NAME

MolecularDescriptorsGenerator

SYNOPSIS

use MolecularDescriptors::MolecularDescriptorsGenerator;

use MolecularDescriptors::MolecularDescriptorsGenerator qw(:all);

DESCRIPTION

MolecularDescriptorsGenerator class provides the following methods:

new, GenerateDescriptors, GetAvailableClassAndDescriptorNames, GetAvailableClassNameForDescriptorName, GetAvailableDescriptorClassNames, GetAvailableDescriptorNames, GetAvailableDescriptorNamesForDescriptorClass, GetDescriptorClassParameters, GetDescriptorNames, GetDescriptorNamesAndValues, GetDescriptorValueByName, GetDescriptorValues, GetRuleOf3DescriptorNames, GetRuleOf5DescriptorNames, IsDescriptorClassNameAvailable, IsDescriptorNameAvailable, IsDescriptorsGenerationSuccessful, SetDescriptorClassParameters, SetDescriptorNames, SetMode, SetMolecule, StringifyMolecularDescriptorsGenerator

MolecularDescriptorsGenerator is derived from is derived from ObjectProperty base class that provides methods not explicitly defined in MolecularDescriptorsGenerator or ObjectProperty classes 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>();
```

MolecularDescriptorsGenerator is designed to provide a plug-in environment for molecular descriptors development. The molecular descriptor class modules available in MolecularDescriptors directory under MayaChemTools/lib directory are automatically detected and loaded into the system. The descriptor names provided by each descriptor class module through its GetDescriptorNames function are retrieved and are made available for calculations of their values for a specified molecule.

Any combination of available descriptor names can be specified during calculation of descriptor values using GenerateDescriptors method. The current release of MayaChemTools supports generation of four sets of descriptors: All available descriptors, rule of 5 or 3 descriptors, or a specified set of descriptor names.

RuleOf5 [Ref 91] descriptor names are: MolecularWeight, HydrogenBondDonors, HydrogenBondAcceptors, SLogP. RuleOf5 states: MolecularWeight <= 500, HydrogenBondDonors <= 5, HydrogenBondAcceptors <= 10, and logP <= 5.

RuleOf3 [Ref 92] descriptor names are: MolecularWeight, RotatableBonds, HydrogenBondDonors, HydrogenBondAcceptors, SLogP, TPSA. RuleOf3 states: MolecularWeight <= 300, RotatableBonds <= 3, HydrogenBondDonors <= 3, HydrogenBondAcceptors <= 3, logP <= 3, and TPSA <= 60.

Before calculation of a specified set of descriptors by GenerateDescriptors method, a set of descriptor calculation control parameters for a specific descriptor class name can be set using SetDescriptorClassParameters method. The specified control parameter names and values are simply passed on to specified descriptor class during instantiation of descriptor class object without performing any validation of parameter names and associated values. It's up to the appropriate descriptor class methods to validate these parameters and values. In addition to specified parameter names and values, the parameter hash must also contain descriptor class name as key and value pair with DescriptorClassName as key with class name as value.

METHODS

new

```
$NewMolecularDescriptorsGenerator = new MolecularDescriptors::
                                    MolecularDescriptorsGenerator(
                                    %NamesAndValues);
```

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

```
Mode = 'All'
@{$This->{DescriptorNames}} = ()
```

```
%{$This->{DescriptorClassParameters}} = ()
@{$This->{DescriptorClassNames}} = ()
%{$This->{DescriptorClassObjects}} = ()
@{$This->{DescriptorValues}} = ()
```

Examples:

```
$MolecularDescriptorsGenerator = new MolecularDescriptors::
                                     MolecularDescriptorsGenerator(
                                 'Molecule' => $Molecule);

@DescriptorNames = qw(MolecularWeight HydrogenBondDonors Fsp3Carbons)
$MolecularDescriptorsGenerator = new MolecularDescriptors::
                                     MolecularDescriptorsGenerator(
                                 'Mode' => 'Specify',
                                 'DescriptorNames' => \@DescriptorNames);

$MolecularDescriptorsGenerator->SetDescriptorClassParameters(
                                 'DescriptorClassName' => 'WeightAndMassDescriptors',
                                 'WeightPrecision' => 2,
                                 'MassPrecision' => 2);

$MolecularDescriptorsGenerator->SetDescriptorClassParameters(
                                 'DescriptorClassName' => 'HydrogenBondsDescriptors',
                                 'HydrogenBondsType' => 'HBondsType1');

$MolecularDescriptorsGenerator->SetMolecule($Molecule);
$MolecularDescriptorsGenerator->GenerateDescriptors();
print "MolecularDescriptorsGenerator: $MolecularDescriptorsGenerator\n";
```

GenerateDescriptors

```
$MolecularDescriptorsGenerator->GenerateDescriptors();
```

Calculates descriptor values for specified descriptors and returns *MolecularDescriptorsGenerator*.

Descriptor class objects are instantiated only once at first invocation. During subsequent calls to GenerateDescriptors method, descriptor values are initialized and GenerateDescriptors method provided by descriptor class is used to calculate descriptor values for specified descriptors.

GetAvailableClassAndDescriptorNames

```
%ClassAndDescriptorNames = $MolecularDescriptorsGenerator->
                            GetAvailableClassAndDescriptorNames();
%ClassAndDescriptorNames = MolecularDescriptors::
                             MolecularDescriptorsGenerator::
                             GetAvailableClassAndDescriptorNames();
```

Returns available descriptors class and descriptors names as a hash containing key and value pairs corresponding to class name and an array of descriptor names available for the class.

GetAvailableClassNameForDescriptorName

```
$DescriptorClassName = $MolecularDescriptorsGenerator->
                    GetAvailableClassNameForDescriptorName($DescriptorName);

$DescriptorClassName = MolecularDescriptors::MolecularDescriptorsGenerator::
                    GetAvailableClassNameForDescriptorName($DescriptorName);
```

Returns available descriptor class name for a descriptor name.

GetAvailableDescriptorClassNames

```
$Return = $MolecularDescriptorsGenerator->GetAvailableDescriptorClassNames();

@DescriptorClassNames = $MolecularDescriptorsGenerator->
                            GetAvailableDescriptorClassNames();
```

```
@DescriptorClassNames = MolecularDescriptors::
                        MolecularDescriptorsGenerator::
                        GetAvailableDescriptorClassNames();
```

Returns available descriptor class names as an array or number of available descriptor class names in scalar context.

GetAvailableDescriptorNames

```
@DescriptorNames = $MolecularDescriptorsGenerator->
                        GetAvailableDescriptorNames();
@DescriptorNames = MolecularDescriptors::
                        MolecularDescriptorsGenerator::
                        GetAvailableDescriptorNames();
```

Returns available descriptor names as an array or number of available descriptor names in scalar context.

GetAvailableDescriptorNamesForDescriptorClass

```
@DescriptorNames = $MolecularDescriptorsGenerator->
      GetAvailableDescriptorNamesForDescriptorClass($DescriptorClassName);
@DescriptorNames = MolecularDescriptors::
                   MolecularDescriptorsGenerator::
      GetAvailableDescriptorNamesForDescriptorClass($DescriptorClassName);
```

Returns available descriptors names for a descriptor class as an array or number of available descriptor names in scalar context.

GetDescriptorClassParameters

```
$DescriptorClassParametersRef = $MolecularDescriptorsGenerator->
                        GetDescriptorClassParameters();
$DescriptorClassParametersRef = MolecularDescriptors::
                                MolecularDescriptorsGenerator::
                                GetDescriptorClassParameters();
```

Returns descriptor name parameters as a reference to hash of hashes with hash keys corresponding to class name and class parameter name with hash value as class parameter value.

GetDescriptorNames

```
@DescriptorNames = $MolecularDescriptorsGenerator->GetDescriptorNames();
@DescriptorNames = MolecularDescriptors::MolecularDescriptorsGenerator::
                   GetDescriptorNames();
```

Returns all available descriptor names as an array or number of available descriptors in scalar context.

GetDescriptorNamesAndValues

```
%NamesAndValues = $MolecularDescriptorsGenerator->
                        GetDescriptorNamesAndValues();
```

Returns calculated molecular descriptor names and values as a hash with descriptor names and values as hash key and value pairs.

GetDescriptorValueByName

```
$Value = $MolecularDescriptorsGenerator->
                        GetDescriptorValueByName($Name);
```

Returns calculated descriptor values for a specified descriptor name.

GetDescriptorValues

```
@DescriptorValues = $MolecularDescriptorsGenerator->GetDescriptorValues();
```

Returns all calculated descriptor values as an array corresponding to specified descriptor names.

GetRuleOf3DescriptorNames

```
@DescriptorNames = $MolecularDescriptorsGenerator->
                   GetRuleOf3DescriptorNames();
@DescriptorNames = MolecularDescriptors::
                   MolecularDescriptorsGenerator::
                   GetRuleOf3DescriptorNames();
```

Returns rule of 3 descriptor names as an array or number of rule of 3 descriptors in scalar context.

RuleOf3 [Ref 92] descriptor names are: MolecularWeight, RotatableBonds, HydrogenBondDonors, HydrogenBondAcceptors, SLogP, TPSA. RuleOf3 states: MolecularWeight <= 300, RotatableBonds <= 3, HydrogenBondDonors <= 3, HydrogenBondAcceptors <= 3, logP <= 3, and TPSA <= 60.

GetRuleOf5DescriptorNames

```
@DescriptorNames = $MolecularDescriptorsGenerator->
                           GetRuleOf5DescriptorNames();
@DescriptorNames = $MolecularDescriptorsGenerator::
                          GetRuleOf5DescriptorNames();
```

Returns rule of 5 descriptor names as an array or number of rule of 4 descriptors in scalar context.

RuleOf5 [Ref 91] descriptor names are: MolecularWeight, HydrogenBondDonors, HydrogenBondAcceptors, SLogP. RuleOf5 states: MolecularWeight <= 500, HydrogenBondDonors <= 5, HydrogenBondAcceptors <= 10, and logP <= 5.

IsDescriptorClassNameAvailable

```
$Status = $MolecularDescriptorsGenerator->
                           IsDescriptorClassNameAvailable($ClassName);
$Status = MolecularDescriptors::
                           MolecularDescriptorsGenerator::
                           IsDescriptorClassNameAvailable($ClassName);
```

Returns 1 or 0 based on whether specified descriptor class name is available.

IsDescriptorNameAvailable

```
$Status = $MolecularDescriptorsGenerator->
                           IsDescriptorNameAvailable($DescriptorName);
$Status = MolecularDescriptors::
                           MolecularDescriptorsGenerator::
                           IsDescriptorNameAvailable($DescriptorName);
```

Returns 1 or 0 based on whether specified descriptor name is available.

IsDescriptorsGenerationSuccessful

```
$Status = $MolecularDescriptorsGenerator->
                           IsDescriptorsGenerationSuccessful();
```

Returns 1 or 0 based on whether descriptors generation is successful.

SetDescriptorClassParameters

```
$MolecularDescriptorsGenerator->SetDescriptorClassParameters(
                           %NamesAndValues);
```

Sets descriptor calculation control parameters for a specified descriptor class name and returns *MolecularDescriptorsGenerator*.

The specified parameter names and values are simply passed on to specified descriptor class during instantiation of descriptor class object without any performing any validation of parameter names and associated values. It's up to the appropriate descriptor class methods to validate these parameters and values.

In addition to specified parameter names and values, the parameter hash must also contain descriptor class name as key and value pair with DescriptorClassName as key with class name as value.

SetDescriptorNames

```
$MolecularDescriptorsGenerator->SetDescriptorNames(@Names);
```

```
$MolecularDescriptorsGenerator->SetDescriptorNames(\@Names);
```

Sets descriptor names to use for generating descriptor values using an array or reference to an array and returns *MolecularDescriptorsGenerator*.

SetMode

```
$MolecularDescriptorsGenerator->SetMode($Mode);
```

Sets descriptors generation mode and returns *MolecularDescriptorsGenerator*. Possible *Mode* values: *All, RuleOf5, RuleOf3, Specify*.

SetMolecule

```
$MolecularDescriptorsGenerator->SetMolecule($Molecule);
```

Sets molecule to use during calculation of molecular descriptors and returns *MolecularDescriptorsGenerator*.

StringifyMolecularDescriptorsGenerator

```
$String = $MolecularDescriptorsGenerator->StringifyMolecularDescriptorsGenerator();
```

Returns a string containing information about *MolecularDescriptorsGenerator* object.

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

MolecularDescriptors.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.