NAME

Molecule - Molecule class

SYNOPSIS

use Molecule;

use Molecule qw(:all);

DESCRIPTION

Molecule class provides the following methods:

new, AddAtom, AddAtoms, AddBond, AddBonds, AddHydrogens, AddPolarHydrogens, ClearRings, Copy, DeleteAromaticity, DeleteAtom, DeleteAtoms, DeleteBond, DeleteBonds, DeleteHydrogens, DeletePolarHydrogens, DetectAromaticity, DetectRings, FormatElementalCompositionInformation, GetAllAtomPaths, GetAllAtomPathsStartingAt, GetAllAtomPathsStartingAtWithLength, GetAllAtomPathsStartingAtWithLengthUpto, GetAllAtomPathsWithLength, GetAllAtomPathsWithLengthUpto, GetAromaticRings, GetAromaticityModel, GetAtomNeighborhoods, GetAtomNeighborhoodsWithRadiusUpto, GetAtomNeighborhoodsWithSuccessorAtoms, GetAtomNeighborhoodsWithSuccessorAtomsAndRadiusUpto, GetAtomPathBonds, GetAtomPaths, GetAtomPathsBetween, GetAtomPathsStartingAt, GetAtomPathsStartingAtWithLength, GetAtomPathsStartingAtWithLengthUpto, GetAtomPathsWithLength, GetAtomPathsWithLengthUpto, GetAtoms, GetBonds, GetCharge, GetConnectedComponents, GetConnectedComponentsAtoms, GetDimensionality, GetElementalComposition, GetElementsAndNonElements, GetExactMass, GetFormalCharge, GetFreeRadicalElectrons, GetFusedAndNonFusedRings, GetLargestConnectedComponent, GetLargestConnectedComponentAtoms, GetLargestRing, GetMolecularFormula, GetMolecularWeight, GetNumOfAromaticRings, GetNumOfAtoms, GetNumOfBonds, GetNumOfConnectedComponents, GetNumOfElementsAndNonElements, GetNumOfHeavyAtoms, GetNumOfHydrogenAtoms, GetNumOfMissingHydrogenAtoms, GetNumOfNonHydrogenAtoms, GetNumOfRings, GetNumOfRingsWithEvenSize, GetNumOfRingsWithOddSize, GetNumOfRingsWithSize, GetNumOfRingsWithSizeGreaterThan, GetNumOfRingsWithSizeLessThan, GetRingBonds, GetRingBondsFromRings, GetRings, GetRingsWithEvenSize, GetRingsWithOddSize, GetRingsWithSize, GetRingsWithSizeGreaterThan, GetRingsWithSizeLessThan, GetSizeOfLargestRing, GetSizeOfSmallestRing, GetSmallestRing, GetSpinMultiplicity, GetSupportedAromaticityModels, GetTopologicallySortedAtoms, GetValenceModel, HasAromaticAtomsInRings, HasAromaticAtomsNotInRings, HasAromaticRings, HasAtom, HasBond, HasFusedRings, HasNoRings, HasOnlyOneRing, HasRings, IsAromatic, IsMolecule, IsRingAromatic, IsSupportedAromaticityModel, IsThreeDimensional, IsTwoDimensional, KeepLargestComponent, KekulizeAromaticAtoms, NewAtom, NewBond, SetActiveRings, SetAromaticityModel, SetID, SetValenceModel, StringifyMolecule

The following methods can also be used as functions:

FormatElementalCompositionInformation, IsMolecule

Molecule class is derived from ObjectProperty base class which provides methods not explicitly defined in Molecule or ObjectProperty class using Perl's AUTOLOAD functionality. These methods are generated on-the-fly for a specified object property:

```
Set<PropertyName>(<PropertyValue>);
$PropertyValue = Get<PropertyName>();
Delete<PropertyName>();
```

METHODS

new

```
$NewMolecule = new Molecule([%PropertyNameAndValues]);
```

Using specified *Atom* property names and values hash, new method creates a new object and returns a reference to newly created Atom object. By default, the following properties are initialized:

```
ID = SequentialObjectID
Name = "Molecule <SequentialObjectID>"
```

Examples:

```
$Molecule = new Molecule();

$WaterMolecule = new Molecule('Name' => 'Water');
```

```
$Oxygen = new Atom('AtomSymbol' => 'O', 'XYZ' => [0, 0, 0]);
$Hydrogen1 = new Atom('AtomSymbol' => 'H',
                      'XYZ' => [0.7144, 0.4125, 0]);
$Hydrogen2 = new Atom('AtomSymbol' => 'H',
                      'XYZ' => [1.1208, -0.2959, 0]);
$WaterMolecule->AddAtoms($Oxygen, $Hydrogen1, $Hydrogen2);

$Bond1 = new Bond('Atoms' => [$Oxygen, $Hydrogen1],
                  'BondOrder' => 1);
$Bond2 = new Bond('Atoms' => [$Oxygen, $Hydrogen2],
                  'BondOrder' => 1);
$WaterMolecule->AddBonds($Bond1, $Bond2);
```

AddAtom

```
$Molecule->AddAtom($Atom);
```

Adds an *Atom* to a *Molecule* and returns *Molecule*.

AddAtoms

```
$Molecule->AddAtoms(@Atoms);
```

Adds *Atoms* to a *Molecule* and returns *Molecule*.

AddBond

```
$Molecule->AddBond($Bond);
```

Adds a *Bond* to a *Molecule* and returns *Molecule*.

AddBonds

```
$Molecule->AddBonds(@Bonds);
```

Adds *Bonds* to a *Molecule* and returns *Molecule*.

AddHydrogens

```
$NumOfHydrogensAdded = $Molecule->AddHydrogens();
```

Adds hydrogens to each atom in a *Molecule* and returns total number of hydrogens added. The current release of MayaChemTools doesn't assign hydrogen positions.

AddPolarHydrogens

```
$NumOfHydrogensAdded = $Molecule->AddPolarHydrogens();
```

Adds hydrogens to each polar atom - N, O, P or S - in a *Molecule* and returns total number of polar hydrogens added. The current release of MayaChemTools doesn't assign hydrogen positions.

ClearRings

```
$Molecule->ClearRings();
```

Deletes all rings associated with *Molecule* and returns *Molecule*.

Copy

```
$MoleculeCopy = $Molecule->Copy();
```

Copies *Molecule* and its associated data using Storable::dclone and returns a new Molecule object.

DeleteAromaticity

```
$Molecule->DeleteAromaticity();
```

Deletes aromatic property associated with all atoms and bonds in a *Molecule* and returns *Molecule*.

DeleteAtom

```
$Molecule->DeleteAtom($Atom);
```

Deletes *Atom* from a *Molecule* and returns *Molecule*.

DeleteAtoms

```
$Molecule->DeleteAtoms(@Atoms);
```

Deletes *Atoms* from a *Molecule* and returns *Molecule*.

DeleteBond

```
$Molecule->DeleteBond($Bond);
```

Deletes *Bond* from a *Molecule* and returns *Molecule*.

DeleteBonds

```
$Molecule->DeleteBonds(@Bonds);
```

Deletes *Bonds* from a *Molecule* and returns *Molecule*.

DeleteHydrogens

```
$NumOfHydrogensDeleted = $Molecule->DeleteHydrogens();
```

Removes hydrogens from each atom in a *Molecule* and returns total number of hydrogens deleted.

DeletePolarHydrogens

```
$NumOfHydrogensDeleted = $Molecule->DeletePolarHydrogens();
```

Removes hydrogens to each polar atom - N, O, P or S - in a *Molecule* and returns total number of polar hydrogens deleted.

DetectAromaticity

```
$Molecule->DetectAromaticity();
```

Associates *Aromatic* property to atoms and bonds involved in aromatic rings or ring systems in a *Molecule* and returns *Molecule*.

This method assumes the ring detection has already been perfomed using DetectRings. And any existing *Aromatic* property associated with atoms and bonds is deleted before performing aromaticity detection.

What is aromaticity? [Ref 124] It's in the code of the implementer, did you say? Agree. The implementation of aromaticity varies widely across different packages [Ref 125]; additionally, the implementation details are not always completely available, and it's not possible to figure out the exact implementation of aromaticity across various packages. Using the publicly available information, however, one can try to reproduce the available results to the extent possible, along with parameterizing all the control parameters used to implement different aromaticity models, and that's exactly what the current release of MayaChemTools does.

The implementation of aromaticity corresponding to various aromaticity models in MayaChemTools package is driven by an external CSV file AromaticityModelsData.csv, which is distributed with the package and is available in lib/data directory. The CSV files contains names of supported aromaticity models, along with various control parameters and their values. This file is loaded and processed during instantiation of Molecule class and data corresponding to specific aromaticity model are used to detect aromaticity for that model. Any new aromaticity model added to the aromaticity data file, using different combinations of values for existing control parameters, would work without any changes to the code; the addition of any new control parameters, however, requires its implementation in the code used to calculate number of pi electrons available towards delocalization in a ring or ring systems.

The current release of MayaChemTools package supports these aromaticity models: MDLAromaticityModel, TriposAromaticityModel, MMFFAromaticityModel, ChemAxonBasicAromaticityModel, ChemAxonGeneralAromaticityModel, DaylightAromaticityModel, MayaChemToolsAromaticityModel.

The current list of control parameters available to detect aromaticity corresponding to different aromaticity models are: AllowHeteroRingAtoms, HeteroRingAtomsList, AllowExocyclicDoubleBonds, AllowHomoNuclearExocyclicDoubleBonds, AllowElectronegativeRingAtomExocyclicDoubleBonds, AllowRingAtomFormalCharge, AllowHeteroRingAtomFormalCharge, MinimumRingSize. The values for these

control parameters are specified in AromaticityModelsData.csv file.

Although definition of aromaticity differs across various aromaticity models, a ring or a ring system containing 4n + 2 pi electrons (Huckel's rule) corresponding to alternate single and double bonds, in general, is considered aromatic.

The available valence free electrons on heterocyclic ring atoms, involved in two single ring bonds, are also allowed to participate in pi electron delocalizaiton for most of the supported aromaticity models.

The presence of exocyclic terminal double bond on ring atoms involved in pi electron delocalization is only allowed for some of the aromaticity models. Additionally, the type atoms involved in exocyclic terminal double bonds may result in making a ring or ring system non-aromatic.

For molecules containing fused rings, each fused ring set is considered as one aromatic system for counting pi electrons to satisfy Huckel's rule; In case of a failure, rings in fused set are treated individually for aromaticity detection. Additionally, non-fused rings are handled on their own during aromaticity detection.

DetectRings

```
$Molecule->DetectRings();
```

Detects rings in a *Molecule* and returns *Molecule*. Ring detection is performed using DetectCycles method avaible in Graph class which in turn uses methods available Graph::CyclesDetection class. Graph::CyclesDetection class implements collapsing path graph [Ref 31] methodology to detect all cycles in a graph.

FormatElementalCompositionInformation

```
$FormattedInfo = $Molecule->FormatElementalCompositionInformation(
                 $ElementsRef, $ElementCompositionRef,
                 [$Precision]);
$FormattedInfo = Molecule::FormatElementalCompositionInformation(
                 $ElementsRef, $ElementCompositionRef,
                 [$Precision]);
```

Using *ElementsRef* and *ElementCompositionRef* arrays referneces containg informatio about elements and their composition, formats elemental composition information and returns a *FormattedInfo* string. Defaule *Precision* value: *2*.

GetAromaticityModel

```
$AromaticityModel = $Molecule->GetAromaticityModel();
```

Returns name of AromaticityModel set for *Molecule* corresponding to AromaticityModel property or default model name of MayaChemToolsAromaticityModel.

GetAllAtomPaths

```
$AtomPathsRef = $Molecule->GetAllAtomPaths([$AllowCycles]);
```

Returns all paths as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to to all possible paths for each atom in molecule with all possible lengths and sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

For molecule without any rings, this method returns the same set of atom paths as GetAtomPaths method.

GetAllAtomPathsStartingAt

```
$AtomPathsRef = $Molecule->GetAllAtomPathsStartingAt($StartAtom,
                [$AllowCycles]);
```

Returns all atom paths starting from *StartAtom* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms atoms correspond to to all possible paths for specified atom in molecule with all possible lengths and sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

For molecule without any rings, this method returns the same set of atom paths as

GetAtomPathsStartingAt method.

GetAllAtomPathsStartingAtWithLength

```
$AtomPathsRef = $Molecule->GetAllAtomPathsStartingAtWithLength(
                $StartAtom, $Length, [$AllowCycles]);
```

Returns all atom paths starting from *StartAtom* with specified *Length*as a reference to an array containing reference to arrays with path Atom objects.

Path atoms atoms correspond to to all possible paths for specified atom in molecule with all possible lengths and sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

For molecule without any rings, this method returns the same set of atom paths as GetAtomPathsStartingAtWithLength method.

GetAllAtomPathsStartingAtWithLengthUpto

```
$AtomPathsRef = $Molecule->GetAllAtomPathsStartingAtWithLengthUpto(
                $StartAtom, $Length, [$AllowCycles]);
```

Returns atom paths starting from *StartAtom* with length up to *Length* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms atoms correspond to all possible paths for specified atom in molecule with length up to a specified length and sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

For molecule without any rings, this method returns the same set of atom paths as *GetAtomPathsStartingAtWithLengthUpto* method.

GetAllAtomPathsWithLength

```
$AtomPathsRef = $Molecule->GetAllAtomPathsWithLength($Length,
                [$AllowCycles]);
```

Returns all atom paths with specified *Length* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to to all possible paths for each atom in molecule with length up to a specified length and sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

For molecule without any rings, this method returns the same set of atom paths as as *GetAtomPathsWithLength* method.

GetAllAtomPathsWithLengthUpto

```
$AtomPathsRef = $Molecule->GetAllAtomPathsWithLengthUpto($Length,
                [$AllowCycles]);
```

Returns all atom paths with length up to *Length* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to to all possible paths for each atom in molecule with length up to a specified length and sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

For molecule without any rings, this method returns the same set of atom paths as as *GetAtomPathsWithLengthUpto* method.

GetAromaticRings

```
@AtomaticRings = $Molecule->GetAromaticRings();
```

Returns aromatic rings as an array containing references to arrays of ring *Atom* objects in a *Molecule*.

GetAtomNeighborhoods

```
@Neighborhoods = $Molecule->GetAtomNeighborhoods($StartAtom);
```

Returns atom neighborhoods around a *StartAtom* as an array containing references to arrays with

neighborhood *Atom* objects at possible radii.

GetAtomNeighborhoodsWithRadiusUpto

```
@Neighborhoods = $Molecule->GetAtomNeighborhoodsWithRadiusUpto($StartAtom,
                 $Radius);
```

Returns atom neighborhoods around a *StartAtom* as an array containing references to arrays with neighborhood *Atom* objects up to *Radius*.

GetAtomNeighborhoodsWithSuccessorAtoms

```
@Neighborhoods = $Molecule->GetAtomNeighborhoodsWithSuccessorAtoms(
                 $StartAtom);
```

Returns atom neighborhood around a specified *StartAtom*, along with their successor connected atoms, collected at all radii as an array containing references to arrays with first value corresponding to neighborhood atom at a specific radius and second value as reference to an array containing its successor connected atoms.

For a neighborhood atom at each radius level, the successor connected atoms correspond to the neighborhood atoms at the next radius level. Consequently, the neighborhood atoms at the last radius level don't contain any successor atoms which fall outside the range of specified radius.

GetAtomNeighborhoodsWithSuccessorAtomsAndRadiusUpto

```
@Neighborhoods = $Molecule->GetAtomNeighborhoodsWithSuccessorAtomsAndRadiusUpto(
                 $StartAtom, $Radius);
```

Returns atom neighborhood around a specified *StartAtom*, along with their successor connected atoms, collected upto specified *Radiud* as an array containing references to arrays with first value corresponding to neighborhood atom at a specific radius and second value as reference to an array containing its successor connected atoms.

For a neighborhood atom at each radius level, the successor connected atoms correspond to the neighborhood atoms at the next radius level. Consequently, the neighborhood atoms at the last radius level don't contain any successor atoms which fall outside the range of specified radius.

GetAtomPathBonds

```
$Return = $Molecule->GetAtomPathBonds(@PathAtoms);
```

Returns an array containing Bond objects corresponding to successive pair of atoms in *PathAtoms*

GetAtomPaths

```
$AtomPathsRef = $Molecule->GetAtomPaths([$AllowCycles]);
```

Returns all paths as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to to all possible paths for each atom in molecule with all possible lengths and no sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

GetAtomPathsBetween

```
$AtomPathsRef = $Molecule->GetAtomPathsBetween($StartAtom, $EndAtom);
```

Returns all paths as between *StartAtom* and *EndAtom* as a reference to an array containing reference to arrays with path Atom objects.

For molecules with rings, atom paths array contains may contain two paths.

GetAtomPathsStartingAt

```
$AtomPathsRef = $Molecule->GetAtomPathsStartingAt($StartAtom, [$AllowCycles]);
```

Returns paths starting at *StartAtom* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to all possible paths for specified atom in molecule with all possible lengths and no sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is

terminated at an atom completing the ring.

GetAtomPathsStartingAtWithLength

```
$AtomPathsRef = $Molecule->GetAtomPathsStartingAtWithLength($StartAtom,
        $Length, [$AllowCycles]);
```

Returns paths starting at *StartAtom* with length *Length* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to all possible paths for specified atom in molecule with length upto a specified length and no sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

GetAtomPathsStartingAtWithLengthUpto

```
$AtomPathsRef = $Molecule->GetAtomPathsStartingAtWithLengthUpto($StartAtom,
        $Length, [$AllowCycles]);
```

Returns paths starting at *StartAtom* with length up to *Length* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to all possible paths for specified atom in molecule with length upto a specified length and no sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

GetAtomPathsWithLength

```
$AtomPathsRef = $Molecule->GetAtomPathsWithLength($Length, [$AllowCycles]);
```

Returns all paths with specified *Length* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to all possible paths for each atom in molecule with length upto a specified length and no sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

GetAtomPathsWithLengthUpto

```
$AtomPathsRef = $Molecule->GetAtomPathsWithLengthUpto($Length, [$AllowCycles]);
```

Returns all paths with length up to *Length* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to all possible paths for each atom in molecule with length upto a specified length and no sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

GetAtoms

```
@AllAtoms = $Molecule->GetAtoms();
@PolarAtoms = $Molecule->GetAtoms('IsPolarAtom');

$NegateMethodResult = 1;
@NonHydrogenAtoms = $Molecule->GetAtoms('IsHydrogenAtom',
                    $NegateMethodResult);

$AtomsCount = $Molecule->GetAtoms();
```

Returns an array of *Atoms* in a *Molecule*. In scalar context, it returns number of atoms. Additionally, Atoms array can be filtered by any user specifiable valid Atom class method and the result of the Atom class method used to filter the atoms can also be negated by an optional negate results flag as third parameter.

GetBonds

```
@Bonds = $Molecule->GetBonds();
$BondsCount = $Molecule->GetBonds();
```

Returns an array of *Bonds* in a *Molecule*. In scalar context, it returns number of bonds.

GetCharge

```
$Charge = $Molecule->GetCharge();
```

Returns net charge on a *Molecule* using one of the following two methods: explicitly set Charge property or sum of partial atomic charges on each atom.

GetConnectedComponents

```
@ConnectedComponents = $Molecule->GetConnectedComponents();
```

Returns a reference to an array containing *Molecule* objects corresponding to connected components sorted in decreasing order of component size in a *Molecule*.

GetConnectedComponentsAtoms

```
@ConnectedComponentsAtoms =
  $Molecule->GetConnectedComponentsAtoms();
```

Returns an array containing references to arrays with *Atom* objects corresponding to atoms of connected components sorted in order of component decreasing size in a *Molecule*.

GetDimensionality

```
$Dimensionality = $Molecule->GetDimensionality();
```

Returns *Dimensionality* of a *Molecule* corresponding to explicitly set *Dimensionality* property value or by processing atomic.

The *Dimensionality* value from atomic coordinates is calculated as follows:

```
3D - Three dimensional: One of X, Y or Z coordinate is non-zero
2D - Two dimensional: One of X or Y coordinate is non-zero; All Z
     coordinates are zero
0D - Zero dimensional: All atomic coordinates are zero
```

GetElementalComposition

```
($ElementsRef, $CompositionRef) =
  $Molecule->GetElementalComposition([$IncludeMissingHydrogens]);
```

Calculates elemental composition and returns references to arrays containing elements and their percent composition in a *Molecule*. By default, missing hydrogens are included during the calculation.

GetElementsAndNonElements

```
($ElementsRef, $NonElementsRef) =
 $Molecule->GetElementsAndNonElements([$IncludeMissingHydrogens]);
```

Counts elements and non-elements in a *Molecule* and returns references to hashes containing element and non-element as hash keys with values corresponding to their count. By default, missing hydrogens are not added to the element hash.

GetExactMass

```
$ExactMass = $Molecule->GetExactMass();
```

Returns exact mass of a *Molecule* corresponding to sum of exact masses of all the atoms.

GetFormalCharge

```
$FormalCharge = $Molecule->GetFormalCharge();
```

Returns net formal charge on a *Molecule* using one of the following two methods: explicitly set FormalCharge property or sum of formal charges on each atom.

FormalCharge is different from Charge property of the molecule which corresponds to sum of partial atomic charges explicitly set for each atom using a specific methodology.

GetFreeRadicalElectrons

```
$FreeRadicalElectrons = $Molecule->GetFreeRadicalElectrons();
```

Returns total number of free radical electrons available in a *Molecule* using one of the following two methods: explicitly set FreeRadicalElectrons property or sum of available free radical electrons on each atom.

GetFusedAndNonFusedRings

```
($FusedRingSetRef, $NonFusedRingsRef) =
   $Molecule->GetFusedAndNonFusedRings();
```

Returns references to array of fused ring sets and non-fused rings in a *Molecule*. Fused ring sets array reference contains refernces to arrays of rings corresponding to ring *Atom* objects; Non-fused rings array reference contains references to arrays of ring *Atom* objects.

GetLargestConnectedComponent

```
$ComponentMolecule = $Molecule->GetLargestConnectedComponent();
```

Returns a reference to Molecule object corresponding to a largest connected component in a *Molecule*.

GetLargestConnectedComponentAtoms

```
@ComponentAtoms = $Molecule->GetLargestConnectedComponentAtoms();
```

Returns a reference to an array of Atom objects corresponding to a largest connected component in a *Molecule*.

GetLargestRing

```
@RingAtoms = $Molecule->GetLargestRing();
```

Returns an array of *Atoms* objects corresponding to a largest ring in a *Molecule*.

GetMolecularFormula

```
$FormulaString = $Molecule->GetMolecularFormula(
                 [$IncludeMissingHydrogens,
                 $IncludeNonElements]);
```

Returns molecular formula of a *Molecule* by collecting information about all atoms in the molecule and composing the formula using Hills ordering system:

- C shows up first and H follows assuming C is present.
- All other standard elements are sorted alphanumerically.
- All other non-stanard atom symbols are also sorted alphanumerically and follow standard elements.

Notes:

- By default, missing hydrogens and nonelements are also included.
- Elements for disconnected fragments are combined into the same formula.
- Formal charge is also used during compoisiton of molecular formula.

GetMolecularWeight

```
$MolWeight = $Molecule->GetMolecularWeight();
```

Returns molecular weight of a *Molecule* corresponding to sum of atomic weights of all the atoms.

GetNumOfAromaticRings

```
$NumOfAromaticRings = $Molecule->GetNumOfAromaticRings();
```

Returns number of aromatic rings in a *Molecule*.

GetNumOfAtoms

```
$NumOfAtoms = $Molecule->GetNumOfAtoms();
```

Returns number of atoms in a *Molecule*.

GetNumOfBonds

```
$NumOfBonds = $Molecule->GetNumOfBonds();
```

Returns number of bonds in a *Molecule*.

GetNumOfConnectedComponents

```
$NumOfComponents = $Molecule->GetNumOfConnectedComponents();
```

Returns number of connected components in a *Molecule*.

GetNumOfElementsAndNonElements

```
($NumOfElements, $NumOfNonElements) = $Molecule->
                            GetNumOfElementsAndNonElements();
($NumOfElements, $NumOfNonElements) = $Molecule->
                GetNumOfElementsAndNonElements($IncludeMissingHydrogens);
```

Returns number of elements and non-elements in a *Molecule*. By default, missing hydrogens are not added to element count.

GetNumOfHeavyAtoms

```
$NumOfHeavyAtoms = $Molecule->GetNumOfHeavyAtoms();
```

Returns number of heavy atoms, non-hydrogen atoms, in a *Molecule*.

GetNumOfHydrogenAtoms

```
$NumOfHydrogenAtoms = $Molecule->GetNumOfHydrogenAtoms();
```

Returns number of hydrogen atoms in a *Molecule*.

GetNumOfMissingHydrogenAtoms

```
$NumOfMissingHydrogenAtoms = $Molecule->GetNumOfMissingHydrogenAtoms();
```

Returns number of hydrogen atoms in a *Molecule*.

GetNumOfNonHydrogenAtoms

```
$NumOfNonHydrogenAtoms = $Molecule->GetNumOfNonHydrogenAtoms();
```

Returns number of non-hydrogen atoms in a *Molecule*.

GetNumOfRings

```
$RingCount = $Molecule->GetNumOfRings();
```

Returns number of rings in a *Molecule*.

GetNumOfRingsWithEvenSize

```
$RingCount = $Molecule->GetNumOfRingsWithEvenSize();
```

Returns number of rings with even size in a *Molecule*.

GetNumOfRingsWithOddSize

```
$RingCount = $Molecule->GetNumOfRingsWithOddSize();
```

Returns number of rings with odd size in a *Molecule*.

GetNumOfRingsWithSize

```
$RingCount = $Molecule->GetNumOfRingsWithSize($Size);
```

Returns number of rings with *Size* in a *Molecule*.

GetNumOfRingsWithSizeGreaterThan

```
$RingCount = $Molecule->GetNumOfRingsWithSizeGreaterThan($Size);
```

Returns number of rings with size greater than *Size* in a *Molecule*.

GetNumOfRingsWithSizeLessThan

```
$RingCount = $Molecule->GetNumOfRingsWithSizeLessThan($Size);
```

Returns number of rings with size less than *Size* in a *Molecule*.

GetRingBonds

```
@RingBonds = $Molecule->GetRingBonds(@RingAtoms);
```

Returns an array of ring `Bond` objects correponding to an array of ring *Atoms* in a *Molecule*.

GetRingBondsFromRings

```
@RingBondsSets = $Molecule->GetRingBondsFromRings(@RingAtomsSets);
```

Returns an array containing references to arrays of ring `Bond` objects for rings specified in an array of references to ring *Atom* objects.

GetRings

```
@Rings = $Molecule->GetRings();
```

Returns rings as an array containing references to arrays of ring *Atom* objects in a *Molecule*.

GetRingsWithEvenSize

```
@Rings = $Molecule->GetRingsWithEvenSize();
```

Returns even size rings as an array containing references to arrays of ring *Atom* objects in a *Molecule*.

GetRingsWithOddSize

```
@Rings = $Molecule->GetRingsWithOddSize();
```

Returns odd size rings as an array containing references to arrays of ring *Atom* objects in a *Molecule*.

GetRingsWithSize

```
@Rings = $Molecule->GetRingsWithSize($Size);
```

Returns rings with *Size* as an array containing references to arrays of ring *Atom* objects in a *Molecule*.

GetRingsWithSizeGreaterThan

```
@Rings = $Molecule->GetRingsWithSizeGreaterThan($Size);
```

Returns rings with size greater than *Size* as an array containing references to arrays of ring *Atom* objects in a *Molecule*.

GetRingsWithSizeLessThan

```
@Rings = $Molecule->GetRingsWithSizeLessThan($Size);
```

Returns rings with size less than *Size* as an array containing references to arrays of ring *Atom* objects in a *Molecule*.

GetSizeOfLargestRing

```
$Size = $Molecule->GetSizeOfLargestRing();
```

Returns size of the largest ring in a *Molecule*.

GetSizeOfSmallestRing

```
$Size = $Molecule->GetSizeOfSmallestRing();
```

Returns size of the smalles ring in a *Molecule*.

GetSmallestRing

```
@RingAtoms = $Molecule->GetSmallestRing();
```

Returns an array containing *Atom* objects corresponding to the smallest ring in a *Molecule*.

GetSpinMultiplicity

```
$SpinMultiplicity = $Molecule->GetSpinMultiplicity();
```

Returns net spin multiplicity of a *Molecule* using one of the following two methods: explicitly set SpinMultiplicity property or sum of spin multiplicity on each atom.

GetSupportedAromaticityModels

```
@SupportedModels = $Molecule->GetSupportedAromaticityModels();
```

Returns an array containing a list of supported aromaticity models.

GetValenceModel

```
$ValenceModel = $Molecule->GetValenceModel();
```

Returns valence model for *Molecule* using one of the following two methods: explicitly set ValenceModel property or defaul value of *InternalValenceModel*.

GetTopologicallySortedAtoms

```
@SortedAtoms = $Molecule->GetTopologicallySortedAtoms([$StartAtom]);
```

Returns an array of topologically sorted *Atom* objects starting from *StartAtom* or an arbitrary atom in a *Molecule*.

HasAromaticRings

```
$Status = $Molecule->HasAromaticRings();
```

Returns 1 or 0 based on whether any aromatic ring is present in a *Molecule*.

HasAromaticAtomsInRings

```
$Status = $Molecule->HasAromaticAtomsInRings();
```

Returns 1 or 0 based on whether any aromatic ring atom is present in a *Molecule*.

HasAromaticAtomsNotInRings

```
$Status = $Molecule->HasAromaticAtomsNotInRings();
```

Returns 1 or 0 based on whether any non-ring atom is marked aromatic in a *Molecule*.

HasAtom

```
$Status = $Molecule->HasAtom($Atom);
```

Returns 1 or 0 based on whether *Atom* is present in a *Molecule*.

HasBond

```
$Status = $Molecule->HasBond($Bond);
```

Returns 1 or 0 based on whether *Bond* is present in a *Molecule*.

HasFusedRings

```
$Status = $Molecule->HasFusedRings();
```

Returns 1 or 0 based on whether any fused rings set is present in a *Molecule*.

HasNoRings

```
$Status = $Molecule->HasNoRings();
```

Returns 0 or 1 based on whether any ring is present in a *Molecule*.

HasOnlyOneRing

```
$Status = $Molecule->HasOnlyOneRing();
```

Returns 1 or 0 based on whether only one ring is present in a *Molecule*.

HasRings

```
$Status = $Molecule->HasRings();
```

Returns 1 or 0 based on whether rings are present in a *Molecule*.

IsAromatic

```
$Status = $Molecule->IsAromatic();
```

Returns 1 or 0 based on whether *Molecule* is aromatic.

IsMolecule

```
$Status = Molecule::IsMolecule();
```

Returns 1 or 0 based on whether *Object* is a Molecule object.

IsRingAromatic

```
$Status = $Molecule->IsRingAromatic(@RingAtoms);
```

Returns 1 or 0 based on whether all *RingAtoms* are aromatic.

IsSupportedAromaticityModel

```
$Status = $Molecule->IsSupportedAromaticityModel($AromaticityModel);
$Status = Molecule::IsSupportedAromaticityModel($AromaticityModel);
```

Returns 1 or 0 based on whether specified *AromaticityModel* is supported.

IsTwoDimensional

```
$Status = $Molecule->IsTwoDimensional();
```

Returns 1 or 0 based on whether any atom in *Molecule* has a non-zero value for X or Y coordinate and all atoms have zero value for Z coordinates.

IsThreeDimensional

```
$Status = $Molecule->IsThreeDimensional();
```

Returns 1 or 0 based on whether any atom in *Molecule* has a non-zero value for Z coordinate.

KeepLargestComponent

```
$Molecule->KeepLargestComponent();
```

Deletes atoms corresponding to all other connected components Except for the largest connected component in a *Molecule* and returns *Molecule*.

KekulizeAromaticAtoms

```
$Status = $Molecule->KekulizeAromaticAtoms();
```

Kekulize marked ring and non-ring aromatic atoms in a molecule and return 1 or 1 based on whether the kekulization succeeded.

NewAtom

```
$NewAtom = $Molecule->NewAtom(%AtomPropertyNamesAndValues);
```

Creates a new atom using *AtomPropertyNamesAndValues*, add its to *Molecule*, and returns new Atom object.

NewBond

```
$NewBond = $Molecule->NewBond(%BondPropertyNamesAndValues);
```

Creates a new bond using *AtomPropertyNamesAndValues*, add its to *Molecule*, and returns new Bond object.

SetActiveRings

```
$Molecule->SetActiveRings($RingsType);
```

Sets up type of detected ring sets to use during all ring related methods and returns *Molecule*. Possible *RingType* values: *Independent or All*. By default, *Independent* ring set is used during all ring methods.

SetAromaticityModel

```
$Molecule = $Molecule->SetAromaticityModel($AromaticityModel);
```

Sets up *AromaticityModel* property value for *Molecule* and retrurns *Molecule*.

SetValenceModel

```
$Molecule = $Molecule->SetValenceModel(ValenceModel);
```

Sets up *ValenceModel* property value for *Molecule* and retrurns *Molecule*.

StringifyMolecule

```
$MoleculeString = $Molecule->StringifyMolecule();
```

Returns a string containing information about *Molecule* object

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

Atom.pm, Bond.pm, MoleculeFileIO.pm, MolecularFormula.pm

COPYRIGHT