded to -1
- Asymmetric Rounding
  - Round anything > than the comparison digit up

    - Round anything < than the comparison digit down
    - Round anything = to the comparison digit up
    - Example -.5 is rounded to 0
- Banker's Rounding
    - Round anything > than the comparison digit up
    - Round anything < than the comparison digit down
    - Round anything = to the comparison to the nearest even number
    - Zero is considered an even number
- Random Rounding
    - Round anything > than the comparison digit up
    - Round anything < than the comparison digit down
    - If the value is equal to the comparison digit
        - Randomly chose to round up or down
- Alternate Rounding
    - Round anything > than the comparison digit up
    - Round anything < than the comparison digit down
    - If the value is equal to the comparison digit
        - Alternate rounding up and rounding down.
        - Must maintain state for this particular strategy.

1.3 Required Algorithms

Define all rounding algorithms.

1.4 Example of the Software Usage

The rounding component will be used in the currency component. Depending on the situation different rounding techniques maybe required.

1.5 Future Component Direction

None identified.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None identified.

2.1.2 External Interfaces

None identified.

2.1.3 Environment Requirements

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

2.1.4 Package Structure

com.topcoder.math.roundingfactory

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?

None identified.

3.2.2 TopCoder Software Component Dependencies:

None identified.

**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 identified.

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.