org.jax.mgi.dbs.mgd.loads Design Document

Author: Mike Walker

Created: December 9, 2004

Last Modified: December 9, 2004 13:47

1 Purpose of Document

This document describes the classes belonging to the org.jax.mgi.dbs.mgd.loads package from the lib_java_dla product and provides source code examples for common usage patterns.

2 Introduction

The org.jax.mgi.dbs.mgd.loads package is comprised of classes that are responsible for providing basic access to database tables within the mgd database. This includes classes which represent raw attributes for a database table and classes which resolve raw attributes to database values. Additionally, there are classes which repesent a combination of tables in mgd. These classes (for example, the **Sequence** class) manage multiple **DAO** classes (from lib_java_dbsmgd) for updating data across multiple tables.

This package consists of multiple subpackages. Each subpackage repesents a database object in mgd or multiple database objects. These subpackages are each addressed within seperate subsections from this document. Table 1 lists these subpackages and provides information on the tables within mgd that they manage.

This package integrates with the lib_java_core product and the lib_java_dbsmgd and lib_java_dbsrdr products

You must be pointing to Java1.4 in your classpath when using this product.

.

Table 1: subpackages

subpackage	tables represented from mgd
Acc	Acc_Accession
Coord	MAP_Coord_Collection, MAP_Coordinate, MAP_Coord_Feature
Seq	SEQ_Sequence, Acc_Accession, MGI_ReferenceAssoc, SEQ_SourceAssoc
SeqRefAssoc	MGI_Reference_Assoc
SeqSrc	PRB_Source

Table 1: subpackages

subpackage	tables represented from mgd
SeqSrcAssoc	SEQ_SourceAssoc

3 Data Access Design Patterns

This section describes a number of patterns developed to facilitate database access. Some of the classes represented in this package and documented in section 4 are based on these patterns.

3.1 CompoundDataObject

This represents a collection of resolved data values which can be inserted or updated into one or more database tables. This pattern allows a user to change the attributes and commit the changes to the database or to insert the data into tables for the first time.

3.2 RawAttributes

Raw provider attributes for an object represented in the database. It is modeled after the standard bean model with setter and getter methods provided for each attribute.

3.3 RawAttributesResolver

Resolves all raw attribute values to the required database values which can be a translated terms or a database keys. One or more Resolvers are used to carry out this task. A new object is created which contains the resolved information and which can be inserted or updated within the database. This object can be either a DAO object (see lib_java_core and lib_java_dbsmgd) or a CompoundDataObject (3.1).

3.4 Resolver

Any object which is designed to resolve an attribute or a group of attributes to a single term or database key.

3.5 Lookup

An object which lookups data within the database and creates a CompoundDataObject (3.1).

3.6 Processor

Uses all the above patterns to process raw provider input and commit data to the database.

4 Class Overview

4.1 Acc

The **AccessionRawAttributes** class represents the raw attributes for the ACC_Accession table. It provides setter and getter methods for these attributes. This class follows the RawAttributes pattern.

The **AccAttributeResolver** class takes an **AccessionRawAttributes** and resolves the attributes to database values. The resolved values are stored within a **SequenceState** class (part of the DAO classes from the lib_java_dbsmgd product). This class follows the RawAttributesResolver pattern.

4.2 Coord

TBD

4.3 Seq

The **Sequence** object represents data from the SEQ_Sequence table and other supporting tables as listed in table 1 above. This object allows one to change attributes within the class and then commit these changes to the database or to insert new data. This class follows the CompondDataObject pattern.

The **SequenceLookup** class provides a query for looking up data from the database and creating a **Sequence** object. This class follows the Lookup pattern.

The **SequenceRawAttributes** class represents the raw attributes obtained from an input file which are to be resolved into database values. This class follows the RawAttributes pattern.

The **SequenceAttributeResolver** class takes a **SequenceRawAttributes** object and resolves the attributes to column values for the SEQ_Sequence table. This class follows the RawAttributesResolver pattern.

The **SequenceResolverException** class represents an exception which may occur during the resolving of attributes.

The **SequenceUpdater** class encapsulates rules for updating SEQ_Sequence column values within the mgd database.

The **SequenceInputProcessor** class processes input from a provider file and loads new data into mgd. It uses all the above classes and encapsulates the involved processing steps. This class follows the Processor pattern.

The **IncremSequenceInputProcessor** class extends SequenceInputProcessor and provides additional functionality for updating existing data within mgd. This class follows the Processor pattern.

4.4 SeqRefAssoc

The **RefAssocRawAttributes** class represents raw values for the MGI_Reference_Assoc table which is obtained from provider input sources. This class follows the RawAttributes pattern.

The **RefAssocAttributeResolver** class takes a **RefAssocRawAttributes** object and resolves it's attributes to database values. This class follows the RawAttributesResolver pattern.

The **SeqRefAssocPair** class holds two **RefAssocRawAttributes** objects, one for Medline and another for Pubmed.

The **SeqRefAssocProcessor** uses all the above objects and encapsulates the steps required to process raw reference information into mgd tables. This class follows the Processor pattern.

4.5 SeqSrc

The **MolecularSource** class represents data from the PRB_Source table. This class follows the CompoundDataObject pattern.

The **MSRawAttributes** class represents raw data from the provider input sources for molecular source information. This class follows the RawAttributes pattern.

The **MSAttrResolver** class takes a **MSRawAttributes** object and resolves the attribute to mgd values. This class is abstract and is implemented by the **GenereicMSAttrResolver**, the **GBMSAttrResolver** and the **NonGBMSAttrResolver**. This class follows the RawAttributesResolver pattern.

The **GenericMSAttrResolver** extends **MSAttrResolver** to provide general functionality for resolving molecular source raw attributes. No special resolving rules are applied in this class. This class follows the RawAttributesResolver pattern.

The **GBMSAttrResolver** extends **MSAttrResolver** to provide special resolving rules for GenBank. This class follows the RawAttributesResolver pattern.

The **NonGBMSAttrResolver** extends **MSAttrResolver** to provide special resolviong rules for swissprot. This class follows the RawAttributesResolver pattern.

The **MSLookup** class is used for creating **MolecularSource** objects from a database query. This class follows the Lookup pattern.

The **MSResolver** class has a **MSAttrResolver**. It resolves attribute values and then locates a PRB_Source object from mgd represented by these attributes. The flavor of the **MSAttrResolver** used by the **MSResolver** is determined at runtime. This class follows the Resolver pattern.

The **MSCollapsedCache** is a special cache used for storing **MolecularSource** objects which have been resolved from provider raw attributes. This cache helps facilitate the reuse of **MolecularSource** objects that share similar attributes across multiple sequence objects.

The **MSException** class represents an exception which can occur during the use of any classes belonging to this package.

The **MSExceptionFactory** class stores named **MSException** instances.

The **MSProcessor** uses all the above classes and encapsulates the steps required to translate raw provided input to mgd data. This class follows the Processor pattern.

4.6 SeqSrcAssoc

The **MSSeqAssoc** class represents a row for the SEQ_Source_Assoc table. It extends **DAO** and therfore can be updated and inserted directly onto an **SQLStream** (see package org.jax.mgi.shr.dbutils.dao from the lib_java_core product)

5 Configuration

- **Acc**

None

- **Coord**

See configuration documentation associated with the CoordLoader from the org.jax.mgi.shr.dla.loader package.

- **Seq**

See configuration documentation associated with the CoordLoader from the org.jax.mgi.shr.dla.loader package.

- **SeqRefAssoc**

None

- **SeqSrc**

MS_OK_TO_SEARCH_ASSOC_CLONES

This setting will control whether to search clones associated with an incoming sequence in order to try and identify a name for a sequence source which otherwise would be anonymous. The default setting is true.

MS_USE_ASSOC_CLONES_FULL_CACHE

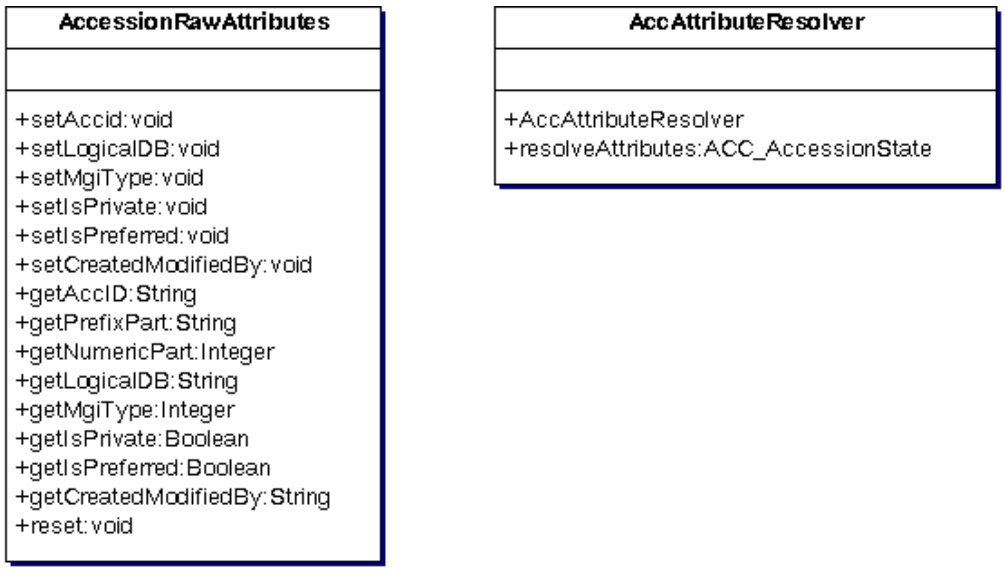
This setting will control whether or not to use a full cache when searching associated clones. By setting this to true, all clones from the database will be loaded into memory for searching purposes. The default setting is true.

- **SeqSrcAssoc**

None

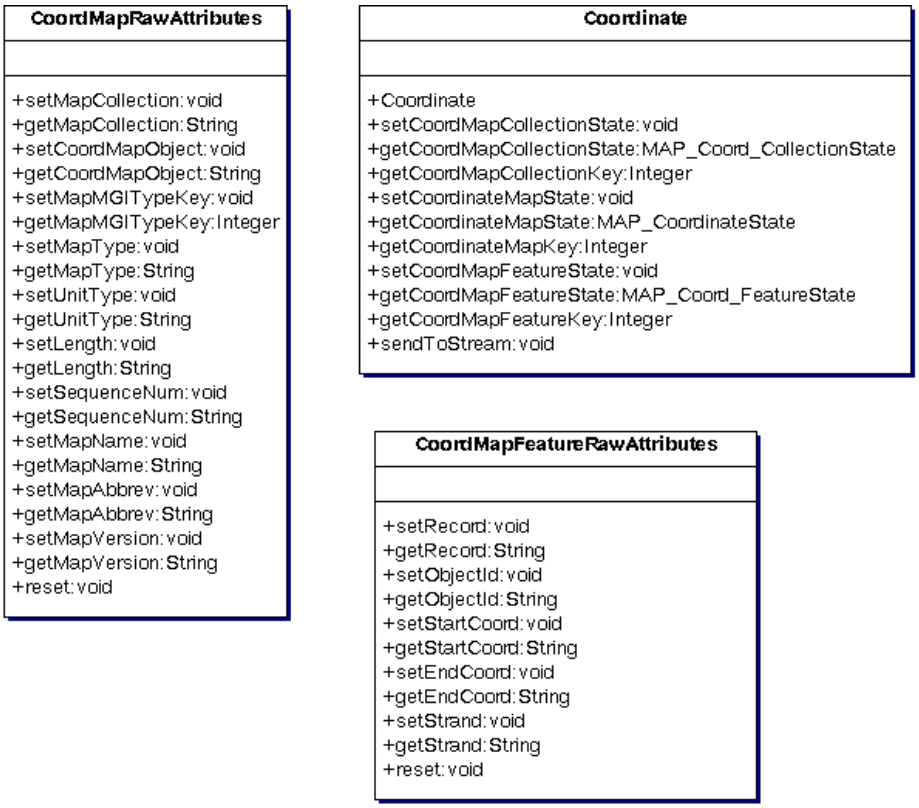
6 Class and Sequence Diagrams

6.1 Acc

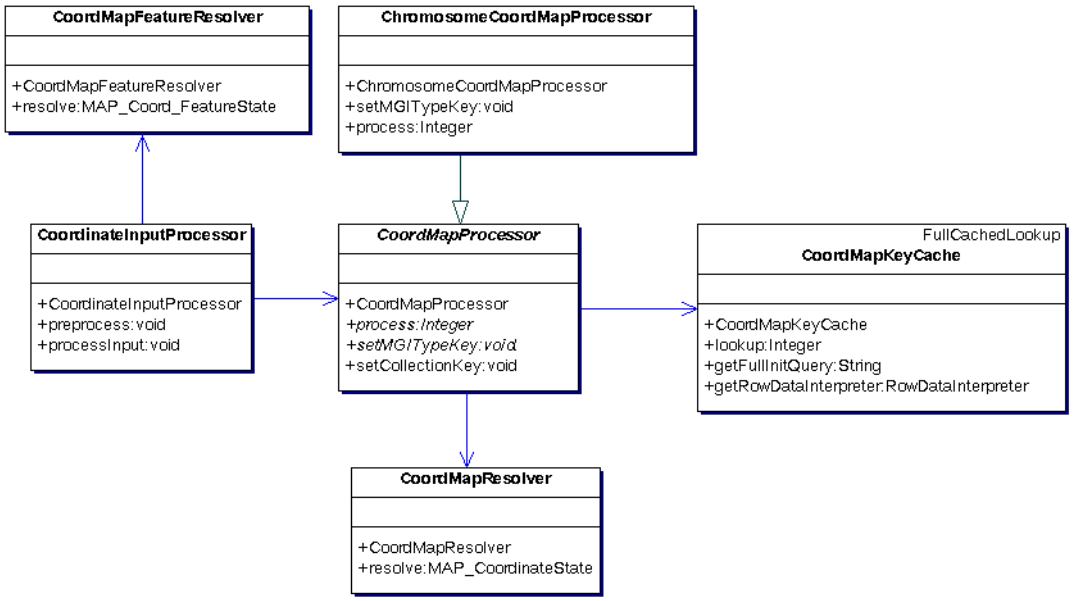


6.2 Coord

6.2.1 Coordinate and RawAttributes



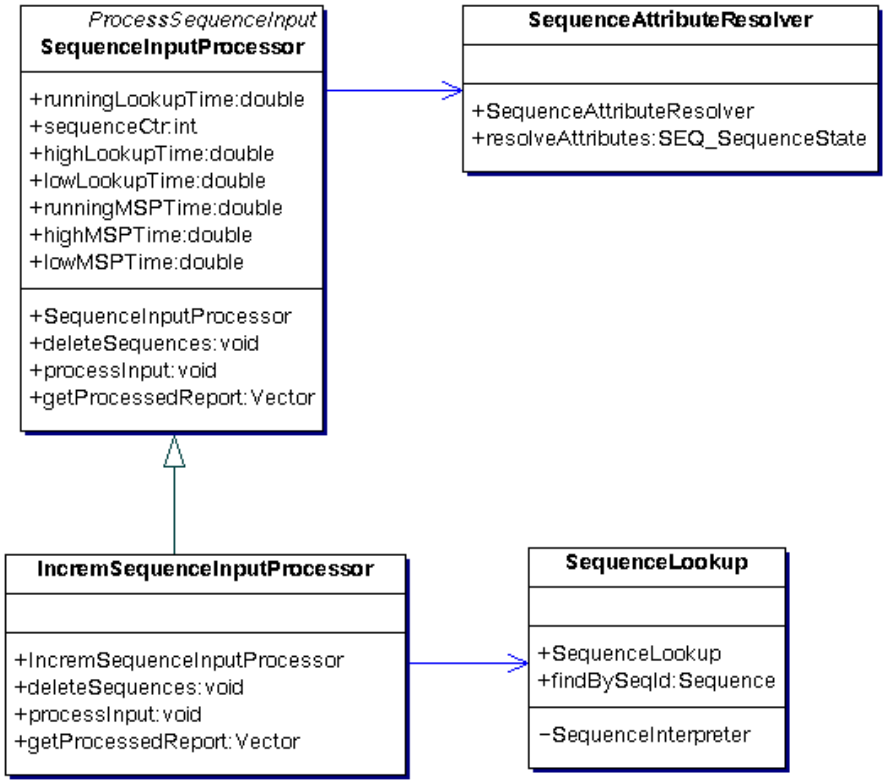
6.2.2 CoordMapProcessor



6.3 Seq

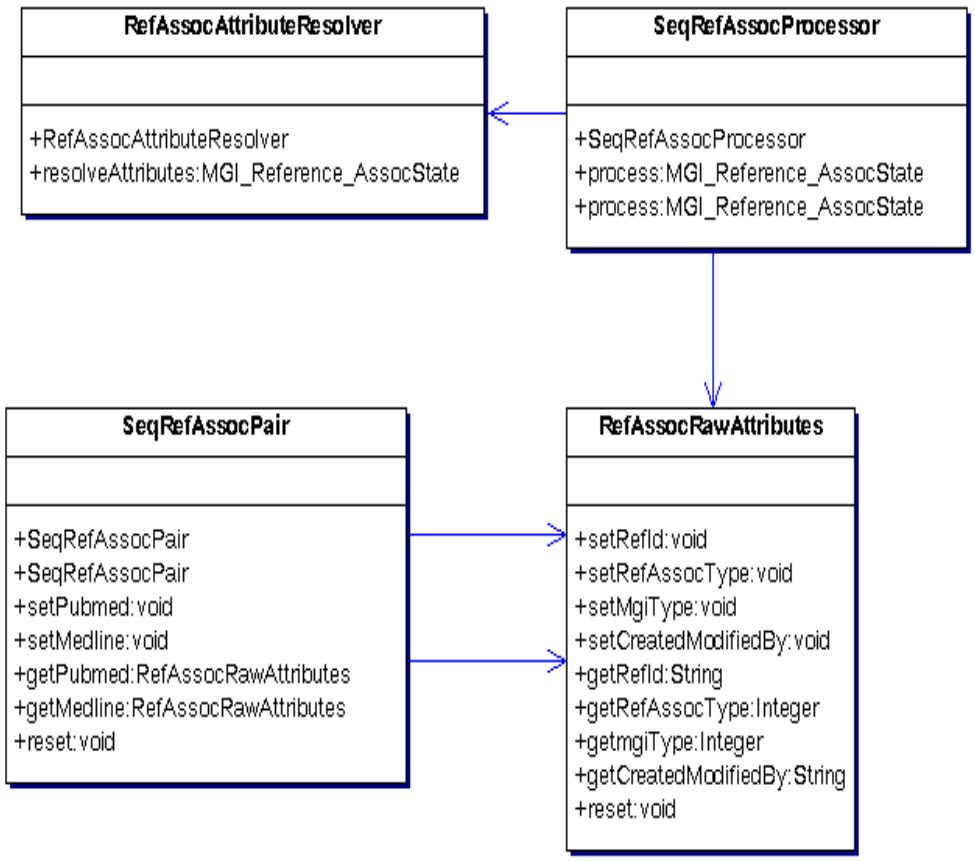
6.3.1 SequenceInput and RawAttributes

6.3.2 SequenceInputProcessor



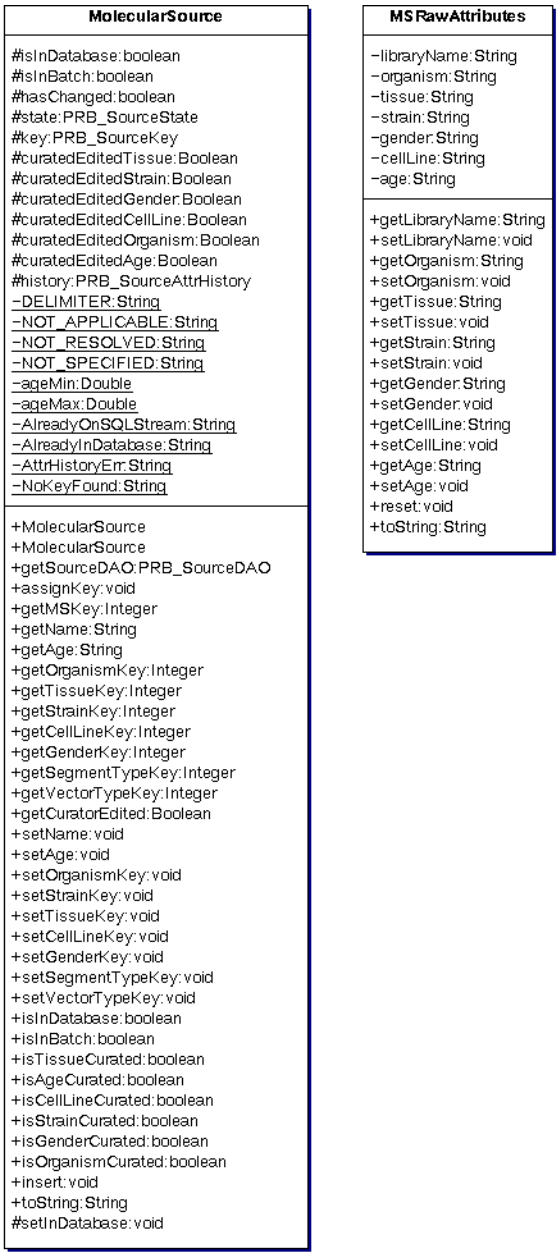
6.4 SeqRefAssoc

6.4.1 SeqRefAssocProcessor

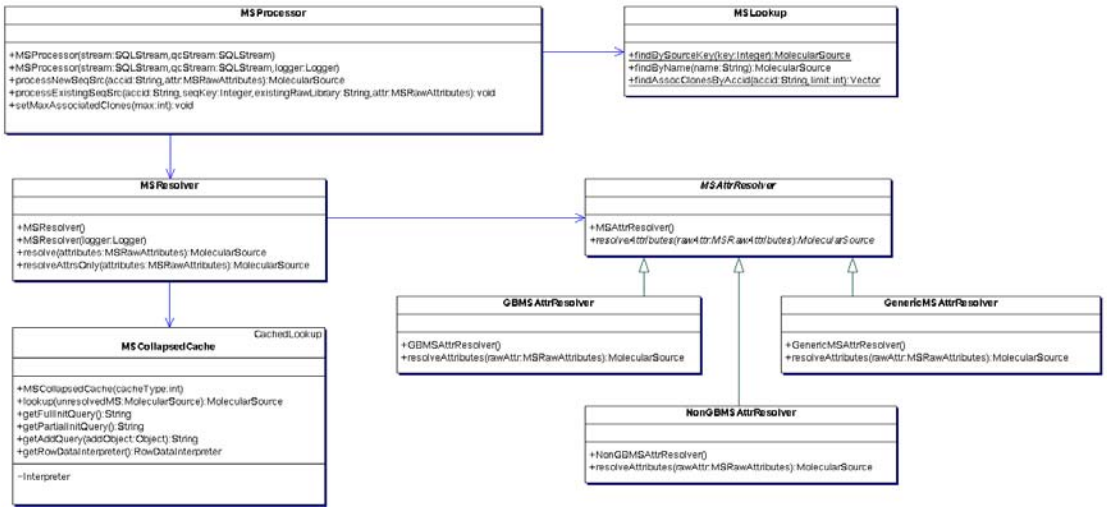


6.5 SeqSrc

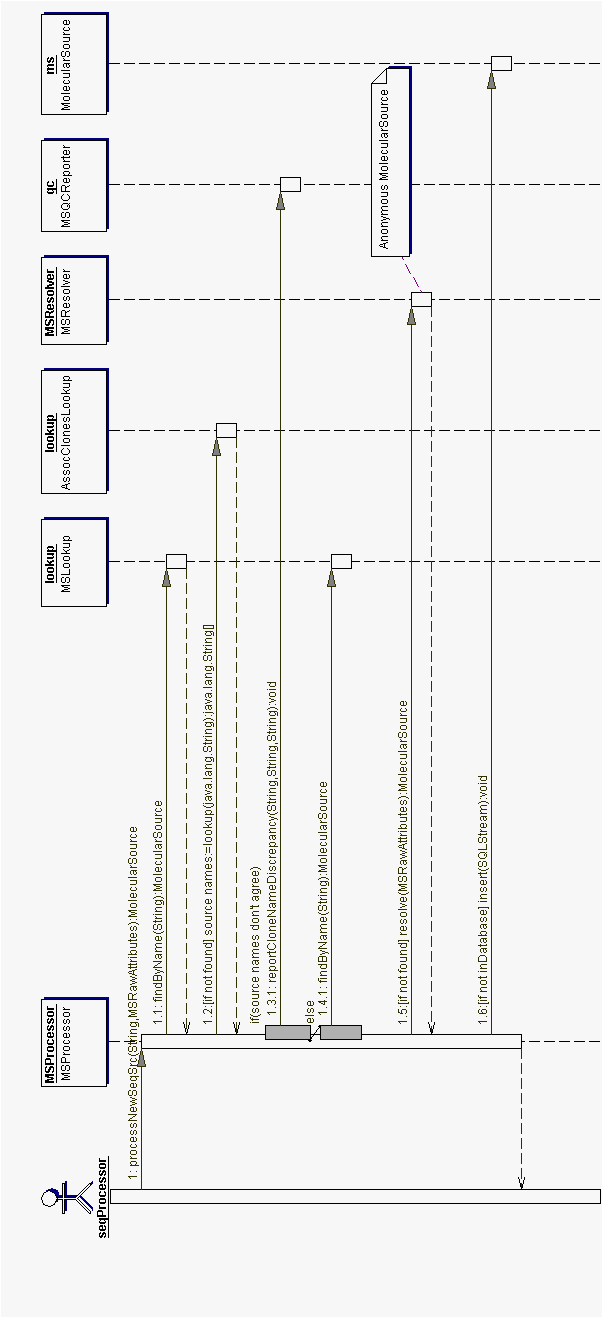
6.5.1 MolecularSource and MSRawAttributes



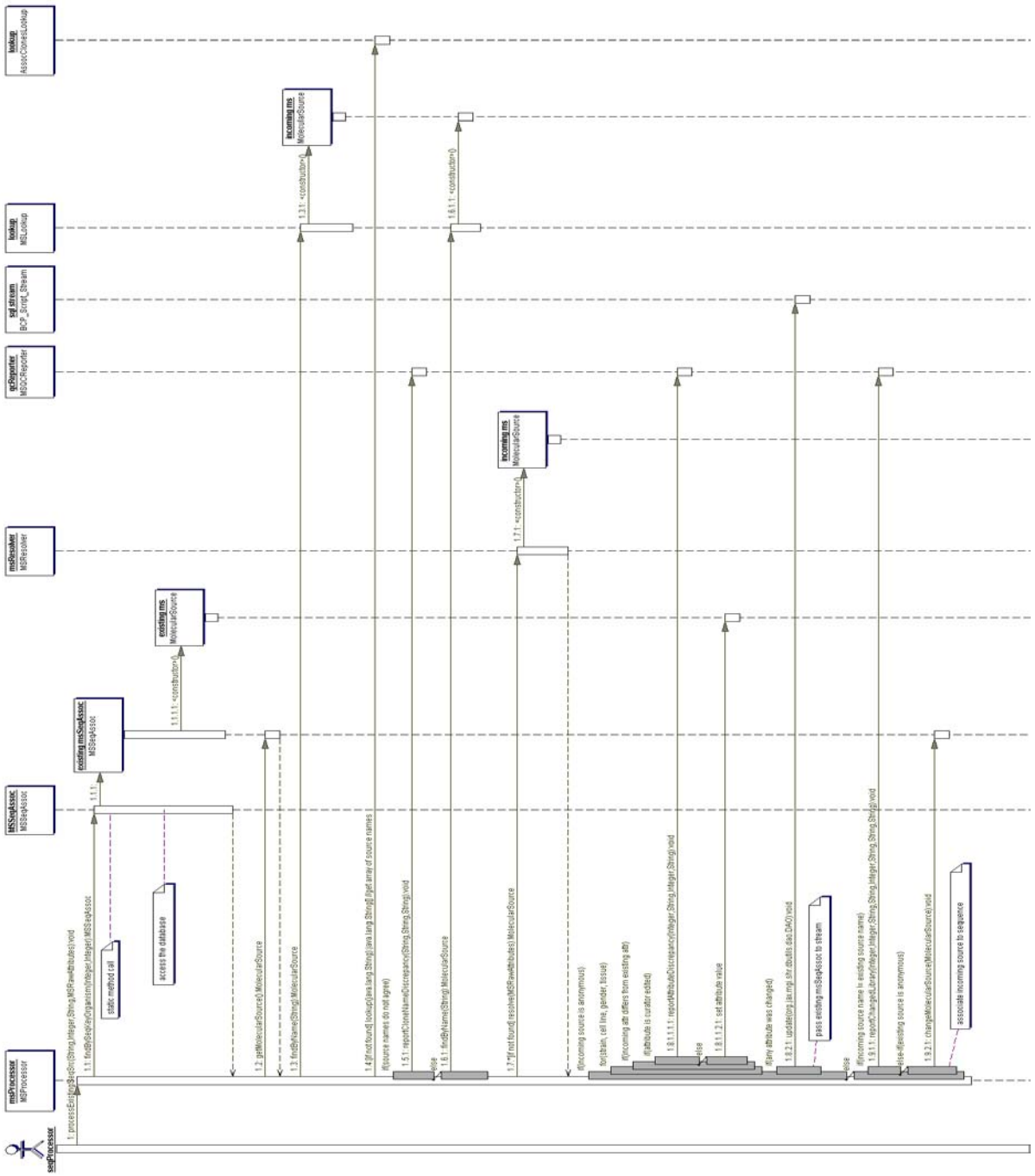
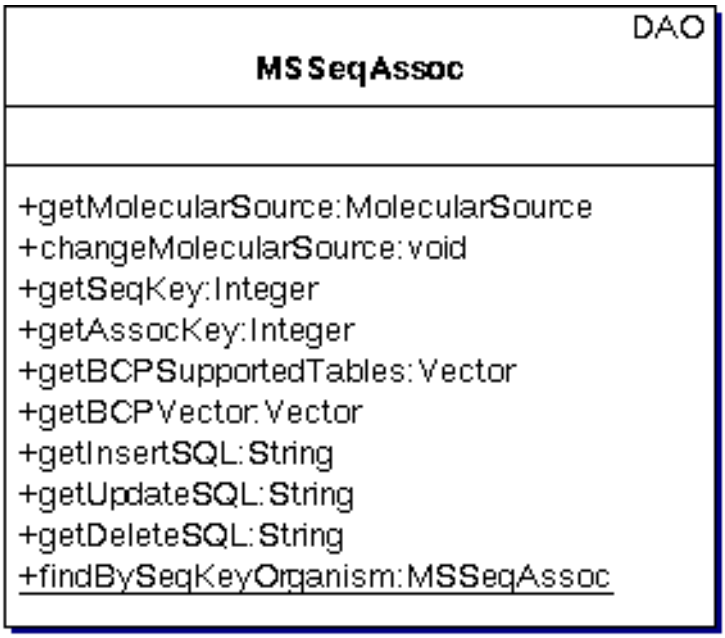
6.5.2 MSProcessor



6.5.3 MSProcessor: processNewSource()



6.5.4 MSProcessor: processExistingSource()



6.6 SeqSrcAssoc