Fall 2020

CS149 – Introduction to Programming

Programming Assignment 11

Chemicalc

`Atom`

Overview

Chemicalc is a large computational chemistry system. The design phase has been completed and the implementation phase is now starting. You must implement the `Atom` class for this system.

You will not be developing the main class, nor will you have access to it.

Textual Description

The following textual description was written and used in the design of the class you must implement.

Atom

An atom is a unit of matter that is made up of protons, electrons, and neutrons. Protons and neutrons are in the nucleus (hence are sometimes called nucleons) and electrons orbit the nucleus. Every atom has the same number of protons and electrons.

An Atom is defined by its symbol (e.g., "Cl" for chlorine), its atomic number (i.e., the number of protons), and its mass number (i.e., the total number of its protons and neutrons).

An atom is sometimes referred to (e.g., in a printed document) in a terse fashion and sometimes in a verbose fashion. In the terse fashion, it is referred to by its symbol (e.g., "Cl"). In the verbose fashion, it is referred to by its symbol, a dash, and its mass number (e.g., "Cl-35"). The normal way of referring to an atom is the terse fashion, and that is the approach we will use.

Specifications

Atom

`Atom` objects must be immutable.

In addition, the `Atom` class must comply with the following specifications:

1. It must have a private attribute named `atomicNumber` that is an `int` containing the atomic number (i.e., the number of protons).
2. It must have a private attribute named `massNumber` that is an `int` containing the mass number (i.e., the number of protons and neutrons).
3. It must have a private attribute named `symbol` that is a `String` containing the symbol.
4. It must have an explicit value constructor with the following signature:

   ```
   public Atom(String symbol, int atomicNumber,
               int massNumber)
   ```
5. It must have a method with the following signature:

   ```
   public boolean equals(Atom other)
   ```

   This method must return `true` if and only if the owning `Atom` has the same number of protons and neutrons as the `Atom` named `other`.
6. It must have a method with the following signature:

   ```
   public String toString()
   ```

   This method must return the `String` representation of the `Atom` (i.e., its `symbol`).
7. It must have the following accessors:

   ```
   public int getAtomicNumber()
   public int getElectrons()
   public int getMassNumber()
   public int getNeutrons()
   public int getProtons()
   ```

The `Atom` class may have other private attributes and/or methods.

Recommended Process

You should think about the order in which you should implement the methods in the `Atom` class. It is **strongly recommended** that you work on (i.e., implement **and test**) one method at a time. However, you are certainly entitled to ignore this advice and “shoot yourself in the foot”.

After you have implemented and tested all of the methods, you must submit `Atom.java` in a file named `pa11.zip` using Autolab. Do not include any other files in the `.zip` file.

Note that your submission will not compile against the official tests if it does not include every method in the class. Hence, if you are unable to complete a method, you should create a “stub” for it (i.e., a method with the correct signature, no body, and a return statement if necessary) before you submit your code. For example, if you are unable to complete the `equals()` method, you should include the following “stub”:

```
public boolean equals(Atom other) {
   return false;
}
```

Grading

Your code will first be graded by Autolab and then by the Professor. The grade you receive from Autolab is the maximum grade that you can receive on the assignment.

Autolab Grading

Your code must compile (in Autolab, this will be indicated in the section on “Does your code compile?”), and all class names and method signatures comply with the specifications (in Autolab, this will be indicated in the section on “Do your class names, method signatures, etc. comply with the specifications?”) for you to receive any points on this assignment.

Autolab will then grade your submission as follows:

Conformance to the Course Style Guide:	**20 points** (All or Nothing)
Correctness:	**80 points** (Partial Credit Possible)

Manual Grading

After the due date, the Professor may manually review your code. At this time, points may be deducted for inelegant code, inappropriate variable names, bad comments, etc.