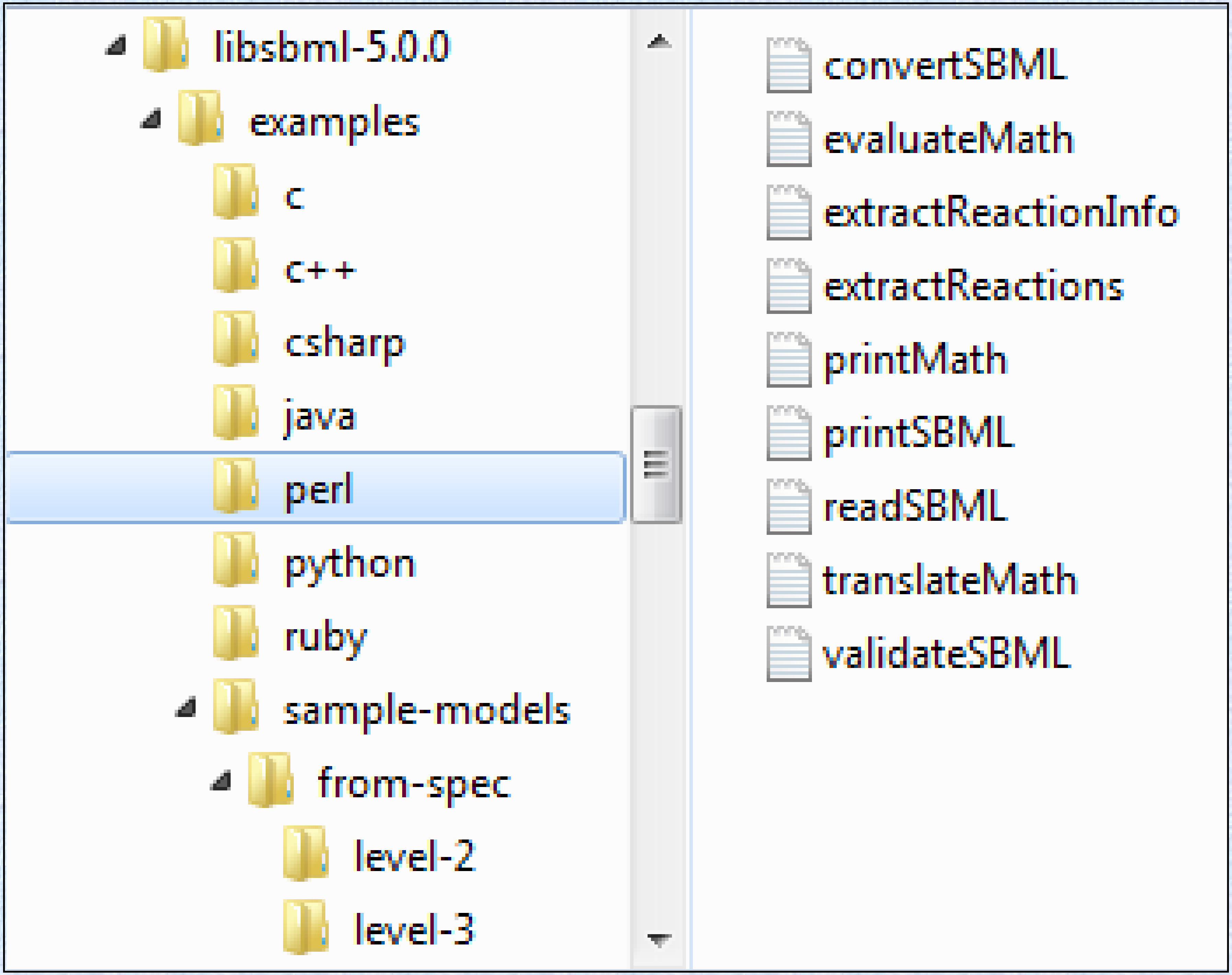
libSBML

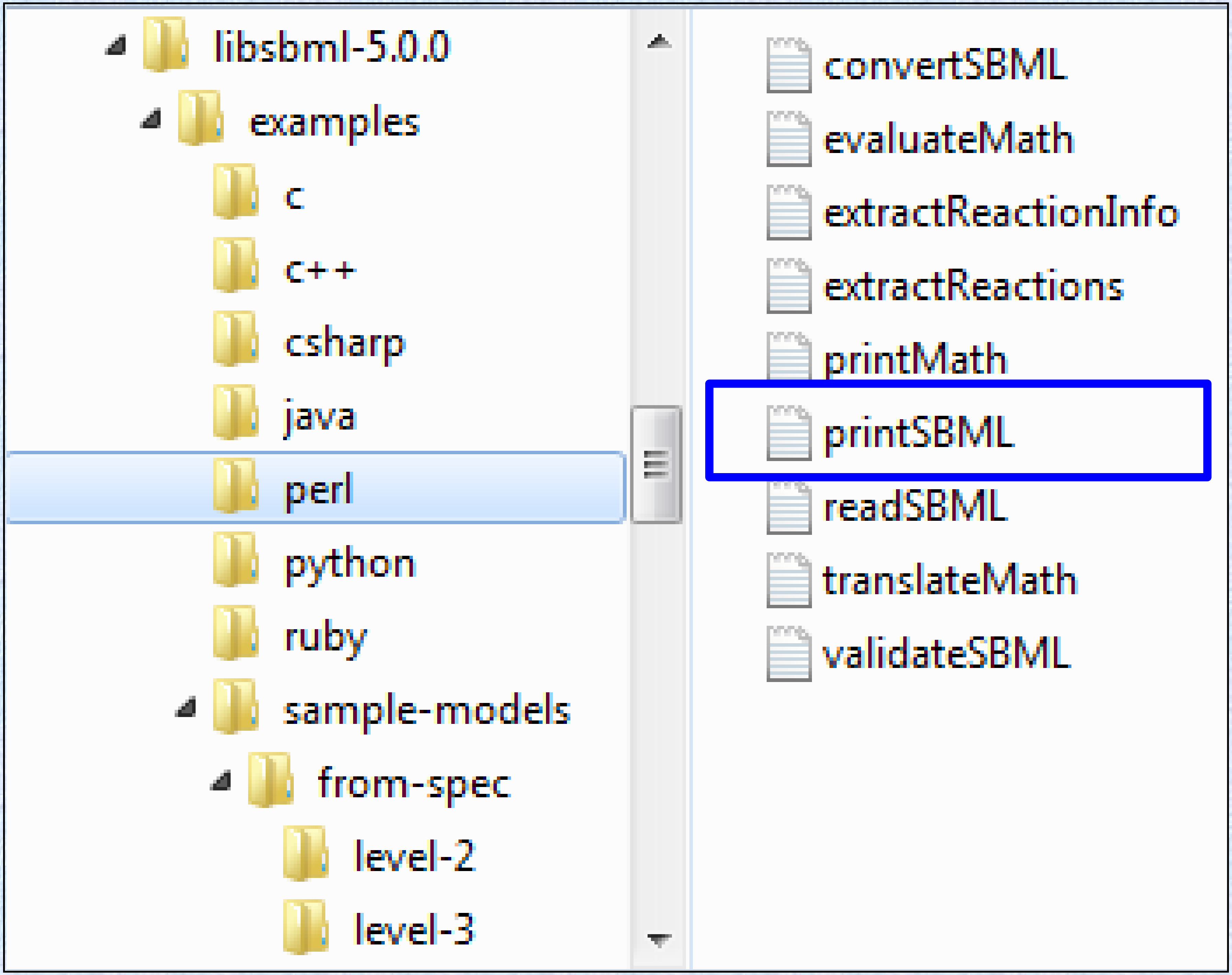
Frank Bergmann and Sarah Keating

Getting started ...



```
libsbml-5.0.0
  examples
    c
    c++
    csharp
    java
    perl
    python
    ruby
  sample-models
    from-spec
      level-2
      level-3
```

```
convertSBML
evaluateMath
extractReactionInfo
extractReactions
printMath
printSBML
readSBML
translateMath
validateSBML
```


libsbml-5.0.0
examples
c
c++
csharp
java
perl
python
ruby
sample-models
from-spec
level-2
level-3
convertSBML
evaluateMath
extractReactionInfo
extractReactions
printMath
printSBML
readSBML
translateMath
validateSBML

```
#!/usr/bin/perl

use File::Basename;
use blib '../../src/bindings/perl';
use LibSBML;
use strict;

my $filename = shift()
    || do { printf STDERR "\n  usage:   @{[basename($0)]}<filename>\n\n";
                         exit (1);
                       };

my $rd = new LibSBML::SBMLReader();
my $d = $rd->readSBML($filename);

$d->printErrors();

my $m = $d->getModel() || exit (2);

my $level = $d->getLevel();
my $version = $d->getVersion();

printf("\n");
printf("File: %s (Level %u, version %u)\n",
                        basename($filename), $level, $version);
printf( "          ");

($level == 1) ? printf("model name: %s\n",  $m->getName()) : printf("  model id: %s\n",
                 $m->isSetId() ? $m->getId() : '(empty)');

printf( "       compartments: %d\n",   $m->getNumCompartments       () );
printf( "            species: %d\n",         $m->getNumSpecies            () );
printf( "         parameters: %d\n",      $m->getNumParameters         () );
printf( "          reactions: %d\n",         $m->getNumReactions          () );
printf( "\n" );
```

```
#!/usr/bin/perl

use File::Basename;
use blib '../../src/bindings/perl';
use LibSBML;
use strict;

my $filename = shift()
    || do { printf STDERR "\n  usage:   @{[basename($0)]}<filename>\n\n";
                        exit (1);
                      };

my $rd = new LibSBML::SBMLReader();
my $d = $rd->readSBML($filename);

$d->printErrors();

my $m = $d->getModel() || exit (2);

my $level = $d->getLevel();
my $version = $d->getVersion();

printf("\n");
printf("File: %s (Level %u, version %u)\n",
                        basename($filename), $level, $version);
printf( "         ");

($level == 1) ? printf("model name: %s\n",  $m->getName()) : printf("  model id: %s\n",
                  $m->isSetId() ? $m->getId() : '(empty)');

printf( "       compartments: %d\n",   $m->getNumCompartments       () );
printf( "            species: %d\n",        $m->getNumSpecies            () );
printf( "         parameters: %d\n",      $m->getNumParameters         () );
printf( "          reactions: %d\n",        $m->getNumReactions          () );
printf( "\n" );
```

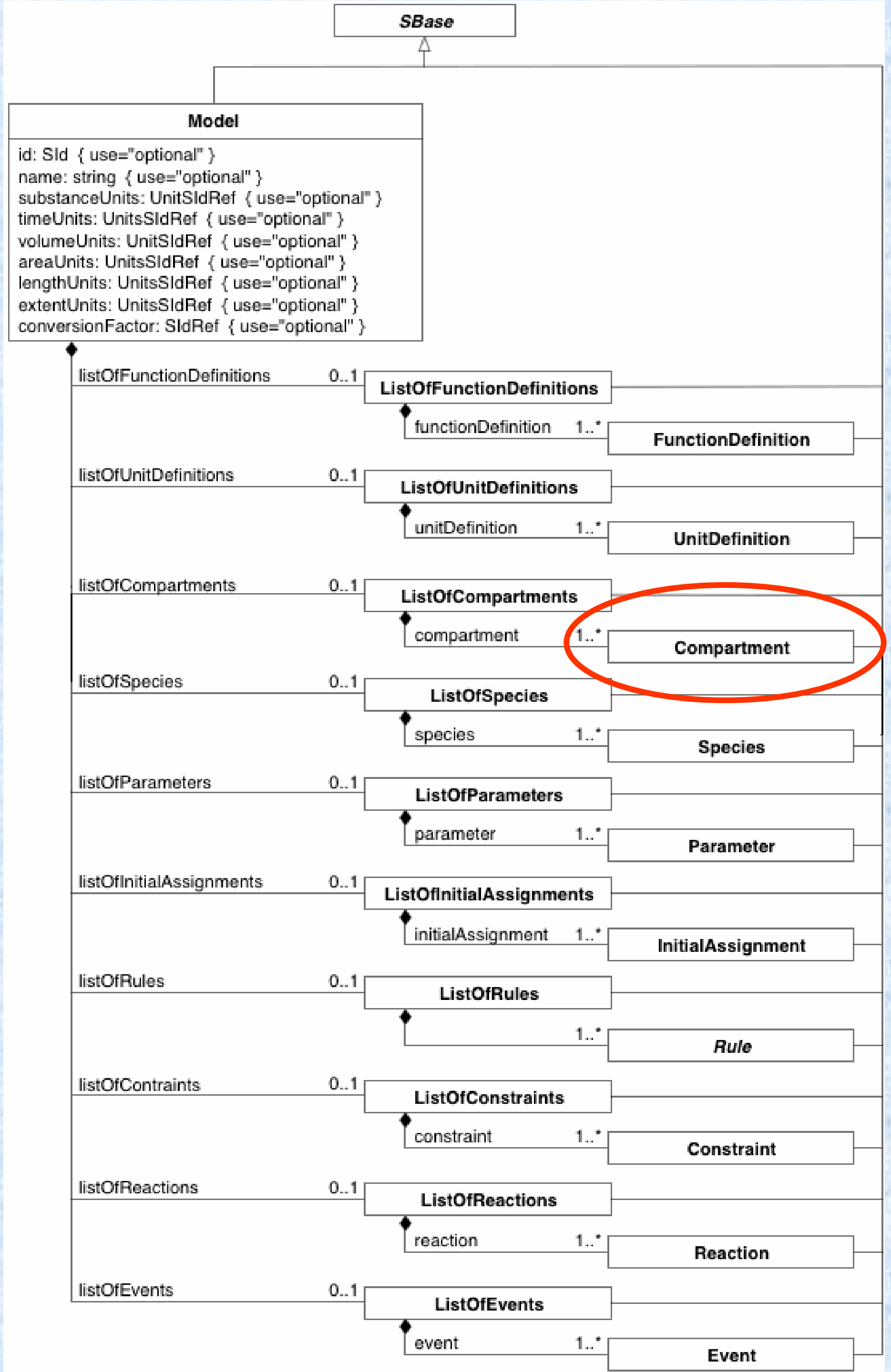
```
#!/usr/bin/perl

use File::Basename;
use blib '../../src/bindings/perl';
use LibSBML;
use strict;

my $filename = shift()
    || do { printf STDERR "\n  usage:   @{[basename($0)]}<filename>\n\n";
                          exit (1);
                        };

my $rd = new LibSBML::SBMLReader();
my $d = $rd->readSBML($filename);

$d->printErrors();

my $m = $d->getModel() || exit (2);

my $level = $d->getLevel();
my $version = $d->getVersion();

printf("\n");
printf("File: %s (Level %u, version %u)\n",
                        basename($filename), $level, $version);
printf( "         ");

($level == 1) ? printf("model name: %s\n",  $m->getName()) : printf("  model id: %s\n",
                  $m->isSetId() ? $m->getId() : '(empty)');

printf( "       compartments: %d\n",   $m->getNumCompartments       () );
printf( "            species: %d\n",        $m->getNumSpecies            () );
printf( "         parameters: %d\n",      $m->getNumParameters         () );
printf( "          reactions: %d\n",        $m->getNumReactions          () );
printf( "\n" );
```

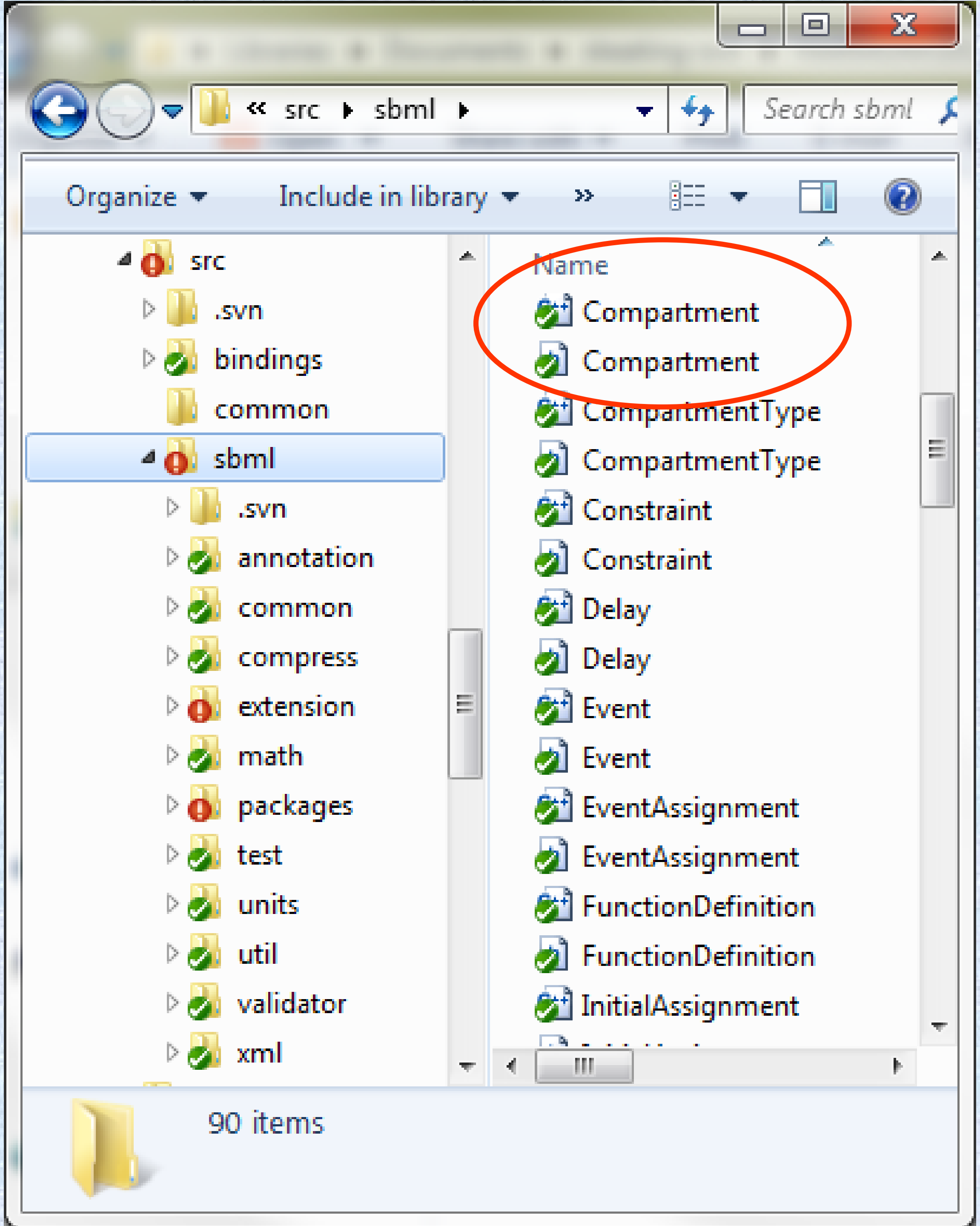
```perl
#!/usr/bin/perl

use File::Basename;
use blib '../../src/bindings/perl';
use LibSBML;
use strict;

my $filename = shift()
    || do { printf STDERR "\n  usage:   @{[basename($0)]}<filename>\n\n";
                          exit (1);
                        };

my $rd = new LibSBML::SBMLReader();
my $d = $rd->readSBML($filename);

$d->printErrors();

my $m = $d->getModel() || exit (2);

my $level = $d->getLevel();
my $version = $d->getVersion();

printf("\n");
printf("File: %s (Level %u, version %u)\n",
                        basename($filename), $level, $version);
printf( "          ");

($level == 1) ? printf("model name: %s\n",  $m->getName()) : printf("  model id: %s\n",
                  $m->isSetId() ? $m->getId() : '(empty)');

printf( "      compartments: %d\n",   $m->getNumCompartments       () );
printf( "           species: %d\n",        $m->getNumSpecies           () );
printf( "        parameters: %d\n",      $m->getNumParameters        () );
printf( "         reactions: %d\n",        $m->getNumReactions          () );
printf( "\n" );
```

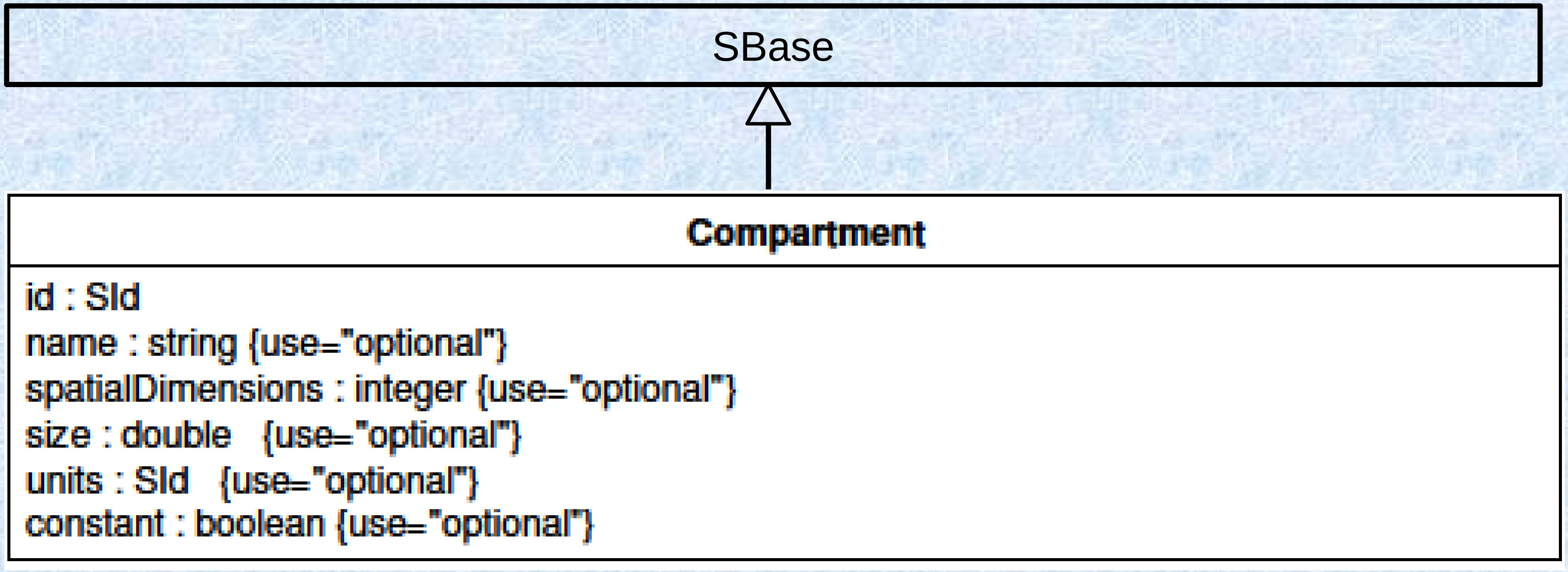
Manipulating SBML – the API

Components in SBML specification

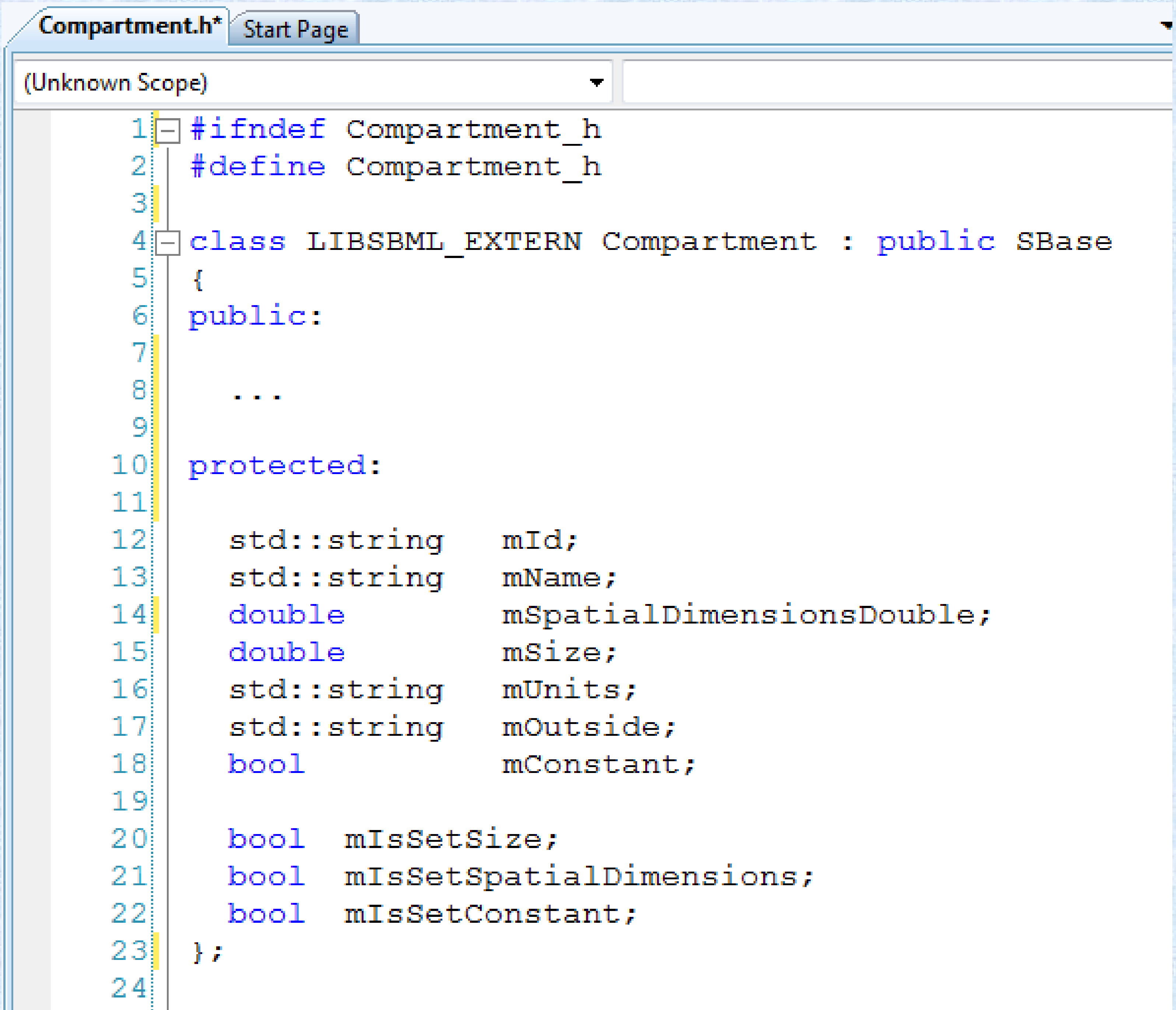
Components in SBML specification



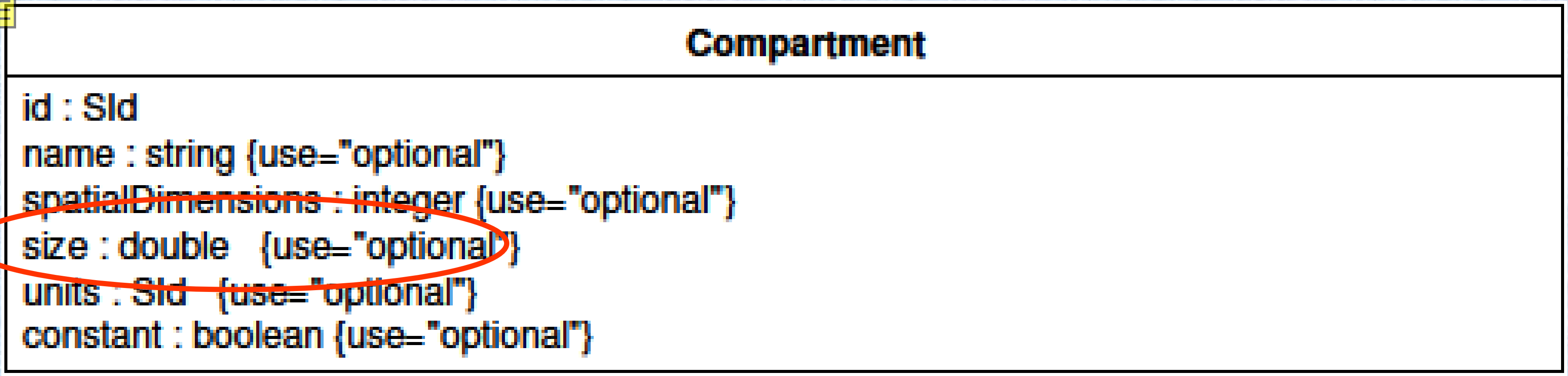

« src ▸ sbml ▸
Search sbml
Organize
Include in library
src
.svn
bindings
common
sbml
.svn
annotation
common
compress
extension
math
packages
test
units
util
validator
xml
Name
Compartment
Compartment
CompartmentType
CompartmentType
Constraint
Constraint
Delay
Delay
Event
Event
EventAssignment
EventAssignment
FunctionDefinition
FunctionDefinition
InitialAssignment
90 items


SBase
Compartment
id : SId
name : string {use="optional"}
spatialDimensions : integer {use="optional"}
size : double {use="optional"}
units : SId {use="optional"}
constant : boolean {use="optional"}

```
<compartment id="Yeast" units="litre" constant="true"/>
```

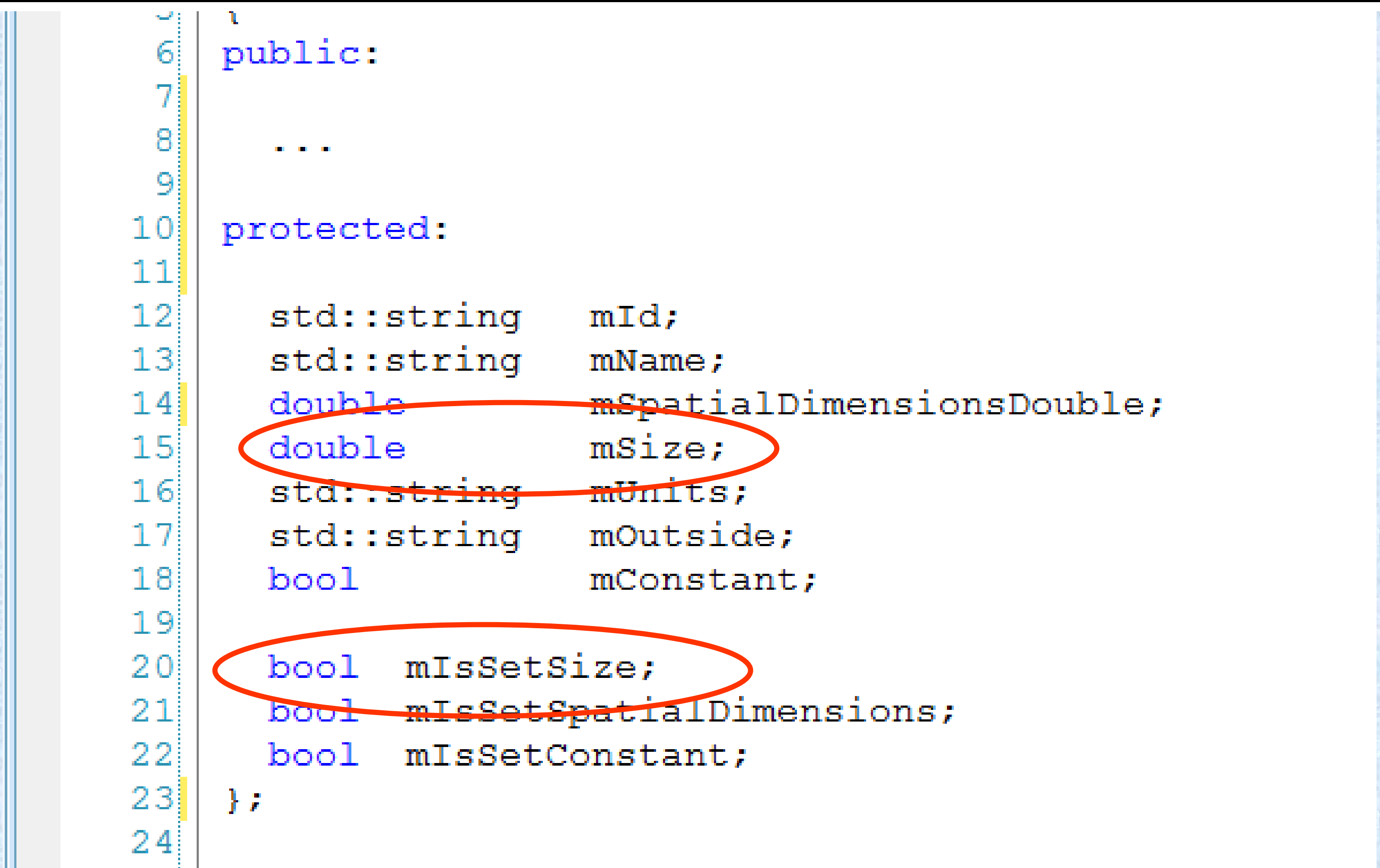


```
#ifndef Compartment_h
#define Compartment_h

class LIBSBML_EXTERN Compartment : public SBase
{
public:

  ...

protected:

  std::string   mId;
  std::string   mName;
  double        mSpatialDimensionsDouble;
  double        mSize;
  std::string   mUnits;
  std::string   mOutside;
  bool          mConstant;

  bool  mIsSetSize;
  bool  mIsSetSpatialDimensions;
  bool  mIsSetConstant;
};

```



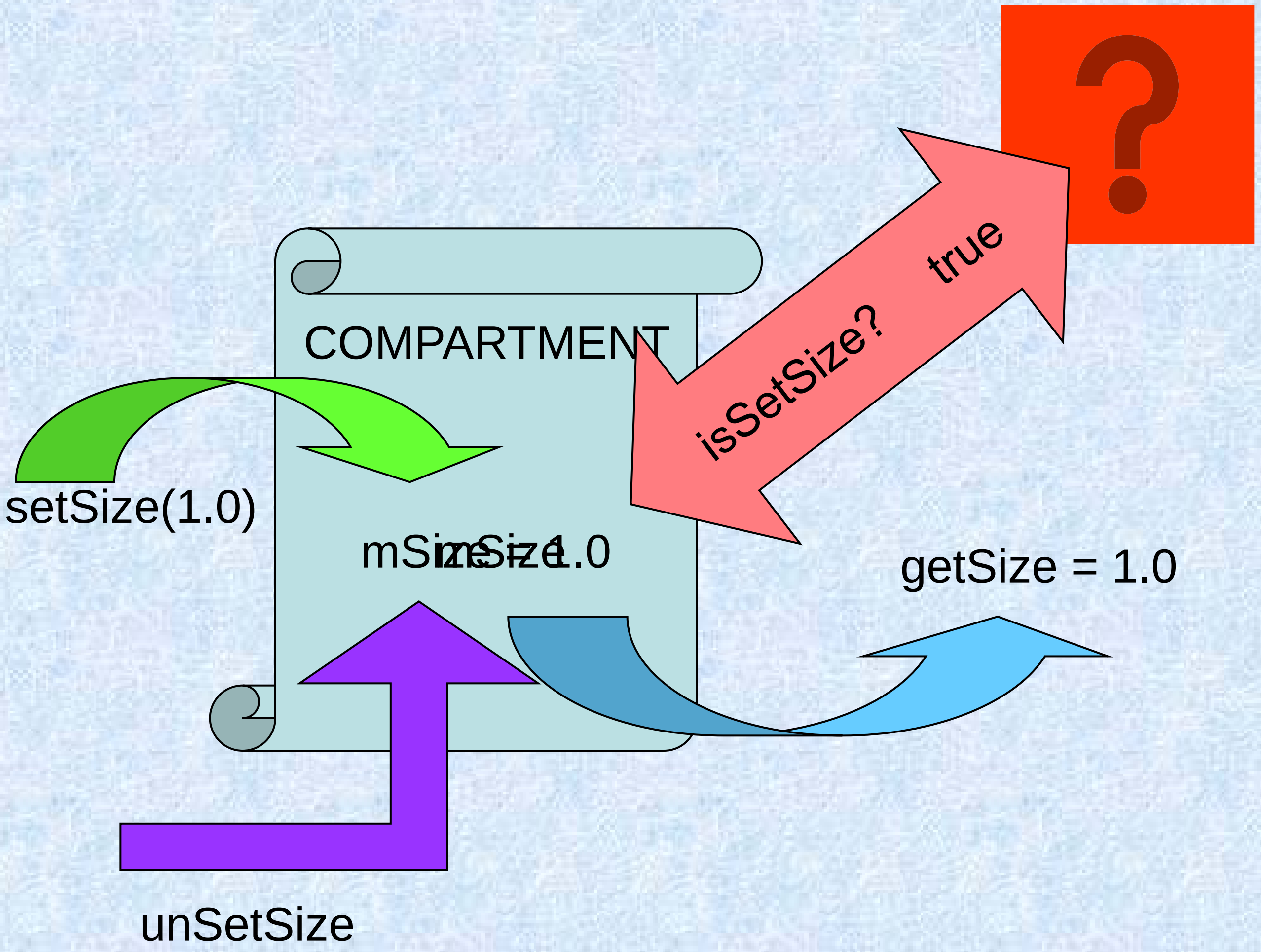
Compartment

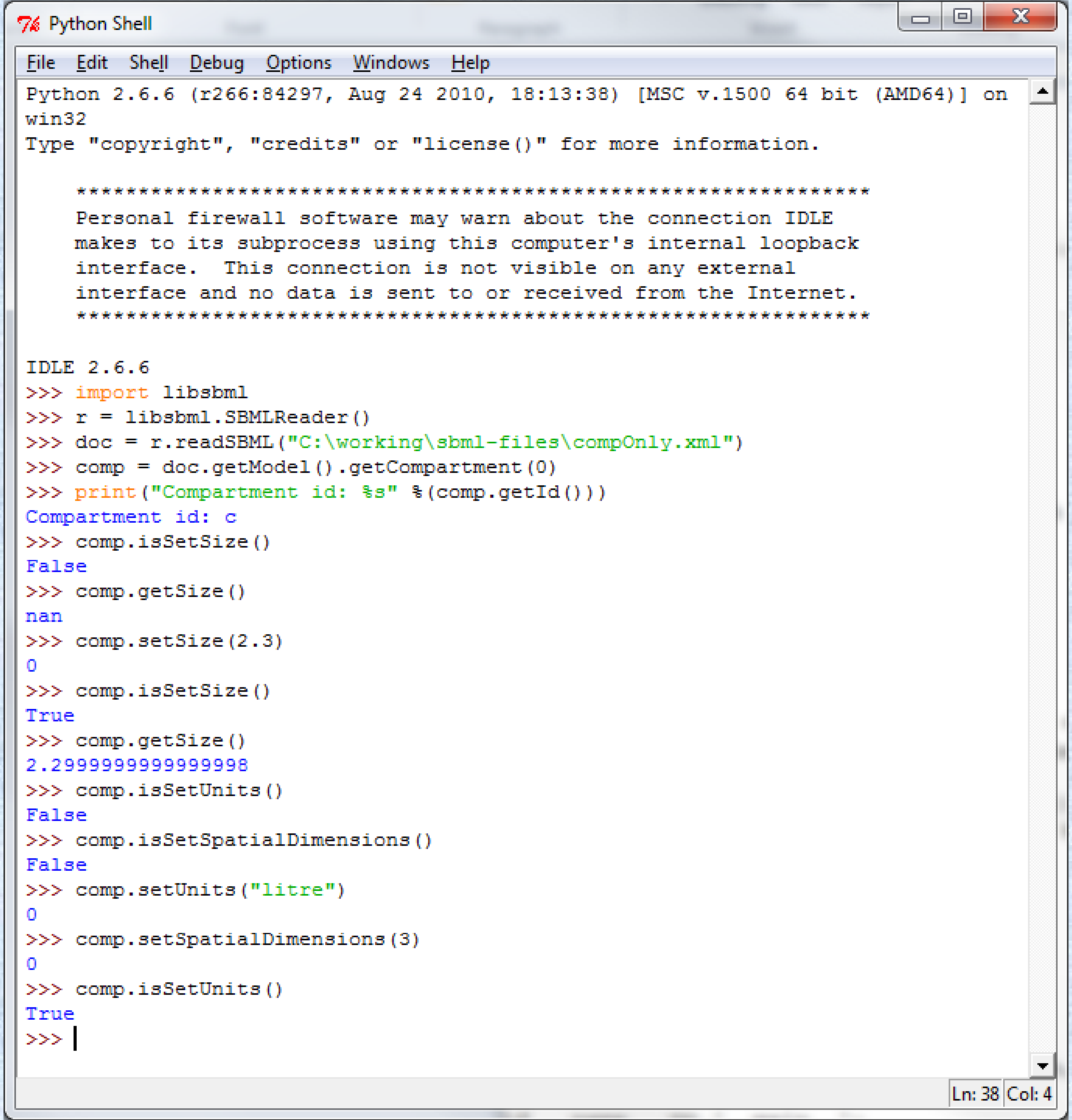
id : SId
name : string {use="optional"}
spatialDimensions : integer {use="optional"}
size : double {use="optional"}
units : SId {use="optional"}
constant : boolean {use="optional"}



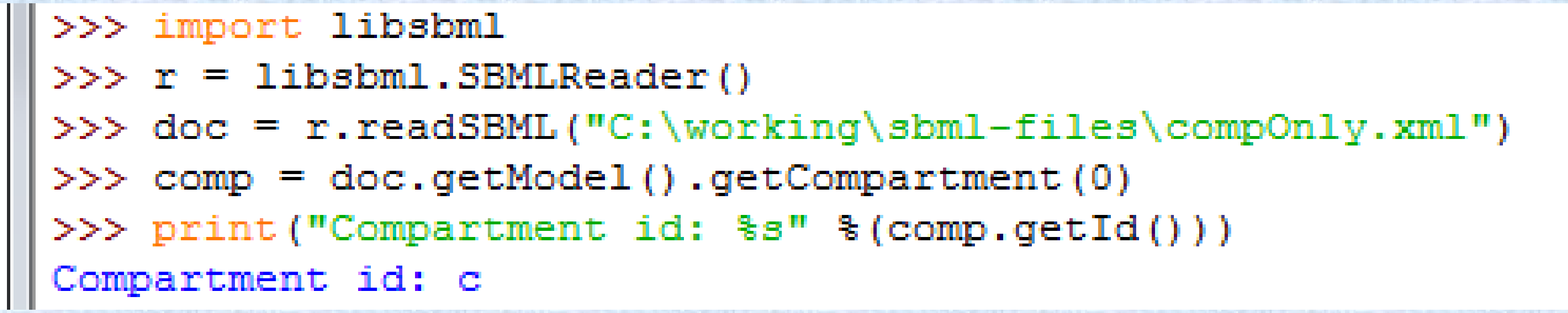
```
{
public:

   ...

protected:

   std::string   mId;
   std::string   mName;
   double        mSpatialDimensionsDouble;
   double        mSize;
   std::string   mUnits;
   std::string   mOutside;
   bool          mConstant;

   bool   mIsSetSize;
   bool   mIsSetSpatialDimensions;
   bool   mIsSetConstant;
};
```

Attribute API

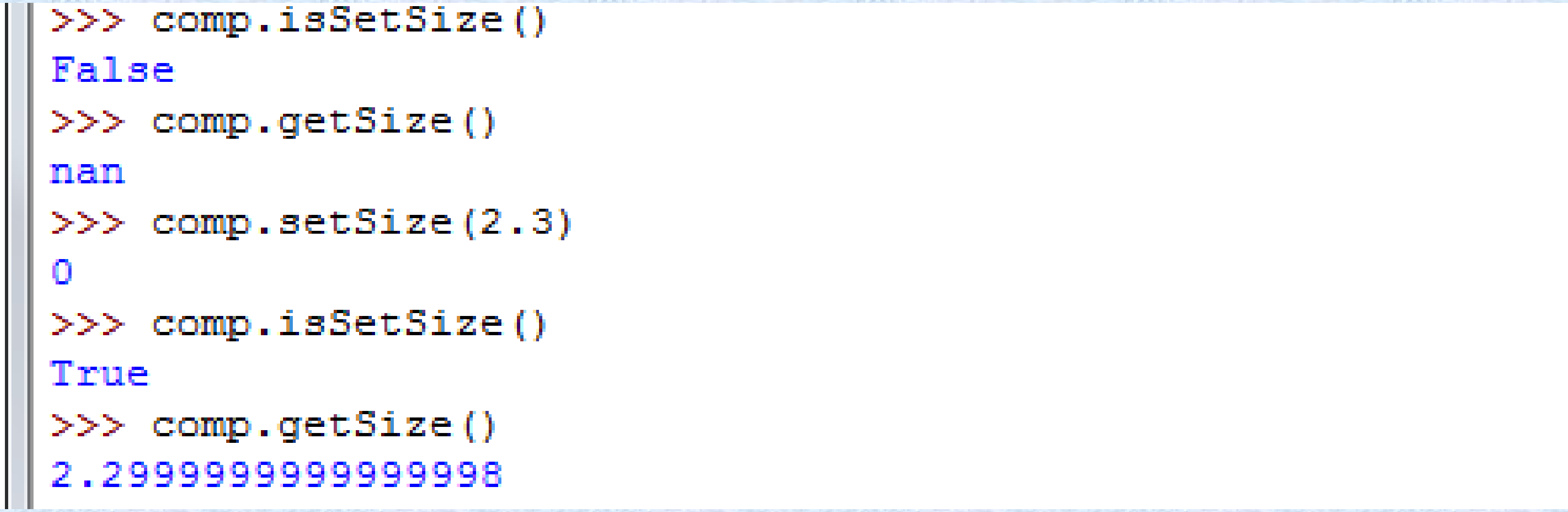




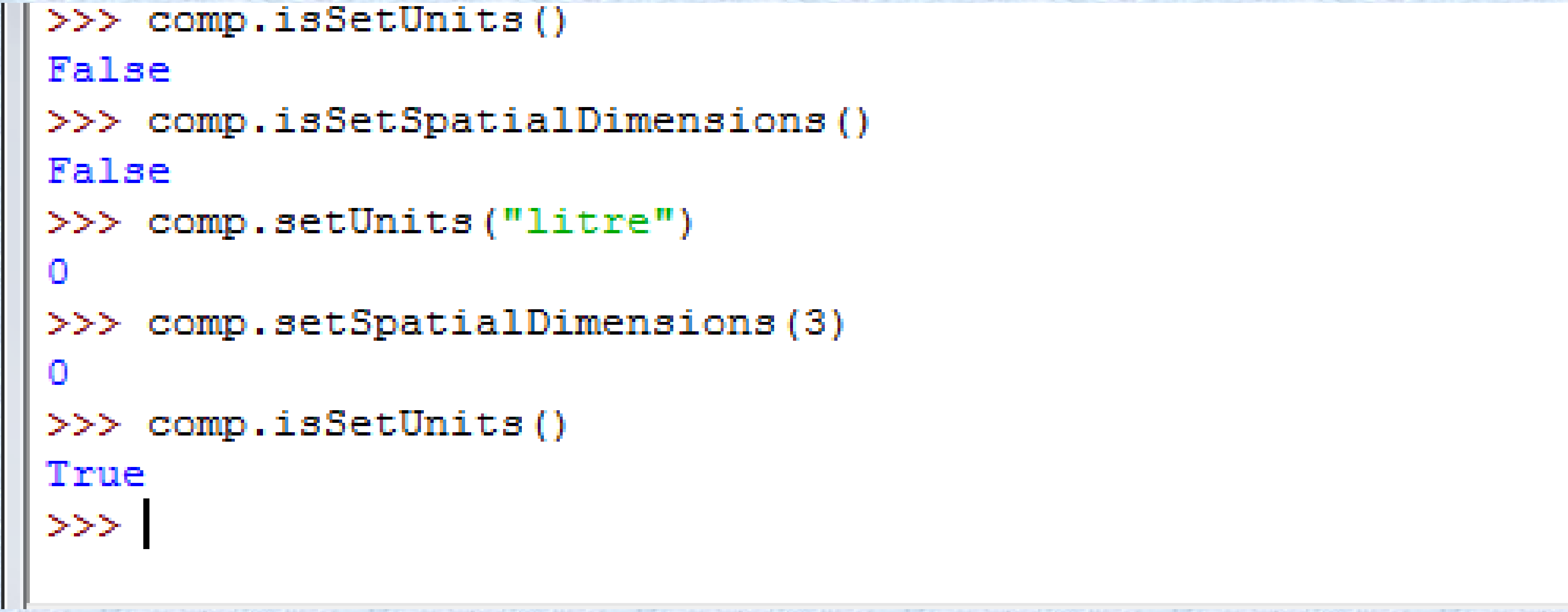
```
Python 2.6.6 (r266:84297, Aug 24 2010, 18:13:38) [MSC v.1500 64 bit (AMD64)] on
win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 2.6.6
>>> import libsbml
>>> r = libsbml.SBMLReader()
>>> doc = r.readSBML("C:\working\sbml-files\compOnly.xml")
>>> comp = doc.getModel().getCompartment(0)
>>> print("Compartment id: %s" %(comp.getId()))
Compartment id: c
>>> comp.isSetSize()
False
>>> comp.getSize()
nan
>>> comp.setSize(2.3)
0
>>> comp.isSetSize()
True
>>> comp.getSize()
2.2999999999999998
>>> comp.isSetUnits()
False
>>> comp.isSetSpatialDimensions()
False
>>> comp.setUnits("litre")
0
>>> comp.setSpatialDimensions(3)
0
>>> comp.isSetUnits()
True
>>> 
```



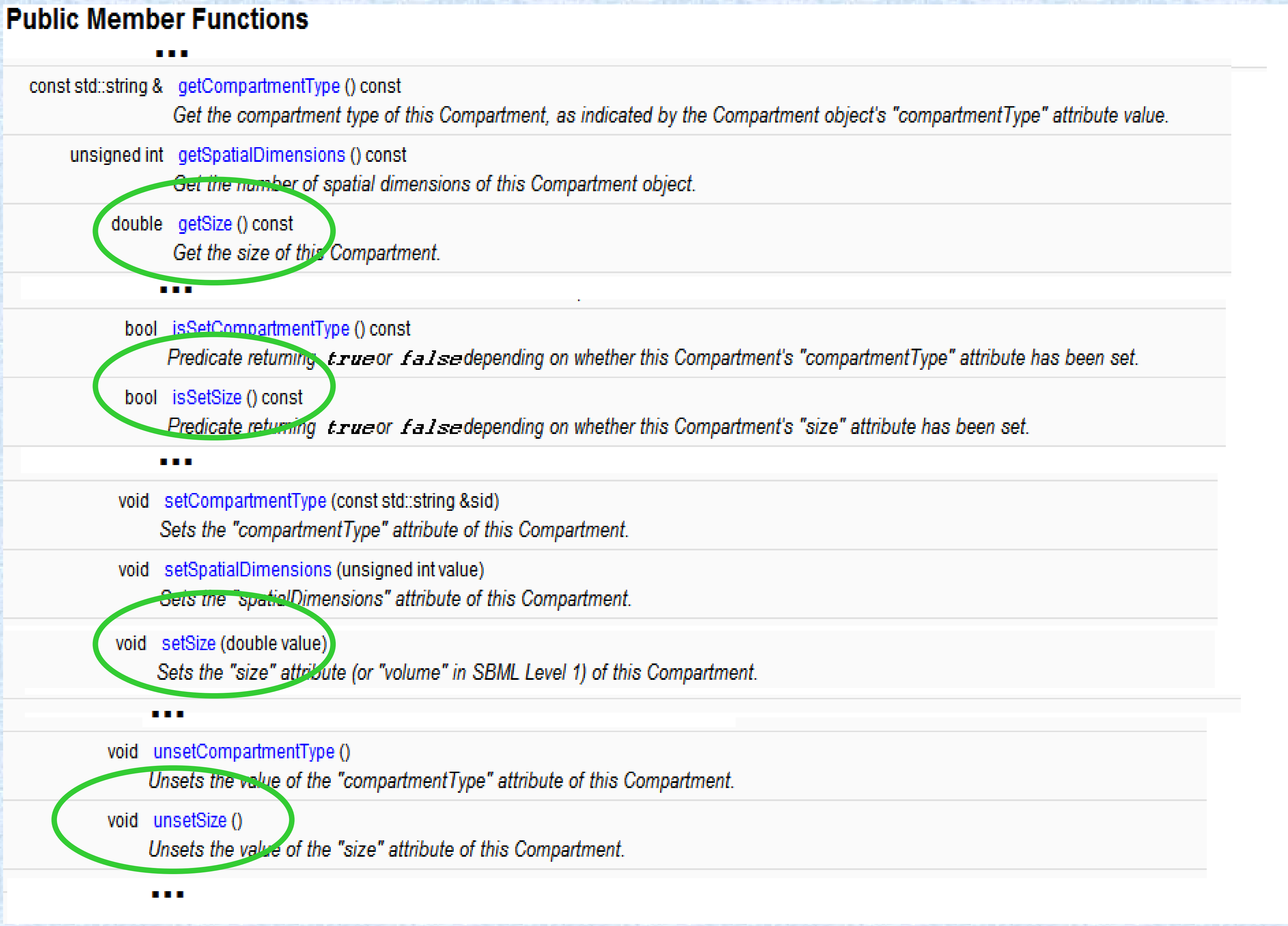
```
>>> import libsbml
>>> r = libsbml.SBMLReader()
>>> doc = r.readSBML("C:\working\sbml-files\compOnly.xml")
>>> comp = doc.getModel().getCompartment(0)
>>> print("Compartment id: %s" %(comp.getId()))
Compartment id: c
```



```
>>> comp.isSetSize()
False
>>> comp.getSize()
nan
>>> comp.setSize(2.3)
0
>>> comp.isSetSize()
True
>>> comp.getSize()
2.2999999999999998
```



```
>>> comp.isSetUnits()
False
>>> comp.isSetSpatialDimensions()
False
>>> comp.setUnits("litre")
0
>>> comp.setSpatialDimensions(3)
0
>>> comp.isSetUnits()
True
>>> 
```



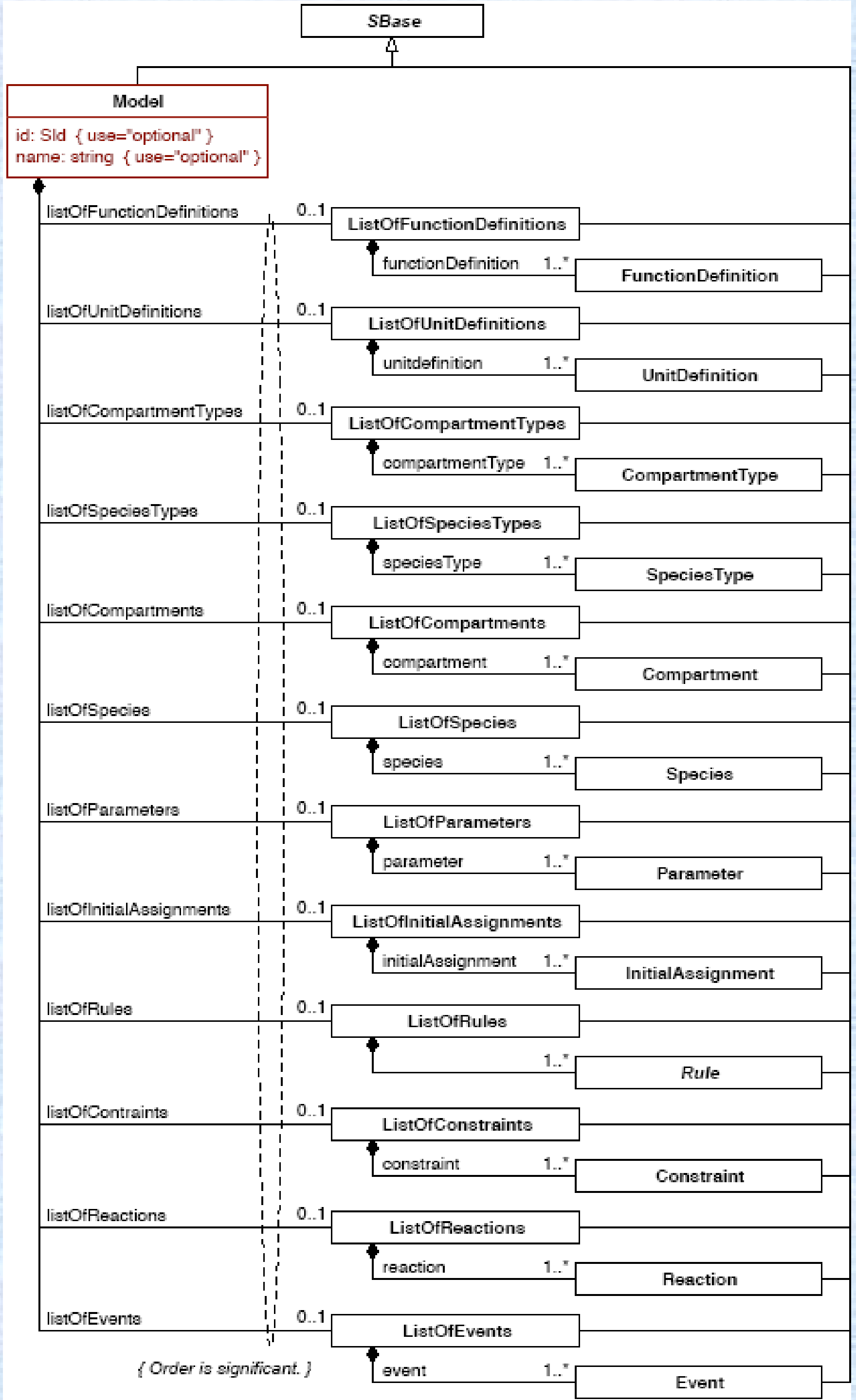
Public Member Functions

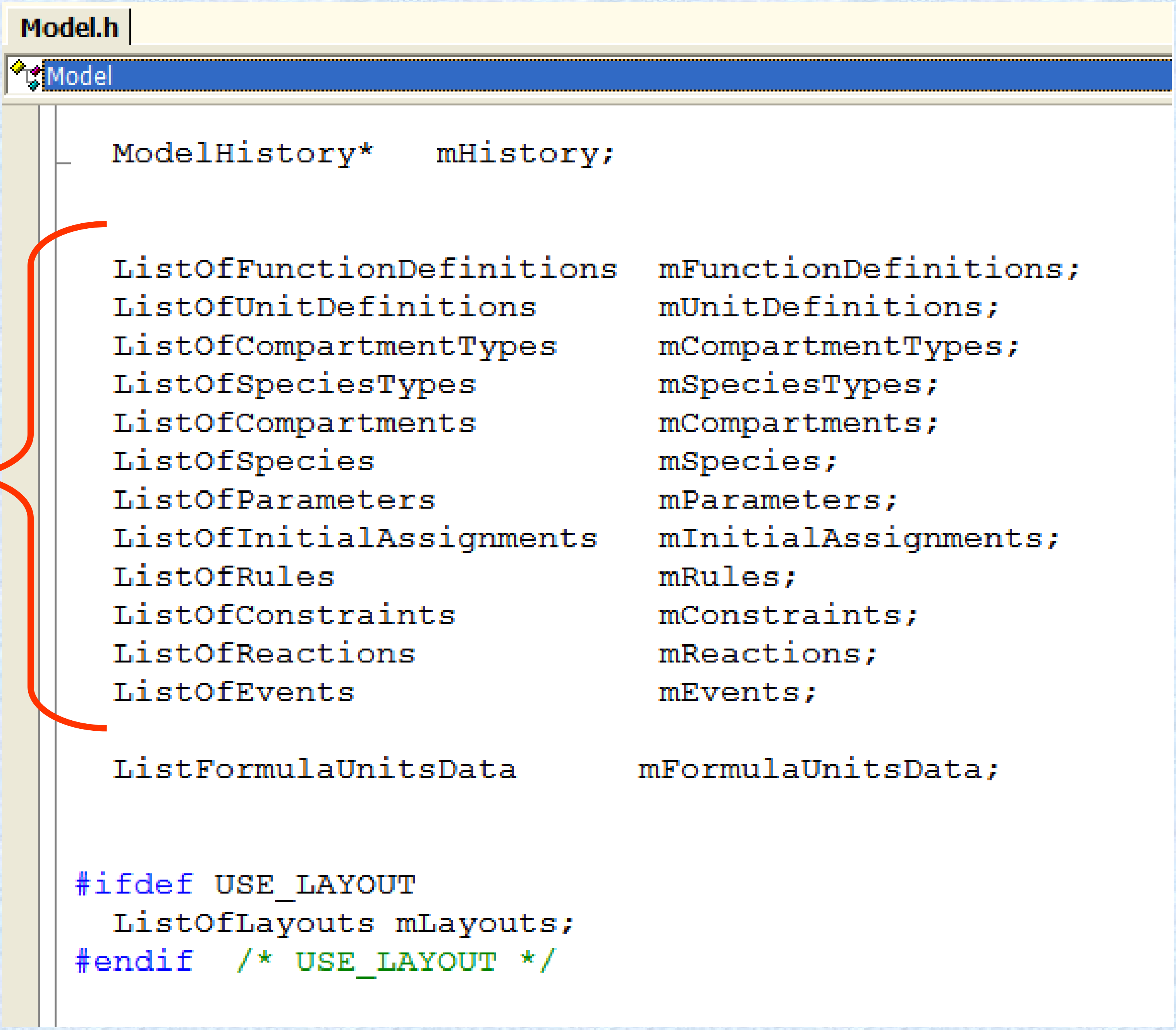
...

const std::string &	getCompartmentType () const *Get the compartment type of this Compartment, as indicated by the Compartment object's "compartmentType" attribute value.*
unsigned int	getSpatialDimensions () const *Get the number of spatial dimensions of this Compartment object.*
double	getSize () const *Get the size of this Compartment.*
...	
bool	isSetCompartmentType () const *Predicate returning* `true` *or* `false` *depending on whether this Compartment's "compartmentType" attribute has been set.*
bool	isSetSize () const *Predicate returning* `true` *or* `false` *depending on whether this Compartment's "size" attribute has been set.*
...	
void	setCompartmentType (const std::string &sid) *Sets the "compartmentType" attribute of this Compartment.*
void	setSpatialDimensions (unsigned int value) *Sets the "spatialDimensions" attribute of this Compartment.*
void	setSize (double value) *Sets the "size" attribute (or "volume" in SBML Level 1) of this Compartment.*
...	
void	unsetCompartmentType () *Unsets the value of the "compartmentType" attribute of this Compartment.*
void	unsetSize () *Unsets the value of the "size" attribute of this Compartment.*

...

ListOf…Elements


SBase
Model
id: SId { use="optional" }
name: string { use="optional" }
listOfFunctionDefinitions 0..1
ListOfFunctionDefinitions
functionDefinition 1..*
FunctionDefinition
listOfUnitDefinitions 0..1
ListOfUnitDefinitions
unitdefinition 1..*
UnitDefinition
listOfCompartmentTypes 0..1
ListOfCompartmentTypes
compartmentType 1..*
CompartmentType
listOfSpeciesTypes 0..1
ListOfSpeciesTypes
speciesType 1..*
SpeciesType
listOfCompartments 0..1
ListOfCompartments
compartment 1..*
Compartment
listOfSpecies 0..1
ListOfSpecies
species 1..*
Species
listOfParameters 0..1
ListOfParameters
parameter 1..*
Parameter
listOfInitialAssignments 0..1
ListOfInitialAssignments
initialAssignment 1..*
InitialAssignment
listOfRules 0..1
ListOfRules
1..*
Rule
listOfContraints 0..1
ListOfConstraints
constraint 1..*
Constraint
listOfReactions 0..1
ListOfReactions
reaction 1..*
Reaction
listOfEvents 0..1
ListOfEvents
event 1..*
Event
{ Order is significant. }

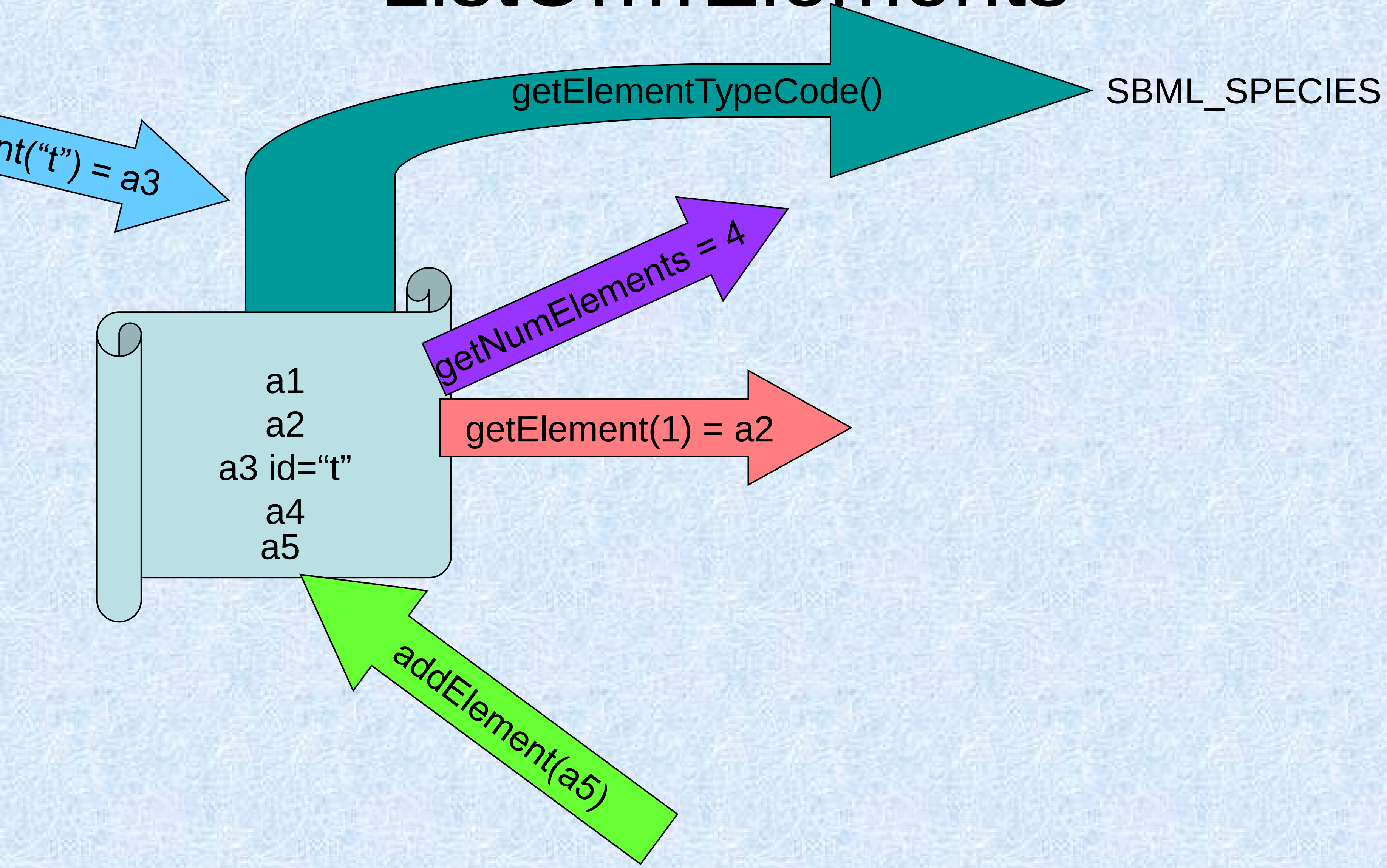


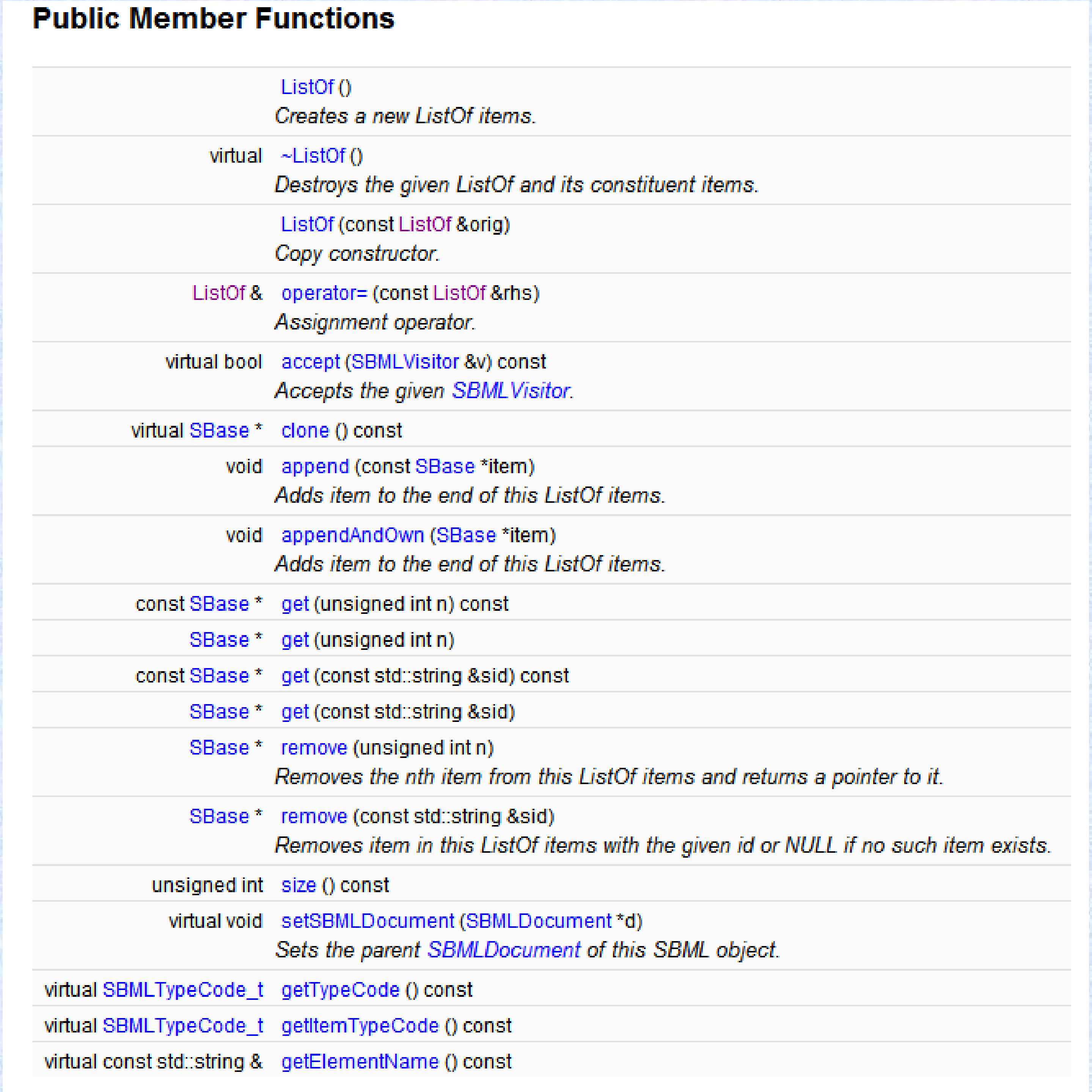
Model.h

Model

```
  ModelHistory*   mHistory;


  ListOfFunctionDefinitions  mFunctionDefinitions;
  ListOfUnitDefinitions      mUnitDefinitions;
  ListOfCompartmentTypes     mCompartmentTypes;
  ListOfSpeciesTypes         mSpeciesTypes;
  ListOfCompartments         mCompartments;
  ListOfSpecies              mSpecies;
  ListOfParameters           mParameters;
  ListOfInitialAssignments   mInitialAssignments;
  ListOfRules                mRules;
  ListOfConstraints          mConstraints;
  ListOfReactions            mReactions;
  ListOfEvents               mEvents;

  ListFormulaUnitsData      mFormulaUnitsData;


#ifdef USE_LAYOUT
  ListOfLayouts mLayouts;
#endif  /* USE_LAYOUT */
```

ListOf...Elements



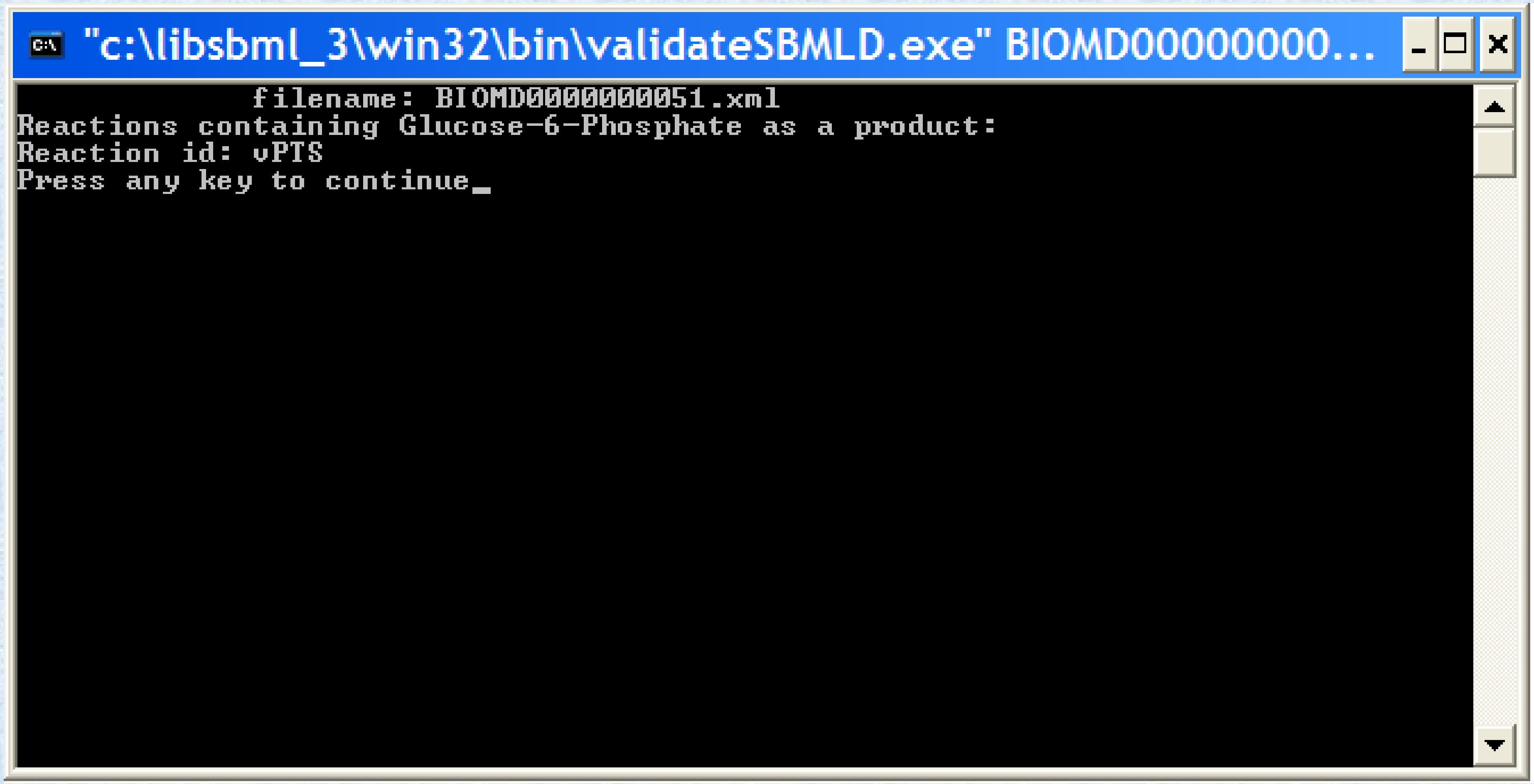


Public Member Functions

	ListOf () *Creates a new ListOf items.*
virtual	~ListOf () *Destroys the given ListOf and its constituent items.*
	ListOf (const ListOf &orig) *Copy constructor.*
ListOf &	operator= (const ListOf &rhs) *Assignment operator.*
virtual bool	accept (SBMLVisitor &v) const *Accepts the given SBMLVisitor.*
virtual SBase *	clone () const
void	append (const SBase *item) *Adds item to the end of this ListOf items.*
void	appendAndOwn (SBase *item) *Adds item to the end of this ListOf items.*
const SBase *	get (unsigned int n) const
SBase *	get (unsigned int n)
const SBase *	get (const std::string &sid) const
SBase *	get (const std::string &sid)
SBase *	remove (unsigned int n) *Removes the nth item from this ListOf items and returns a pointer to it.*
SBase *	remove (const std::string &sid) *Removes item in this ListOf items with the given id or NULL if no such item exists.*
unsigned int	size () const
virtual void	setSBMLDocument (SBMLDocument *d) *Sets the parent SBMLDocument of this SBML object.*
virtual SBMLTypeCode_t	getTypeCode () const
virtual SBMLTypeCode_t	getItemTypeCode () const
virtual const std::string &	getElementName () const

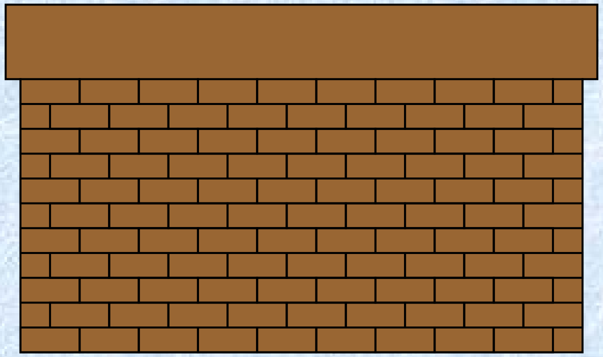
Which reactions produce a particular species ?

```
unsigned int n;
Model* model = document->getModel();
std::string speciesId;
for (n = 0; n < model->getNumSpecies(); n++)
{
  if (model->getSpecies(n)->getName() == "Glucose-6-Phosphate")
  {
    speciesId = model->getSpecies(n)->getId();
    break;
  }
}
ListOfReactions * myReactions = new ListOfReactions();

for (n = 0; n < model->getNumReactions(); n++)
{
  Reaction * r = model->getReaction(n);

  if (r->getProduct(speciesId) != NULL)
  {
    myReactions->appendAndOwn(r);
  }
}
cout << "Reactions containing Glucose-6-Phosphate as a product:\n";
for (n = 0; n < myReactions->size(); n++)
{
  cout << "Reaction id: " << myReactions->get(n)->getId() << endl;
}
```



```
             filename: BIOMD0000000051.xml
Reactions containing Glucose-6-Phosphate as a product:
Reaction id: vPTS
Press any key to continue_
```

Creating SBML – the API



```
use blib '../../src/bindings/perl';
use LibSBML;
use strict;
use blib '../../src/bindings/perl';
use LibSBML;
use strict;

# create the namespace for the level and
version of SBML

my $sbmlns = new
          LibSBML::SBMLNamespaces(3, 1);

# create the document

my $document=new
        LibSBML::SBMLDocument($sbmlns);
```

```
my $document=new LibSBML::SBMLDocument($sbmlns);

# create the Model

my $model=$document->createModel();

$model->setId("TestModel");
```

```
my $document=new LibSBML::SBMLDocument($sbmlns);

my $model=$document->createModel();

# create the Compartment

my $compartment=
                    $model->createCompartment();

$compartment->setId("Compartment_1");

$compartment->setConstant(1);
```

```
my $document=new LibSBML::SBMLDocument($sbmlns);
my $model=$document->createModel();
my $compartment=$model->createCompartment();
# create the Species

my $species1=$model->createSpecies();

$species1->setId("Species_1");

$species1->setCompartment
                                        ($compartment->getId());
$species1->setHasOnlySubstanceUnits(0);
$species1->setBoundaryCondition(0);
$species1->setConstant(0);
```

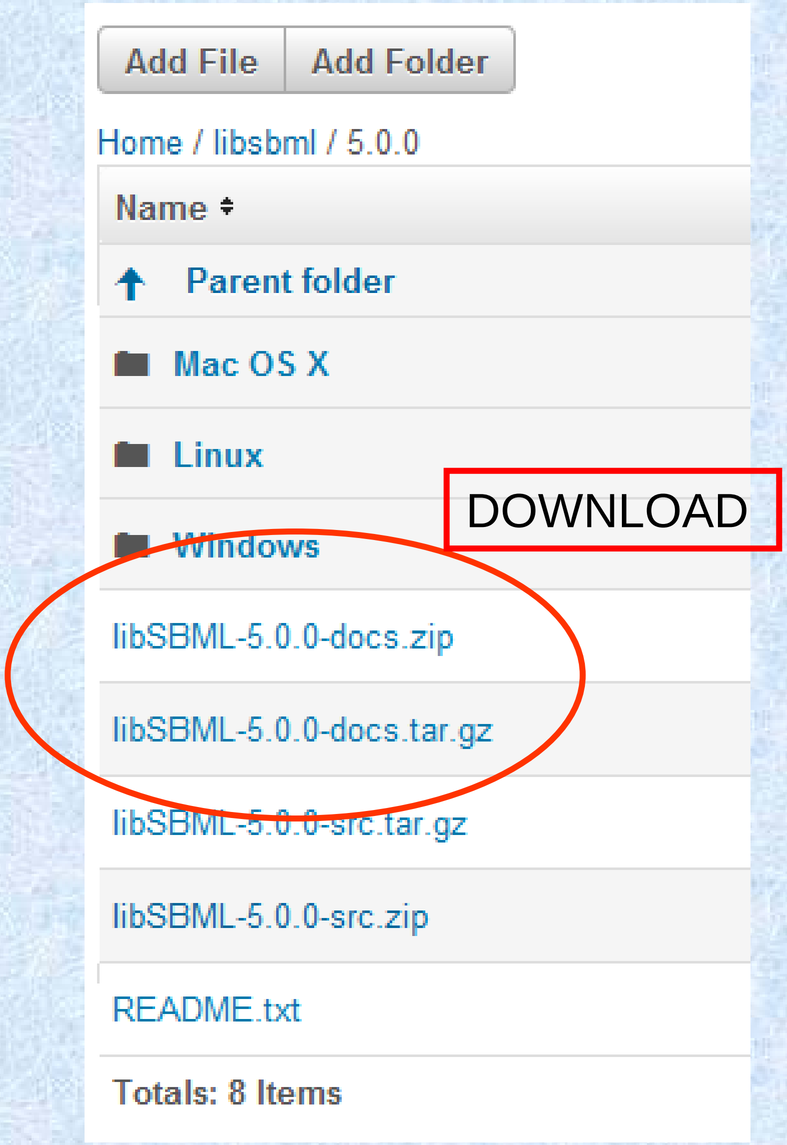
```
my $document=new LibSBML::SBMLDocument($sbmlns);

my $model=$document->createModel();

my $compartment=$model->createCompartment();

my $species1=$model->createSpecies();

my $species2=$model->createSpecies();


# create the Reactions


my $reaction1=$model->createReaction();
$reaction1->setId("Reaction_1");
$reaction1->setReversible(0);
$reaction1->setFast(0);
```

```
my $reaction1=$model->createReaction();
$reaction1->setId("Reaction_1");
$reaction1->setReversible(0);
$reaction1->setFast(0);

# create the Reactant


my $reference1=$reaction1->createReactant();
$reference1->setSpecies($species1->getId());
$reference1->setId("SpeciesReference_1");
$reference1->setConstant(0);
```

```
my $reaction1=$model->createReaction();

my $reference1=$reaction1->createReactant();

# create the Product


my $reference2= $reaction1->createProduct();
$reference2->setSpecies($species2->getId());
$reference2->setId("SpeciesReference_2");
$reference2->setConstant(0);
```

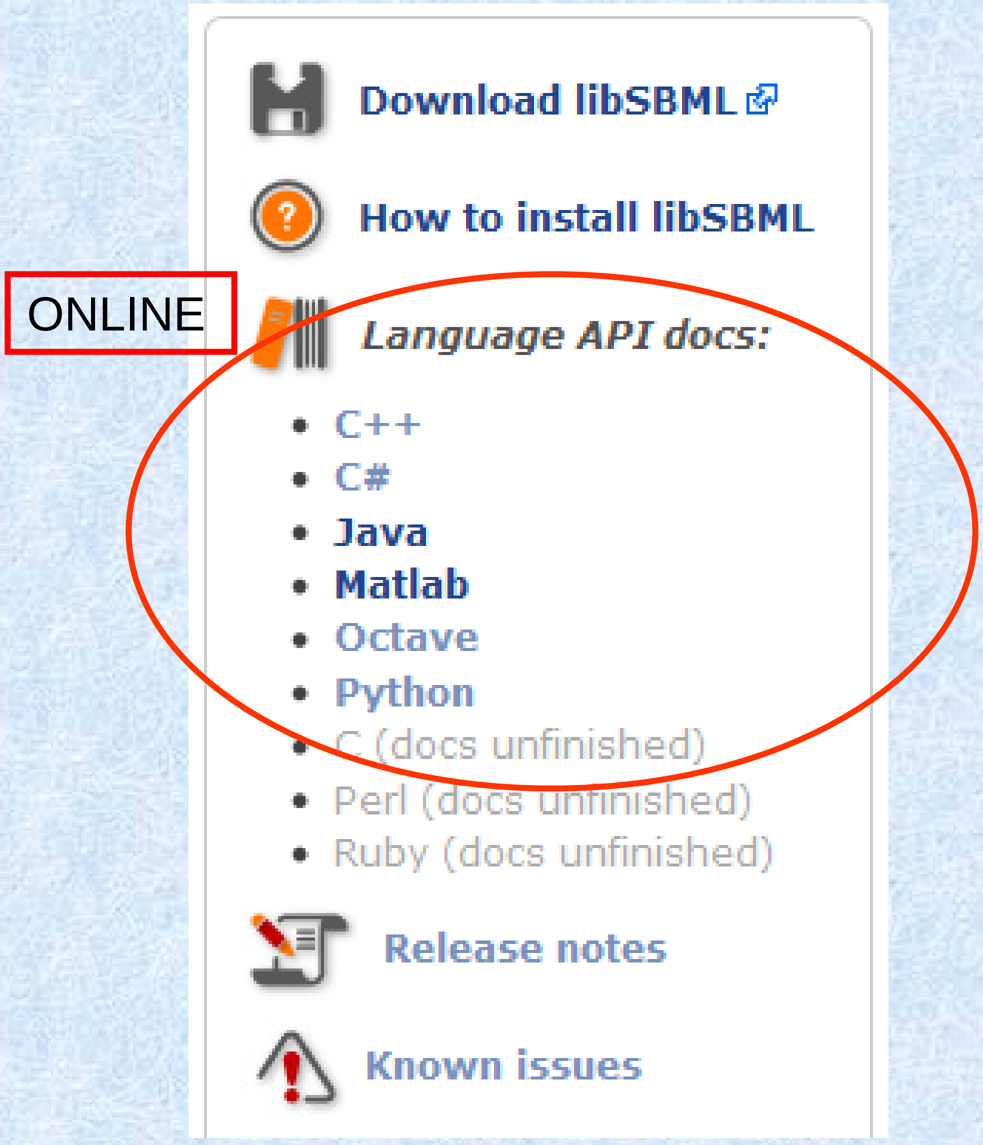
```
my $reaction1=$model->createReaction();
my $reference1=$reaction1->createReactant();
my $reference2= $reaction1->createProduct();

# create the KineticLaw

my $kineticLaw=
                    $reaction1->createKineticLaw();

my $KLmath=
          LibSBML::parseFormula(‘species2 * 2’);

$kineticLaw->setMath($KLmath);
```

Documentation



Home / libsbml / 5.0.0

Name
Parent folder
Mac OS X
Linux
Windows
libSBML-5.0.0-docs.zip
libSBML-5.0.0-docs.tar.gz
libSBML-5.0.0-src.tar.gz
libSBML-5.0.0-src.zip
README.txt
Totals: 8 Items



Download libSBML

How to install libSBML

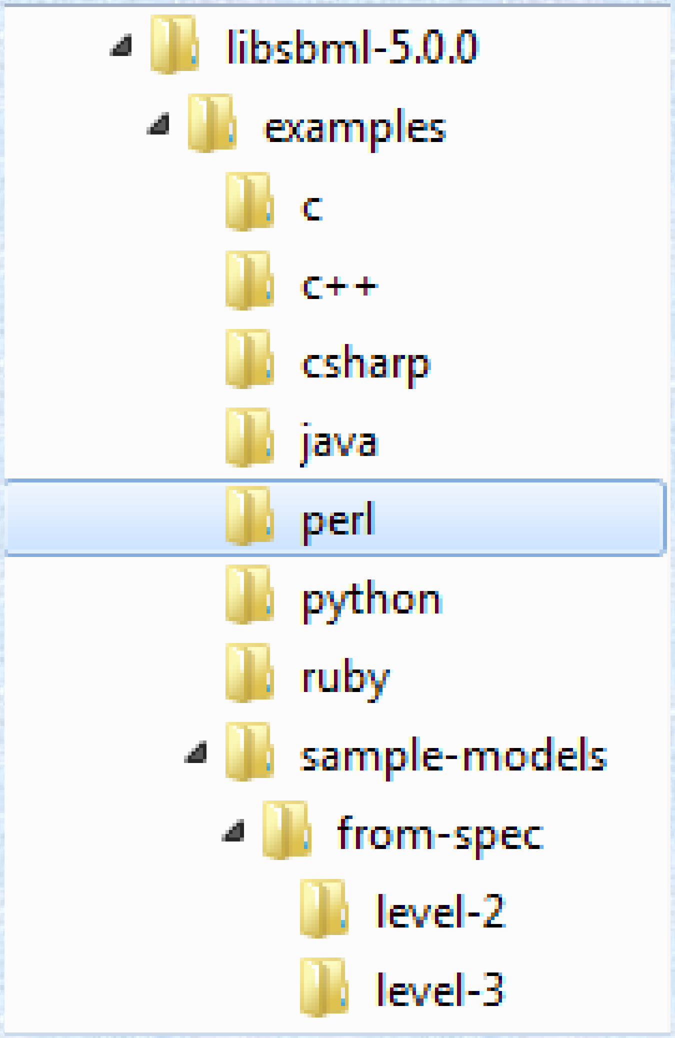
Language API docs:

- C++
- C#
- Java
- Matlab
- Octave
- Python
- C (docs unfinished)
- Perl (docs unfinished)
- Ruby (docs unfinished)

Release notes

Known issues

Getting started ...



```
libsbml-5.0.0
  examples
    c
    c++
    csharp
    java
    perl
    python
    ruby
    sample-models
      from-spec
        level-2
        level-3
```

http://sbml.org/Software/libSBML/Tutorials