1 Implementation: Specialized Domain Objects

GeoAPI package:	org.opengis.coverage.core
GeoTools package:	org.geotools.coverage.core
Implementor:	*Your Name Here (change text style to "Table Contents" when entered.)*
Specified Implementation Classes:	• DomainPoint • DomainCurve • DomainSurface • DomainSolid • PointValuePair • GridPointValuePair • CurveValuePair • SurfaceValuePair • SolidValuePair
Specified Interface Classes:	• *all specified implementation classes...*

1.1 Problem definition

In the ISO19123 Primer, some incorrect UML was noted in the specialized children of GeometryValuePair. The precise manner in which the UML was incorrect may be summarized as follows: certain attributes of GeometryValuePair were overridden with unrelated types in the children. In itself, this is not illegal. In Java (and probably in other languages, too) this is handled in such a manner as to cause an insidious problem. In actuality, there are three problems: one is conceptual, as the specification is either in error or it does not meet my needs; one inspired by the Java language specification itself; and the last caused by the fact that in the GeoAPI/GeoTools environment, good object oriented design requires that attributes be represented by setter and getter methods rather than as publicly accessible fields.

The first problem is conceptual. If the ISO 19123 specification were to be accepted as it is published, GeometryValuePair.geometry would refer to a DomainObject (which maintains separate references for the spatial and temporal components.) However, all the specialized children of GeometryValuePair redefine the geometry attribute to be a purely spatial entity. If accepted at face value, this would mean that all coverages which use children of GeometryValuePair in their definition are restricted to be purely spatial. Such a radical departure from the generic nature of a Coverage, if it were intentional, should be explicitly stated in the specification. Obviously, this is a subjective judgment on my part. However I have no wish to restrict the child classes in such a manner, and I have an immediate need for spatio-temporal DiscreteGridPointCoverages, so this implicit restriction, as well as the definition which imposes it, shall be considered an error in want of correction.

The second problem, the Java language specification itself, applies to the treatment of inherited fields which are

overridden. The term for this action is "field hiding", and may apply to class variables or instance variables. (See: http://java.sun.com/docs/books/jls/third_edition/html/classes.html#8.3.3.1 for an example) In short, there are two unrelated fields in the resulting child class, one corresponding to the parent's definition, and one corresponding to the child's definition. Setting one does not set the other, and the parent has no knowledge of the child's version of the field. Probably the easiest way to think about the problem is that the child's definition behaves exactly like any field definition in a child class would be expected to behave. The fact that the field shares the same name as a field in the parent is completely coincidental. This problem is insidious because of the "coincidental" naming: code in the child class which appears to be setting a field in the parent is in fact setting the unrelated field in the child, and the parent's field remains unset. When the object is later passed to something which treats it as if it were an instance of the parent class, the parent's unset field is retrieved. Implementors are to be wary of this second problem whenever an attribute in the parent is redefined in the child.

The third and last problem is the representation of attributes by setter and getter methods in the GeoAPI/GeoTools environment. With this technique, attributes in the child which are overridden require that the methods to access the parent's attributes be overridden by methods to access the child's attributes. As the attribute has the same name in both parent and child, this directly leads to the declaration of a method in the child with the same signature as a method in the parent, except for the return value. While Java 1.5 will permit this, as long as the child's return value is a child of the parent's return value, it will not permit the child to return a totally unrelated object. This practice is sometimes known as "type narrowing," although the Java Language Specification calls this practice "covariant returns." (See: http://java.sun.com/docs/books/jls/third_edition/html/classes.html#8.4.5) Java 1.4 will not even permit type narrowing: when a child aims to override a parent's method, the return method must match exactly.

Solutions to these problems take two forms, tactical and strategic. Tactically, a method of implementing type narrowing must be realized in order to generate GeoAPI interfaces and GeoTools classes. This tactical solution addresses a common design pattern in the specification. Strategically, the specification itself must be reformulated such that the implicit restriction on these derived child types is removed.

As the objective is to maintain compatibility with Java 1.4, the tactical solution to the third problem is that child classes must maintain separate accessor methods for the specialized types. Additionally, the parent's setter method must be overridden to ensure that the provided type abides by the restrictions imposed by the child. A careful solution to this tactical problem also resolves the insidious issues surrounding field hiding. Strategically, the specification in ISO 19123 must be revised in order to relax the implicit restriction on the use of specialized children of GeometryValuePair. This redefinition of ISO 19123 will not eliminate type narrowing, which is clearly the intent of the standard, but will specify type narrowing in such a manner that the temporal component of the domain object is retained.

1.2 Departure from ISO 19123

To address the conceptual problem pointed out in the previous section, the ISO 19123 specification must be revised. The objects of this revision are the children of GeometryValuePair. These children are intended to narrow the types of acceptable geometries to some specific subclass of GM_Object. The goal of this revision is to retain this type-narrowing function in a manner which permits the separate specification of a temporal element. The proposed revision is presented in Figure 1.

The proposal in Figure 1 includes four new classes in yellow. All of the children of CV_GeometryValuePair,

specified in green, have different types for the geometry attribute than what is specified in ISO 19123, with the exception of CV_GridPointValuePair.

CV_GridPointValuePair, as well as CV_GridPoint, are exceptions to the general discussion in that no changes were required to provide for the desired functionality. CV_GridPoint is also an exception because most of the other children of CV_DomainObject in the diagram exist only to narrow the type of the SpatialComposition association. CV_GridPoint serves this function as well as many other functions, all of which are defined in the Quadrilateral Grid package. For this reason, implementation of CV_GridPointValuePair depends on the definition of CV_GridPoint, which is part of a set of foundation classes in the quadrilateral grid package.

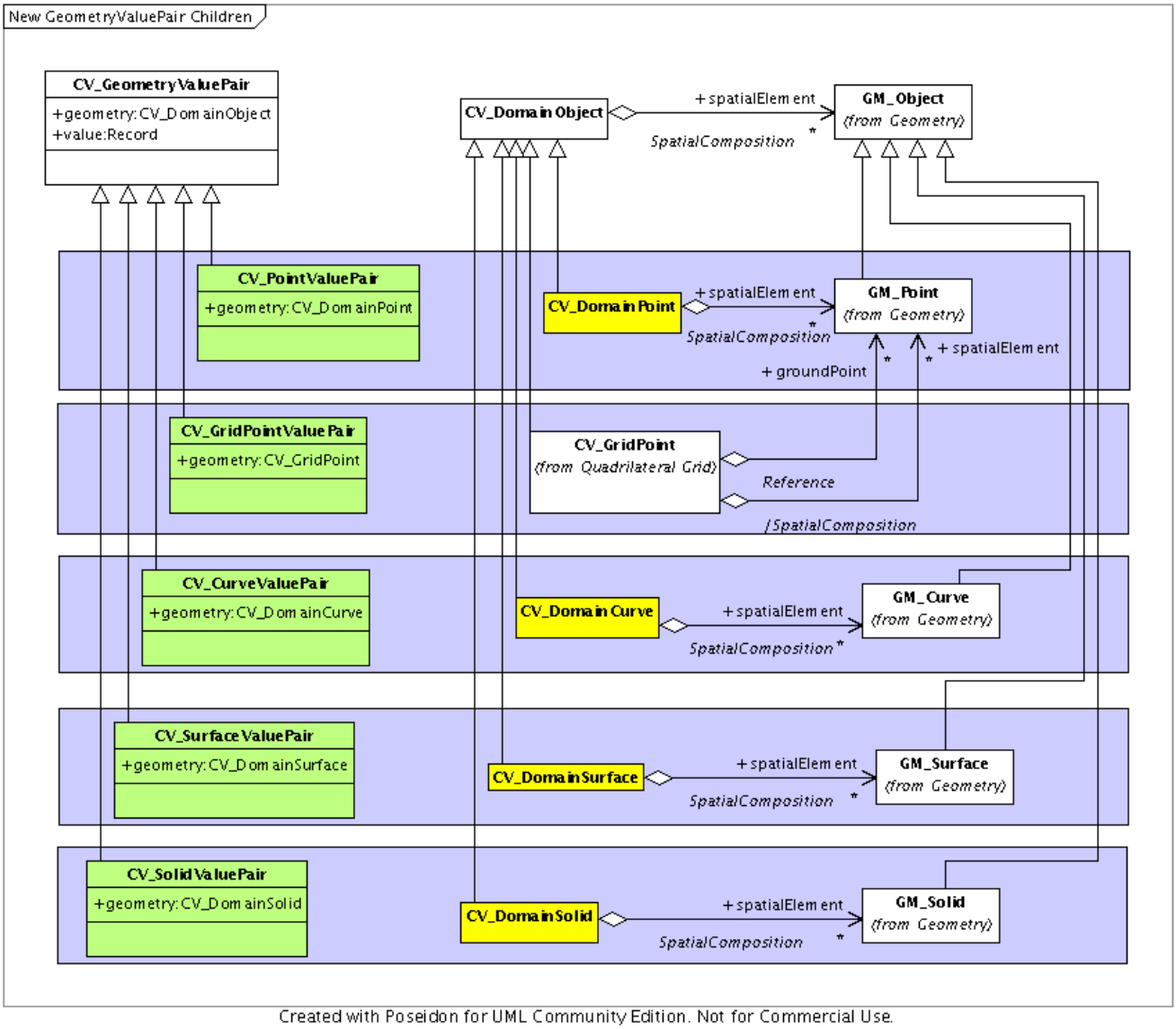

Figure 1: Proposed revision to ISO 19123 which allows children of CV_GeometryValuePair to narrow the type of spatial element while preserving the ability to specify temporal elements. Yellow classes do not exist in the specification and green classes have a different type for the geometry attribute.

1.3 GeoAPI Interface Diagram

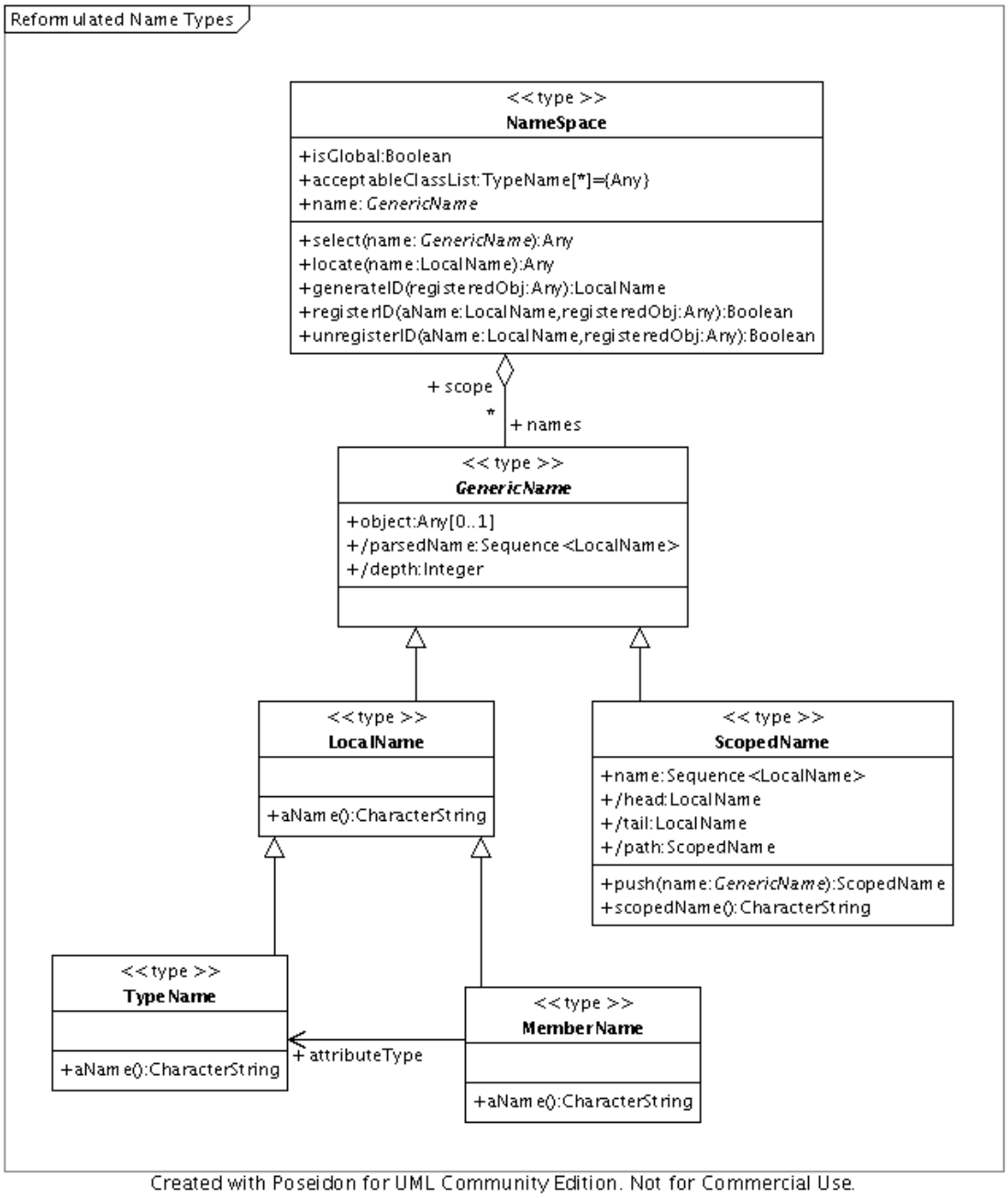

Figure 2: GeoAPI Interfaces for Specialized Domain Objects.

1.4 Implementation Design

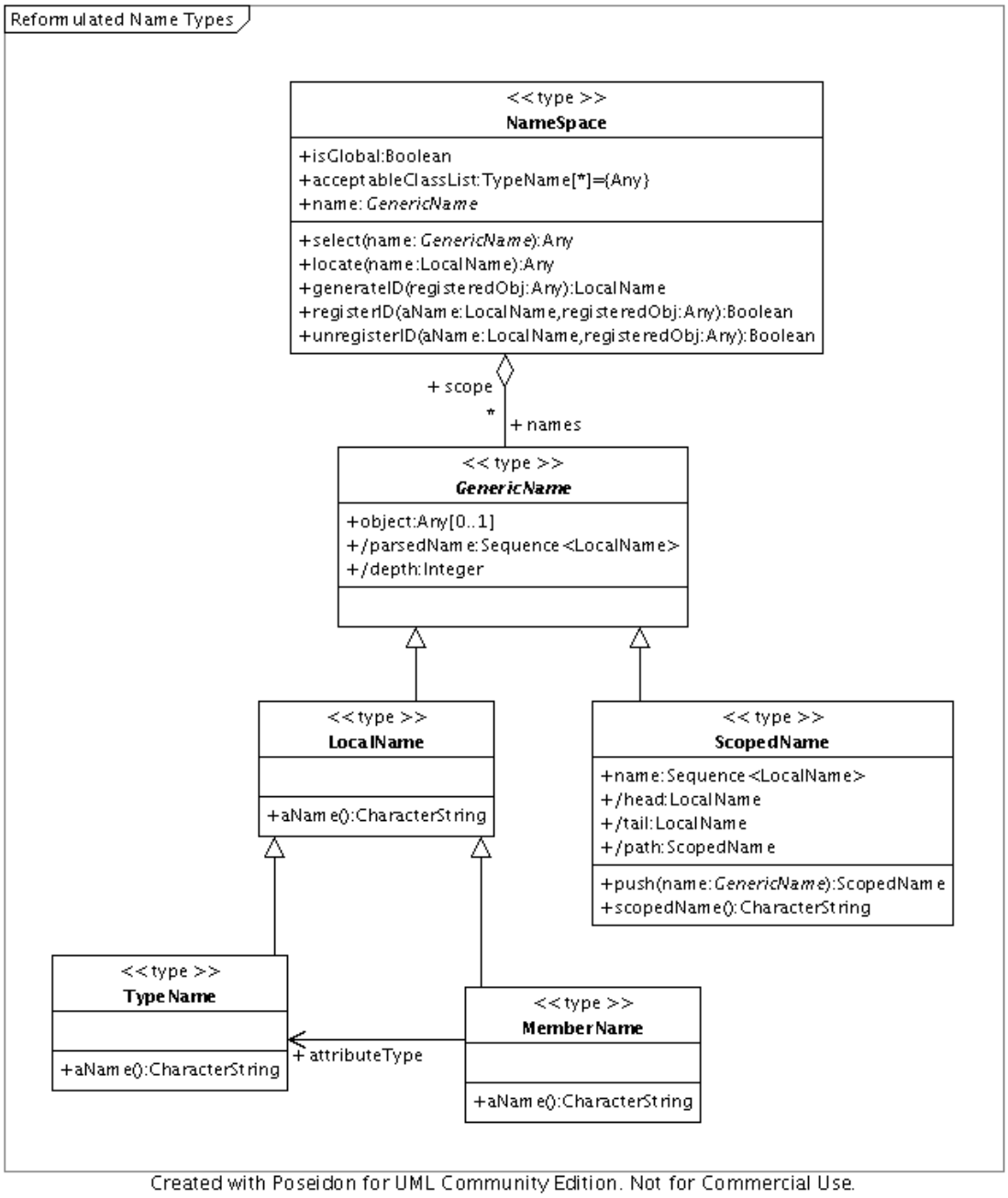

Figure 3: Implementation design for Specialized Domain Objects.

1.5 Detailed Discussion

1.6 Modeling Pre-work

1.6.1 Classification of GeoAPI methods

TODO: Use the following spreadsheet object to classify the GeoAPI methods. Double-click the spreadsheet (and not the frame) to edit..

GeoAPI Package: org.opengis.xxx

GeoAPI Class/Interface:

Method	Classification (A, R, O)	Type

Table 1: GeoAPI Method Classification for Specialized Domain Objects.

1.6.2 Interface and Data Type Proxies

TODO: Use the following spreadsheet object to list the Interface and Data Type proxies required for this implementation effort.

GeoAPI Package:

Interface Proxies	Data Type Proxies

Table 2: Proxy types for Specialized Domain Objects

1.7 Test cases

TODO: Describe potential test cases and indicate whether they have been implemented...

Name	*Done?*	*Description*
	y/n	

Table 3: Potential Test Cases for Specialized Domain Objects