Component Specification

1.Design

The Object Formatter will format objects with the best-fitting format method. That is, it looks through its association list for a type that most closely matches the run-time type of the object to be formatted. If it can't find a matching type, or if there is more than one matching type, then an exception is thrown.

There are two levels of formatters. The first is the format method level, where a class formats one specific object type. The interfaces corresponding to this level are DateFormatMethod and ObjectFormatMethod. Default instances of these interfaces can be created using the methods in FormatMethodFactory.

The second level of formatters are the PrimitiveFormatter and ObjectFormatter interfaces. Conceptually, these formatters take what they're given and decide the best way to format it. PrimitiveFormatters format each primitive type, and ObjectFormatters format Objects. Default instances of the formatters can be created using the methods in PrimitiveFormatterFactory and ObjectFormatterFactory.

The default primitive and date formatters know about locale info. It seemed easiest not to reinvent the wheel, but rather to have the number and Date format string formats be exactly the same as for java.text.DecimalFormat and java.text.SimpleDateFormat respectively.

If an Object obj could be formatted by more than one format method, then the following algorithm will be used to determine the format method to use, if any. The format method whose associated type is closest to obj's type will be used. This closest type is determined by first determining all format methods (a) whose associated type is a super-type of obj (that is, it is an interface that obj implements, or it is a superclass of obj), and (b) that actually could format obj (because either the type matches exactly, or because the type is a super-type and is allowed to format sub-types). This set of types type$_1$, type$_2$, ... type$_n$ is then scanned to determine if there is any type$_i$ that is a sub-type of all the others. If so, then the format method associated with type$_i$ is used to format obj; if not, an exception is thrown. Note that this formatting algorithm ensures that if there is a format method associated with obj's exact run-time type, then that format method will be used to format obj.

As an example, suppose an ObjectFormatter has format methods associated with Object, Collection, List, and RandomAccess types, and suppose that all of these format methods are allowed to format sub-types. If a LinkedList were to be formatted, then the format method for Lists would be used, because Object, Collection, and List are super-types of LinkedList, and List is a sub-type of the other two. If, however, an ArrayList were to be formatted, then the ObjectFormatter would throw an exception, because List and RandomAccess are both super-types if ArrayList, but neither is a sub-type of the other.

Some methods in the interfaces and factory classes have reasonable default values to use when passed null references; when none such exist, or when an argument is non-sensical, an IllegalArgumentException is thrown.

The code skeleton and javadocs are the first and best source for info about the design.

1.1 Reference any design patterns used

Abstract factory, strategy.

1.2 Reference any standards used in the design

None.

1.3 Explain any required algorithms for the implementation (provide pseudo code)

None.

1.4 Component Exception Definitions

2. Environment Requirements

2.1 TopCoder Software Components:

None.

2.2Third Party Components:

None.

3. Installation and Configuration

3.1 Package Name

Com.topcoder.util.format

3.2 Configuration Parameters

None.

3.3 Dependencies Configuration

None.

4. Usage Notes

None.

4.1 Required steps to use the component

(These are actually developer hints.)

For formatting numbers, just create a java.text.DecimalFormat and pass everything through to it. For formatting Dates, just create a java.text.SimpleDateFormat object and pass everything through to it.

Be sure to make the factory classes have private empty constructors (or constructors that throw UnsupportedOperationException) to prevent those classes from being instantiated.

4.2Demo

None.