

NAME

Atom

SYNOPSIS

use Atom;

DESCRIPTION

Atom class provides the following methods:

new, AddHydrogens, Copy, DeleteAtom, DeleteHydrogens, GetAtomicWeight, GetBondToAtom, GetBonds, GetBondsToHeavyAtoms, GetBondsToHydrogenAtoms, GetBondsToNonHydrogenAtoms, GetExactMass, GetExplicitHydrogens, GetHeavyAtomNeighbors, GetHighestCommonValence, GetHydrogenAtomNeighbors, GetImplicitHydrogens, GetImplicitValence, GetLargestRing, GetLowestCommonValence, GetMassNumber, GetNeighbors, GetNonHydrogenAtomNeighbors, GetNumOfBonds, GetNumOfBondsToHeavyAtoms, GetNumOfBondsToHydrogenAtoms, GetNumOfBondsToNonHydrogenAtoms, GetNumOfHeavyAtomNeighbors, GetNumOfHydrogenAtomNeighbors, GetNumOfMissingHydrogens, GetNumOfNeighbors, GetNumOfNonHydrogenAtomNeighbors, GetNumOfRings, GetNumOfRingsWithEvenSize, GetNumOfRingsWithOddSize, GetNumOfRingsWithSize, GetNumOfRingsWithSizeGreaterThan, GetNumOfRingsWithSizeLessThan, GetRings, GetRingsWithEvenSize, GetRingsWithOddSize, GetRingsWithSize, GetRingsWithSizeGreaterThan, GetRingsWithSizeLessThan, GetSizeOfLargestRing, GetSizeOfSmallestRing, GetSmallestRing, GetSumOfBondOrders, GetSumOfBondOrdersToHeavyAtoms, GetSumOfBondOrdersToHydrogenAtoms, GetSumOfBondOrdersToNonHydrogenAtoms, GetValence, GetValenceElectrons, GetValenceFreeElectrons, GetX, GetXYZ, GetXYZVector, GetY, GetZ, IsAromatic, IsBromine, IsCarbon, IsChlorine, IsFluorine, IsHetroAtom, IsHydrogen, IsInRing, IsInRingOfSize, IsIodine, IsNitrogen, IsNotInRing, IsOnlyInOneRing, IsOxygen, IsPhosphorus, IsPolarAtom, IsPolarHydrogen, IsStereoCenter, IsSulfur, SetAtomSymbol, SetAtomicNumber, SetID, SetMassNumber, SetStereoCenter, SetStereochemistry, SetX, SetXYZ, SetY, SetZ, StringifyAtom

Atom class is derived from ObjectProperty base class which provides methods not explicitly defined in Atom 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

```
$NewAtom = new Atom([%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, following properties are initialized:

```
ID = SequentialObjectID
Name = "Atom <SequentialObjectID>"
AtomSymbol = ""
AtomicNumber = 0
XYZ = ZeroVector
```

Except for *ID* property, all other default properties and other additional properties can be set during invocation of this method.

Examples:

```
$Atom = new Atom();
$CarbonAtom = new Atom(AtomSymbol' => 'C', 'XYZ' => (0.0, 1.0,
0.0));
$OxygenAtom = new Atom('AtomName' => 'Oxygen', AtomSymbol' => 'O',
'XYZ' => (1.0, 1.0, 1.0));
```

AddHydrogens

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

Adds hydrogens to an Atom present in a Molecule object and returns the number of added hydrogens. The current release of MayaChemTools doesn't assign hydrogen positions.

Copy

```
$AtomCopy = $Atom->Copy();
```

Copy *Atom* and its associated data using Storable::dclone and return a new Atom object

DeleteAtom

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

Delete *Atom* from a molecule

DeleteHydrogens

```
$NumOfHydrogensDeleted = $Atom->AddHydrogens();
```

Delete hydrogens from an Atom present in a Molecule object and returns the number of deleted hydrogens

GetAtomicWeight

```
$Return = $Atom->GetAtomicWeight();
```

Returns atomic weight of an Atom which corresponds to either explicitly set *AtomicWeight* atom property or atomic weight of the corresponding element in the periodic table available by PeriodicTable module

GetBondToAtom

```
$Bond = $Atom->GetBondToAtom($OtherAtom);
```

Returns a Bond object corresponding to bond between *Atom* and *OtherAtom* in a molecule

GetBonds

```
@Bonds = $Atom->GetBonds();
```

Returns an array of Bond objects corresponding to all bonds from *Atom* to other bonded atoms in a molecule

GetBondsToHeavyAtoms

```
@Bonds = $Atom->GetBondsToHeavyAtoms();
```

Returns an array of Bond objects corresponding to bonds from *Atom* to other bonded non-hydrogen atoms in a molecule

GetBondsToHydrogenAtoms

```
@Bonds = $Atom->GetBondsToHydrogenAtoms();
```

Returns an array of Bond objects corresponding to bonds from *Atom* to any other hydrogen atom in a molecule

GetBondsToNonHydrogenAtoms

```
@Bonds = $Atom->GetBondsToNonHydrogenAtoms();
```

Returns an array of Bond objects corresponding to bonds from *Atom* to other bonded non-hydrogen atoms in a molecule

GetExactMass

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

Returns exact mass of an *Atom* which correspond to one of these three values: explicitly set *ExactMass* property; mass of natural isotope for an explicitly set value of *MassNumber*; most abundant natural isotope mass for *Atom* with valid atomic number value available by PeriodicTable module.

GetExplicitHydrogens

```
$NumOfExplicitHydrogens = $Atom->GetExplicitHydrogens();
```

Returns number of hydrogens explicitly bonded to an *Atom* in a molecule

GetHeavyAtomNeighbors

```
$NumOfHeavyAtoms = $Atom->GetHeavyAtomNeighbors();  
@HeavyAtoms = $Atom->GetHeavyAtomNeighbors();
```

Return number of heavy atoms or an array of Atom objects corresponding to heavy atoms bonded to an *Atom* in a molecule

GetHighestCommonValence

```
$HighestCommonValence = $Atom->GetHighestCommonValence();
```

Returns highest common valence of an *Atom* which corresponds to either explicitly set *HighestCommonValence* atom property or highest common valence of the corresponding element in the periodic table available by PeriodicTable module.

GetHydrogenAtomNeighbors

```
$NumOfHydrogenAtomNeighbors = $Atom->GetHydrogenAtomNeighbors();  
@HydrogenAtomNeighbors = $Atom->GetHydrogenAtomNeighbors();
```

Return number of hydrogen atoms or an array of *Atom* objects corresponding to hydrogen atoms bonded to an *Atom* in a molecule

GetImplicitHydrogens

```
$NumOfImplicitHydrogens = $Atom->GetImplicitHydrogens();
```

Returns number of implicit hydrogens for an *Atom* in a molecule. This value either corresponds to explicitly set *ImplicitHydrogens* atom property or calculated as the difference between the value of implicit valence and sum of bond orders to explicit non-hydrogen atoms.

GetImplicitValence

```
$ImplicitValence = $Atom->GetImplicitValence();
```

Returns implicit valence of an *Atom* in a molecule which corresponds to either explicitly set *ImplicitValence* atom property or calculated as described below.

For atoms with only one possible value of common valence available by PeriodicTable module, *ImplicitValence* corresponds to: $\text{CommonValence} + \text{FormalCharge} + \text{SpinMultiplicityCorrection}$.

For atoms with multiple values of common valence available by PeriodicTable module and formal charge, *ImplicitValence* corresponds to: $\text{HighestCommonValence} + \text{FormalCharge} + \text{SpinMultiplicityCorrection}$.

For atoms with multiple values of common valence available by PeriodicTable module and no formal charge, *ImplicitValence* corresponds to: First available valence higher than sum of bond orders to non-hydrogen atoms + *SpinMultiplicityCorrection*.

Notes:

For atoms with explicit assignment of *SpinMultiplicity* atom property values

```

Singlet - two unpaired electrons corresponding to one spin state
Doublet - free radical; an unpaired electron corresponding to two
          spin states
Triplet - two unpaired electrons corresponding to three spin states
          (divalent carbon atoms: carbenes)

```

SpinMultiplicityCorrection is calculated as follows:

```

Doublet: -1 (one valence electron not available for bonding)
Singlet: -2 (two valence electrons not available for bonding)
Triplet: -2 (two valence electrons not available for bonding)

```

GetLargestRing

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

Returns an array of ring *Atom* objects corresponding to the largest ring containing *Atom* in a molecule

GetLowestCommonValence

```
$LowestCommonValence = $Atom->GetLowestCommonValence();
```

Returns lowest common valence of an *Atom* which corresponds to either explicitly set *LowestCommonValence* atom property or highest common valence of the corresponding element in the periodic table available by PeriodicTable module

GetMassNumber

```
$MassNumber = $Atom->GetMassNumber();
```

Returns atomic weight of an *Atom* which corresponds to either explicitly set *MassNumber* atom property or mass number of the most abundant natural isotope of the corresponding element in the periodic table available by PeriodicTable module

GetNeighbors

```
$NumOfNeighbors = $Atom->GetNeighbors();
@Neighbors = $Atom->GetNeighbors();
```

Returns number of neighbor atoms or an array of *Atom* objects corresponding to all atoms bonded to an *Atom* in a molecule

GetNonHydrogenAtomNeighbors

```
$NumOfNeighbors = $Atom->GetNonHydrogenAtomNeighbors();
@Neighbors = $Atom->GetNonHydrogenAtomNeighbors();
```

Returns number of non-hydrogen atoms or an array of *Atom* objects corresponding to non-hydrogen atoms bonded to an *Atom* in a molecule

GetNumOfBonds

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

Returns number of bonds from *Atom* to other atoms in a molecule

GetNumOfBondsToHeavyAtoms

```
$NumOfBondsToHeavyAtoms = $Atom->GetNumOfBondsToHeavyAtoms();
```

Returns number of bonds from *Atom* to other heavy atoms in a molecule

GetNumOfBondsToHydrogenAtoms

```
$NumOfBonds = $Atom->GetNumOfBondsToHydrogenAtoms();
```

Returns number of bonds from *Atom* to other hydrogen atoms in a molecule

GetNumOfBondsToNonHydrogenAtoms

```
$NumOfBonds = $Atom->GetNumOfBondsToNonHydrogenAtoms();
```

Returns number of bonds from *Atom* to other non-hydrogen atoms in a molecule

GetNumOfHeavyAtomNeighbors

```
$NumOfNeighbors = $Atom->GetNumOfHeavyAtomNeighbors();
```

Returns number heavy atom neighbors for an *Atom* in a molecule

GetNumOfHydrogenAtomNeighbors

```
$NumOfNeighbors = $Atom->GetNumOfHydrogenAtomNeighbors();
```

Returns number hydrogens atom neighbors for an *Atom* in a molecule

GetNumOfMissingHydrogens

```
$NumOfMissingHydrogens = $Atom->GetNumOfMissingHydrogens();
```

Returns number of missing hydrogens for an *Atom* in a molecule corresponding to the difference between number of implicit and explicit hydrogens.

GetNumOfNeighbors

```
$NumOfNeighbors = $Atom->GetNumOfNeighbors();
```

Returns number atom neighbors for an *Atom* in a molecule

GetNumOfNonHydrogenAtomNeighbors

```
$NumNeighbors = $This->GetNumOfNonHydrogenAtomNeighbors();
```

Returns number non-hydrogens atom neighbors for an *Atom* in a molecule

GetNumOfRings

```
$NumOfRings = $Atom->GetNumOfRings();
```

Returns number of rings containing *Atom* in a molecule

GetNumOfRingsWithEvenSize

```
$NumOfRings = $Atom->GetNumOfRingsWithEvenSize();
```

Returns number of rings with even size containing *Atom* in a molecule

GetNumOfRingsWithOddSize

```
$NumOfRings = $Atom->GetNumOfRingsWithOddSize();
```

Returns number of rings with odd size containing *Atom* in a molecule

GetNumOfRingsWithSize

```
$NumOfRings = $Atom->GetNumOfRingsWithSize($RingSize);
```

Returns number of rings with specific *RingSize* containing *Atom* in a molecule

GetNumOfRingsWithSizeGreaterThan

```
$NumOfRings = $Atom->GetNumOfRingsWithSizeGreaterThan($RingSize);
```

Returns number of rings with size greater than specific *RingSize* containing *Atom* in a molecule

GetNumOfRingsWithSizeLessThan

```
$NumOfRings = $Atom->GetNumOfRingsWithSizeLessThan($RingSize);
```

Returns number of rings with size less than specific *RingSize* containing *Atom* in a molecule

GetRings

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

Returns an array of references to arrays containing ring atoms corresponding to all rings containing *Atom* in a molecule

GetRingsWithEvenSize

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

Returns an array of references to arrays containing ring atoms corresponding to all rings with even size containing *Atom* in a molecule

GetRingsWithOddSize

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

Returns an array of references to arrays containing ring atoms corresponding to all rings with odd size containing *Atom* in a molecule

GetRingsWithSize

```
@Rings = $Atom->GetRingsWithSize($RingSize);
```

Returns an array of references to arrays containing ring atoms corresponding to all rings with specific *RingSize* containing *Atom* in a molecule

GetRingsWithSizeGreaterThan

```
@Rings = $Atom->GetRingsWithSizeGreaterThan($RingSize);
```

Returns an array of references to arrays containing ring atoms corresponding to all rings with size greater than specific *RingSize* containing *Atom* in a molecule

GetRingsWithSizeLessThan

```
@Rings = $Atom->GetRingsWithSizeLessThan($RingSize);
```

Returns an array of references to arrays containing ring atoms corresponding to all rings with size less than specific *RingSize* containing *Atom* in a molecule

GetSizeOfLargestRing

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

Returns size of the largest ring containing *Atom* in a molecule

GetSizeOfSmallestRing

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

Returns size of the smallest ring containing *Atom* in a molecule

GetSmallestRing

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

Returns an array of ring *Atom* objects corresponding to the largest ring containing *Atom* in a molecule

GetSumOfBondOrders

```
$SumBondOrders = $Atom->GetSumOfBondOrders();
```

Returns sum of bond orders corresponding to all atoms bonded to an *Atom* in a molecule

GetSumOfBondOrdersToHeavyAtoms

```
$SumBondOrders = $Atom->GetSumOfBondOrdersToHeavyAtoms();
```

Returns sum of bond orders corresponding to all heavy atoms bonded to an *Atom* in a molecule

GetSumOfBondOrdersToHydrogenAtoms

```
$SumBondOrders = $Atom->GetSumOfBondOrdersToHydrogenAtoms();
```

Returns sum of bond orders corresponding to all hydrogen atoms bonded to an *Atom* in a molecule

GetSumOfBondOrdersToNonHydrogenAtoms

```
$SumBondOrders = $Atom->GetSumOfBondOrdersToNonHydrogenAtoms();
```

Returns sum of bond orders corresponding to all non-hydrogen atoms bonded to an *Atom* in a molecule

GetValence

```
$Valence = $Atom->GetValence();
```

Returns valence of an *Atom* in a molecule. Valence corresponds to number of electrons used by an atom in bonding:

$$\text{Valence} = \text{ValenceElectrons} - \text{ValenceFreeElectrons} = \text{BondingElectrons}$$

Single, double and triple bonds with bond orders of 1, 2, and 3 correspond to contribution of 1, 2, and 3 bonding electrons. So:

$$\text{Valence} = \text{SumOfBondOrders} + \text{FormalCharge}$$

where positive and negative values of FormalCharge increase and decrease the number of bonding electrons, respectively.

Notes:

- . For neutral molecules, valence and sum of bond orders are equal.
- . For molecules containing only single bonds, SumOfBondOrders and NumOfBonds are equal.

GetValenceElectrons

```
$ValenceElectrons = $Atom->GetValenceElectrons();
```

Returns valence electrons for an Atom which corresponds to either explicitly set *ValenceElectrons* atom property or valence electrons for the corresponding element in the periodic table available by PeriodicTable module

GetValenceFreeElectrons

```
$ValenceFreeElectrons = $Atom->GetValenceFreeElectrons();
```

Returns valence free electrons for an Atom in a molecule. It corresponds to:

$$\text{ValenceElectrons} - \text{SumOfBondOrders} - \text{FormalCharge}$$

Examples:

```
NH3: ValenceFreeElectrons = 5 - 3 - 0 = 2
NH4+: ValenceFreeElectrons = 5 - 4 - 1 = 0
C(=O)O- : ValenceFreeElectrons on O- = 6 - 1 + 1 = 6
C(=O)O- : ValenceFreeElectrons on =O = 6 - 2 - 0 = 4
```

GetX

```
$X = $Atom->GetX();
```

Returns value of X-coordinate for an *Atom*

GetXYZ

```
@XYZ = $Atom->GetXYZ();
$XYZRef = $Atom->GetXYZ();
```

Returns an array or a reference to an array containing values for *Atom* coordinates

GetXYZVector

```
$XYZVector = $Atom->GetXYZVector();
```

Returns a *Vector* object containing values for *Atom* coordinates

GetY

```
$Y = $Atom->GetY();
```

Returns value of Y-coordinate for an *Atom*

GetZ

```
$Z = $Atom->GetZ();
```

Returns value of Z-coordinate for an *Atom*

IsAromatic

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

Returns 1 or 0 based on whether it's an aromatic *Atom*

IsBromine

```
$Status = $Atom->IsBromine();
```

Returns 1 or 0 based on whether it's a bromine *Atom*

IsCarbon

```
$Status = $Atom->IsCarbon();
```

Returns 1 or 0 based on whether it's a carbon *Atom*

IsChlorine

```
$Status = $Atom->IsChlorine();
```

Returns 1 or 0 based on whether it's a chlorine *Atom*

IsFluorine

```
$Status = $Atom->IsFluorine();
```

Returns 1 or 0 based on whether it's a fluorine *Atom*

IsHetroAtom

```
$Status = $Atom->IsHetroAtom();
```

Returns 0 or 1 based on whether it's a hetro *Atom*. Following atoms are considered hetro atoms: N, O, F, P, S, Cl, Br, I

IsHydrogen

```
$Status = $Atom->IsHydrogen();
```

Returns 1 or 0 based on whether it's a hydrogen *Atom*

IsInRing

```
$Status = $Atom->IsInRing();
```

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

IsInRingOfSize

```
$Status = $Atom->IsInRingOfSize($Size);
```

Returns 1 or 0 based on whether *Atom* is present in a ring of specific *Size*

IsIodine

```
$Status = $Atom->IsIodine();
```

Returns 1 or 0 based on whether it's an iodine *Atom*

IsNitrogen

```
$Status = $Atom->IsNitrogen();
```

Returns 1 or 0 based on whether it's a nitrogen *Atom*

IsNotInRing

```
$Status = $Atom->IsNotInRing();
```

Returns 1 or 0 based on whether *Atom* is not present in a ring

IsOnlyInOneRing

```
$Status = $Atom->IsOnlyInOneRing();
```

Returns 1 or 0 based on whether *Atom* is only present in one ring

IsOxygen

```
$Status = $Atom->IsOxygen();
```

Returns 0 or 1 based on whether it's an oxygen *Atom*

IsPhosphorus

```
$Status = $Atom->IsPhosphorus();
```

Returns 0 or 1 based on whether it's a phosphorus *Atom*

IsPolarAtom

```
$Status = $Atom->IsPolarAtom();
```

Returns 0 or 1 based on whether it's a polar *Atom*. Following atoms are considered polar atoms: N, O, P, S

IsPolarHydrogen

```
$Status = $Atom->IsPolarHydrogen();
```

Returns 0 or 1 based on whether it's a hydrogen *Atom* bonded to a polar atom

IsStereoCenter

```
$Status = $Atom->IsStereoCenter();
```

Returns 0 or 1 based on whether it's marked as a stereo center *Atom* by explicit setting of *StereoCenter* atom property to value of 1.

IsSulfur

```
$Status = $Atom->IsSulfur();
```

Returns 0 or 1 based on whether it's a sulfur *Atom*

SetAtomSymbol

```
$Atom->SetAtomSymbol($AtomicSymbol);
```

Sets atom symbol for *Atom* and returns *Atom* object. The appropriate atomic number is also set automatically.

SetAtomicNumber

```
$Atom->SetAtomicNumber($AtomicNumber);
```

Sets atomic number for *Atom* and returns *Atom* object. The appropriate atom symbol is also set automatically.

SetMassNumber

```
$Atom->SetMassNumber($MassNumber);
```

Sets mass number for *Atom* and returns *Atom* object

SetStereoCenter

```
$Atom->SetStereoCenter($StereoCenter);
```

Sets stereo center for *Atom* and returns *Atom* object.

SetStereochemistry

```
$Atom->SetStereochemistry($Stereochemistry);
```

Sets stereo chemistry for *Atom* and returns *Atom* object

SetX

```
$Atom->SetX($Value);
```

Sets X-coordinate value for *Atom* and returns *Atom* object

SetXYZ

```
$Atom->SetXYZ(@XYZValues);  
$Atom->SetXYZ($XYZValuesRef);  
$Atom->SetXYZ($XYZVector);
```

Sets *Atom* coordinates using an array, reference to an array or a *Vector* object and returns *Atom* object

SetY

```
$Atom->SetY($Value);
```

Sets Y-coordinate value for *Atom* and returns *Atom* object

SetZ

```
$Atom->SetZ($Value);
```

Sets Z-coordinate value for *Atom* and returns *Atom* object

StringifyAtom

```
$AtomString = $Atom->StringifyAtom();
```

Returns a string containing information about *Atom* object

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

Bond.pm, Molecule.pm, MoleculeFileIO.pm

COPYRIGHT

Copyright (C) 2004-2008 Manish Sud. All rights reserved.

This file is part of MayaChemTools.

MayaChemTools is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.