NAME

Bond

SYNOPSIS

use Bond;

use Bond qw(:all);

DESCRIPTION

Bond class provides the following methods:

new, Copy, DeleteBond, GetAtoms, GetBondBeginAtom, GetBondEndAtom, GetBondFromAtom, GetBondToAtom, GetBondedAtom, GetCommonAtom, GetLargestRing, GetNumOfRings, GetNumOfRingsWithEvenSize, GetNumOfRingsWithOddSize, GetNumOfRingsWithSize, GetNumOfRingsWithSizeGreaterThan, GetNumOfRingsWithSizeLessThan, GetRings, GetRingsWithEvenSize, GetRingsWithOddSize, GetRingsWithSize, GetRingsWithSizeGreaterThan, GetRingsWithSizeLessThan, GetSizeOfLargestRing, GetSizeOfSmallestRing, GetSmallestRing, IsAromatic, IsBondStereochemistrySpecified, IsBondTypeSpecified, IsCis, IsCisOrTrans, IsCoordinate, IsDative, IsDouble, IsDown, IsDownward, IsHash, IsInRing, IsInRingOfSize, IsIonic, IsNotInRing, IsOnlyInOneRing, IsQuadruple, IsQuintuple, IsSextuple, IsSingle, IsTautomeric, IsTrans, IsTriple, IsUp, IsUpOrDown, IsUpward, IsWedge, IsWedgeOrHash, SetAtoms, SetBondOrder, SetBondStereochemistry, SetBondType, StringifyBond, SwitchBondFromAndToAtoms

Bond 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

```
$NewBond = new Bond([%PropertyNameAndValues]);
```

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

```
ID = SequentialObjectID
@Atoms = ();
BondType = ""
BondOrder = ""
```

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

Examples:

```
$Bond = new Bond();
$DoubleBond = new Bond('Atoms' => [$Atom2, $Atom1],
                       'BondOrder' => 2);
```

Copy

```
$BondCopy = $Bond->Copy();
```

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

DeleteBond

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

Delete *Bond* between atoms in from a molecule.

GetAtoms

```
@BondedAtoms = $Bond->GetAtoms();
```

Returns an array containing *Atoms* invoved in *Bond*.

GetBondedAtom

```
$BondedAtom = $Bond->GetBondedAtom($Atom);
```

Returns BondedAtom bonded to *Atom* in *Bond*.

GetBondBeginAtom

```
$BeginAtom = $Bond->GetBondBeginAtom();
```

Returns BeginAtom corresponding to bond starting atom in *Bond*.

GetBondEndAtom

```
$EndAtom = $Bond->GetBondEndAtom();
```

Returns EndAtom corresponding to bond ending atom in *Bond*.

GetBondFromAtom

```
$FromAtom = $Bond->GetBondFromAtom();
```

Returns FromAtom corresponding to bond starting atom in *Bond*.

GetBondToAtom

```
$ToAotm = $Bond->GetBondToAtom();
```

Returns ToAtom corresponding to bond ending atom in *Bond*.

GetCommonAtom

```
$CommonAtom = $Bond->GetCommonAtom($OtherBond);
```

Returns Atom common to both *Bond* and *$OtherBond*.

GetLargestRing

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

Returns an array of ring *Atoms* corresponding to the largest ring containing *Bond*. in a molecule

GetNumOfRings

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

Returns number of rings containing *Bond* in a molecule.

GetNumOfRingsWithEvenSize

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

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

GetNumOfRingsWithOddSize

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

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

GetNumOfRingsWithSize

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

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

GetNumOfRingsWithSizeGreaterThan

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

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

GetNumOfRingsWithSizeLessThan

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

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

GetRings

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

Returns an array of references to arrays containing ring atoms corressponding to all rings containing *Bond* in a molecule.

GetRingsWithEvenSize

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

Returns an array of references to arrays containing ring atoms corressponding to all rings with even size containing *Bond* in a molecule.

GetRingsWithOddSize

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

Returns an array of references to arrays containing ring atoms corressponding to all rings with odd size containing *Bond* in a molecule.

GetRingsWithSize

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

Returns an array of references to arrays containing ring atoms corressponding to all rings with specific *RingSize* containing *Bond* in a molecule.

GetRingsWithSizeGreaterThan

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

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

GetRingsWithSizeLessThan

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

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

GetSizeOfLargestRing

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

Returns size of the largest ring containing *Bond* in a molecule.

GetSizeOfSmallestRing

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

Returns size of the smallest ring containing *Bond* in a molecule.

GetSmallestRing

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

Returns an array of ring *Atoms* corresponding to the largest ring containing *Bond* in a molecule.

IsAromatic

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

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

IsBondStereochemistrySpecified

```
$Status = $Bond->IsBondStereochemistrySpecified();
```

Returns 1 or 0 based on whether *Bond*'s sterochemistry is specified.

IsBondTypeSpecified

```
$Status = $Bond->IsBondTypeSpecified();
```

Returns 1 or 0 based on whether *Bond*'s type is specified.

IsCis

```
$Status = $Bond->IsCis();
```

Returns 1 or 0 based on whether it's a cis *Bond*.

IsCisOrTrans

```
$Status = $Bond->IsCisOrTrans();
```

Returns 1 or 0 based on whether it's a cis or trans *Bond*.

IsCoordinate

```
$Status = $Bond->IsCoordinate();
```

Returns 1 or 0 based on whether it's a coordinate or dative *Bond*.

IsDative

```
$Status = $Bond->IsDative();
```

Returns 1 or 0 based on whether it's a coordinate or dative *Bond*.

IsDouble

```
$Status =$Bond->IsDouble();
```

Returns 1 or 0 based on whether it's a double *Bond*.

IsDown

```
$Status = $Bond->IsDown();
```

Returns 1 or 0 based on whether it's a hash or down single *Bond*.

IsDownward

```
$Return = $Bond->IsDownward();
```

Returns 1 or 0 based on whether it's a downward *Bond*.

IsHash

```
$Status = $Bond->IsHash();
```

Returns 1 or 0 based on whether it's a hash or down single *Bond*.

IsInRing

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

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

IsInRingOfSize

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

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

IsIonic

```
$Status = $Bond->IsIonic();
```

Returns 1 or 0 based on whether it's an ionic *Bond*.

IsNotInRing

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

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

IsOnlyInOneRing

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

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

IsQuadruple

```
$Status = $Bond->IsQuadruple();
```

Returns 1 or 0 based on whether it's a quadruple *Bond*.

IsQuintuple

```
$Status = $Bond->IsQuintuple();
```

Returns 1 or 0 based on whether it's a quintuple *Bond*.

IsSextuple

```
$Status = $Bond->IsSextuple();
```

Returns 1 or 0 based on whether it's a sextuple *Bond*.

IsSingle

```
$Status =$Bond->IsSingle();
```

Returns 1 or 0 based on whether it's a single *Bond*.

IsTriple

```
$Status =$Bond->IsTriple();
```

Returns 1 or 0 based on whether it's a triple *Bond*.

IsTautomeric

```
$Status = $Bond->IsTautomeric();
```

Returns 1 or 0 based on whether it's a *Bond*.

IsTrans

```
$Status = $Bond->IsTrans();
```

Returns 1 or 0 based on whether it's a trans *Bond*.

IsUp

```
$Status = $Bond->IsUp();
```

Returns 1 or 0 based on whether it's a up *Bond*.

IsUpOrDown

```
$Status = $Bond->IsUpOrDown();
```

Returns 1 or 0 based on whether it's an up or down *Bond*.

IsUpward

```
$Status = $Bond->IsUpward();
```

Returns 1 or 0 based on whether it's an upward *Bond*.

IsWedge

```
$Status = $Bond->IsWedge();
```

Returns 1 or 0 based on whether it's a wedge *Bond*.

IsWedgeOrHash

```
$Status = $Bond->IsWedgeOrHash();
```

Returns 1 or 0 based on whether it's a wedge or hash *Bond*.

SetAtoms

```
$Bond->SetAtoms($AtomsRef);
$Bond->SetAtoms(@Atoms);
```

Set atoms of *Bond* to atoms in *Atoms* array or in a reference to an array of atoms and return *Bond*.

SetBondOrder

```
$Bond->SetBondOrder($BondOrder);
```

Sets bond order of *Bond* to specified *BondOrder* and returns *Bond*. Possible bond order values: 1 = Single, 1.5 = Aromatic, 2 = Double, 3 = Triple, 4 = Quadruple, 5 = Quintuple, 6 = Sextuple, 7 = Septuple

Notes:

```
. BondType property is automatically assigned using default BondType
  values for specified BondOrder.
. BondType values can also be explicit set.
. To make bonds aromatic in a ring, explicitly set "Aromatic"
  property for bond/atoms and make sure appropriate BondOrder
  values are assigned.
. Dative or coordinate bond types are treated as single bond types with
  explicit formal charge of + and - on first and second bond atoms.
```

SetBondType

```
$Bond->SetBondType($BondType);
```

Sets bond type for *Bond* to specified *BondType* and returns *Bond*. Possible bond type values for different bond orders are:

```
0: None, Ionic, Unspecified
1 : Single, Dative, Coordinate, SingleOrDouble, SingleOrAromatic, Tautomeric
2 : Double, SingleOrDouble, DoubleOrAromatic, Tautomeric
3 : Triple
4 : Quadruple
5 : Quintuple
6 : Sextuple
7 : Septuple
1.5 : Aromatic, Resonance, SingleOrAromatic, DoubleOrAromatic
```

Notes:

```
o BondType Any is valid for all BondOrders.
o BondOrder property is automatically assigned using default BondOrder
  values for specified BondType.
```

Possible bond stereochemistry values for different bond orders are:

```
0 : None, Unspecified
1 : Wedge, Up, Hash, Down, Wavy, WedgeOrHash, UpOrDown, Upward, Downward,
    None, Unspecified
2 : Cis, Trans, Z, E, DoubleCross, CisOrTrans, None, Unspecified
```

SetBondStereochemistry

```
$Bond = $Bond->SetBondStereochemistry($BondStereochemistry);
```

Sets bond stereochemistry of *Bond* to specified *BondStereochemistry* and returns *Bond*. Possible *BondStereoChemistry* values for different bond orders are:

BondOrder: 1

```
None, Unspecified: Not a stereo bond or unspecified

Wedge, Up : Wedge end pointing up
Hash, Down: Wedge end pointing down
Wavy, WedgeOrHash, UpOrDown: Wedge end up or down

Upward: Single bond around cis/trans double bonds pointing upward
Downward: Single bond around cis/trans double bonds pointing upward
```

Notes:

```
o Wedge starts at begin atom of a bond making wedge pointed end always
  at this atom.
o Upward/downward bonds start at atoms involved in cis/trans double bonds.
```

BondOrder: 2

```
None, Unspecified: Not a stereo bond or unspecified

Z, cis: Similar groups on same side of double bond
E, trans: Similar groups on different side of double bond

CisOrTrans, DoubleCross: cis or trans
```

StringifyBond

```
$BondString = $Bond->StringifyBond();
```

Returns a string containing information about *bond* object.

SwitchBondFromAndToAtoms

```
$Bond = $Bond->SwitchBondFromAndToAtoms();
```

Swaps bond from and to atoms in *Bond* and returns *Bond*.

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

Atom.pm, Molecule.pm

COPYRIGHT

Copyright (C) 2018 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.