# Atoms and Molecules

## Table of contents

## 1. Atoms and Isotopes

Each isotope of an atom shares the same atomic number (Z) whereas it differs from the other by its number of neutrons, it has a different mass number (A).

Then each atom contains a set of several isotopes. Moreover, each isotope in an atom has a specific relative abundance in nature. We'll see later the implication of this parameter in the calculation of atomic mass.

In the JPL, atoms and isotopes are thus evaluated differently. Atomic objects are evaluated in a set context compared with isotopes. But both objects implement the same interface `JPLIAtom`

## 2. Molecules

A molecule is an assembling of atoms. In JPL, molecules can be neutral or charged.

We tried to respect the nomenclature of molecular designation with groups of atom followed by their number of occurrences. We defined a grammar that describe what is an atom or a molecule.

Each query is transmitting through a string that represents a chemical formula.

> **Note:**
> The string representation of a molecule is not really respecting the atom ordering. For example "OH" will be displayed "HO". In fact, atoms in a molecule are sorted lexicographically whereas it is not always the case in the chemistry nomenclature. Take the example of "H2O" to convince yourself.

## 3. Formal Atomic/Molecular Definition

- `Formula := Begin-avg-accuracy ? Group-element+Charge End-avg-accuracy?`
- `Begin-avg-accuracy := '<'`
- `End-avg-accuracy := '>'`
- `Group-element := Element | Group`
- `Group := '(' Element ')' Number-of-groups`
- `Number-of-groups := [0-9]+`
- `Element := Atom-name Number-of-mass? Number-of-atoms`
- `Atom-name := [A-Z][a-z]?`
- `Number-of-mass := '[' [0-9]{1,3} ']'`
- `Number-of-atoms := -?[0-9]+`
- `Charge := '(' [0-9]* [+-] ')'`

## 4. Chemical Access and Caching

We define a single entry point to access atoms and molecules.

The public static method `JPLChemicalFacade.getMolecule(String formula)` returns a molecule or throws a parse exception if the given formula is not recognized.

Atoms and isotopes are also accessible from this class through static methods `getAtom(String name)`, `getIsotope(String name, int massNumber)` or `getElement(String element)`.

Through those static methods we ensure to control their instanciation. And as such, every instance of atoms, isotopes and molecules are unique.

## 5. About Masses and Mass Accuracy

Atoms and isotopes express a slight difference concerning their mass evaluation in the JPL.
- Each mass of atom has been averaged over all its isotopes.
- Each mass of isotope is directly evaluated.

The presence of isotopes at their natural abundances makes it essential to define whether our molecular mass value is an "average" value, equivalent to taking the average of all isotopic masses considering their relative abundance, or a "monoisotopic" value, the mass of the most abundant isotope.

When considering the mass of molecule we have to decide a priori what kind of atom or isotope we have.

## 6. Global Evaluation

Any formula is implicitly evaluated in a *monoisotopic* context. That means the returned molecule will be composed of isotopes only ! See below for an average evaluation.

For instance `CH3` is evaluated as `C[12]H[1]3`. A call to getMass() on this molecule will returns the sum of all isotopic mass.

> **Note:**
> Now that JPLMassCalculator exists, we perhaps have to disable the mass access from molecule and force this single entry point rendering `getMass()` deprecated in the future.

## 7. Mass Average Accuracy

To define a mass average context, you will have to surround your formula with diamonds '<' and '>'. For instance <CH3> specify that each molecular element will correspond to the specific atom (and not the isotope). You can see all the grammar above.

## 8. Examples of Formulae

CH3, H2O, C2H6OH, OH(-), H3O(+), CH2, O[16]H(-), CH3(CH2)2O(CH2)3OH(-), <C[13]2H-3ON-1(2+)>