Java Classes for Managing Chemical Information and Solving Generalized Equilibrium Problems

David N. Blauch†

Department of Chemistry, Davidson College, P.O. Box 7120, Davidson, North Carolina 28115

Received August 7, 2001

Java classes have been created for organizing chemical information and solving generalized equilibrium problems. An object-oriented approach is employed for the organization and manipulation of chemical information. Classes have been created to represent chemical species, phases, and chemical reactions. The representation of the entire chemical system is encapsulated in the ChemSystem class. The Equilibria class provides methods for analyzing a chemical system, as described by a ChemSystem object, and determining the amounts of each species in the system at equilibrium. The ChemEquilibria applet has been created to facilitate deployment of this software over the World Wide Web.

A variety of programs have been written to solve equilibrium problems of various types. Most of this software is customized for the solution of very specific types of problems such as acid−base equilibria[1] or a single reaction at equilibrium.[2] Some programs for solving relatively general equilibrium problems have been produced; notable examples include SEQEx2 (written in Pascal),[3] SALT (written in FORTRAN IV),[4] SEQS,[5] and EQUIL.[6] While each of these programs is useful in its own context, the programming language and organization of information limits the use of each program to a specific platform and does not permit modification of the software.

This paper describes a Java software package named Equilibria and an associated Java applet named ChemEquilibria that solve arbitrary equilibrium problems. What distinguishes this software package from programs such as those mentioned above is that it is written in Java[7] and includes an object-oriented approach to management of chemical information. Java offers several extremely important advantages over other programming languages. Perhaps the most significant advantage is portability. A program written in Java can be executed on any machine equipped with the Java Virtual Machine, which is now common on virtually all operating systems, including the various Windows operating systems, Mac OS, and Unix (including Linux). Thus unlike programs written in Pascal, FORTRAN, or C, a single copy of a program written in Java can run on almost any platform without modification. Associated with this portability is support for applets, which permits Java programs to be readily distributed via the Internet and to run within a Web browser.

Java is an object-oriented language,[8] and the principles of object-oriented programming have been applied to the organization of chemical information in the Equilibria package. An important limitation of conventional programming techniques, and thus of existing computational chemistry programs (including, for example, those designed to solve equilibrium problems), is that it is difficult to customize or modify an existing program for a new application. Despite a programmer's best foresight and imagination, it is impossible to anticipate every application for which a program might be employed. Users will inevitably desire new features and modification of existing features. For programs written in FORTRAN, Pascal, and C, for example, such improvements are not possible, unless the original programmer is willing to make the source code available, and even in this case it is often difficult to modify conventional programs owing to the highly intertwined nature of traditional (i.e., non-object-oriented) data organization. This limitation is at least partially remedied through the inheritance and polymorphism behavior of object-oriented programming.

Any computational chemistry problem involves both a representation of a chemical system and computations based upon that representation. Object-oriented programming provides a powerful technique for organizing and manipulating information, including chemical information, as will be shown below. In the future, chemists might employ an archive of classes representing the components and features of chemical systems, and these classes would be used to create objects describing the specific molecules, reactions, and processes of interest to the chemist. These objects would then be submitted to programs which would perform various manipulations (e.g., equilibrium calculations, geometry optimizations, and energy calculations) based upon the information contained in the object. The Equilibria package does not yet represent the full realization of this vision, but it does provide a clear example of how object-oriented programming can be used to advantage for the description of chemical information and will hopefully facilitate the development of more extensive classes for use in computational chemistry.

REPRESENTATION OF A CHEMICAL SYSTEM

The classes in the Equilibria package fall into various categories, as illustrated in Figure 1. The data classes provide a framework for organizing the chemical information for the system using object-oriented concepts. The compositional

† Corresponding author phone: (704) 894-2308; e-mail: dablauch@davidson.edu.

10.1021/ci010074+ CCC: $22.00
Published on Web 01/18/2002

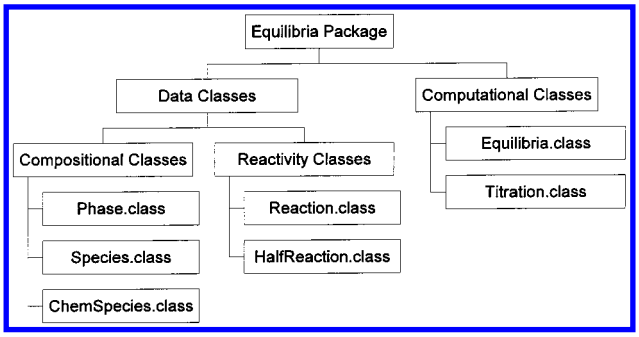

Figure 1. Class types in the Equilibria package.

classes (Phase, Species, and ChemSpecies) describe the composition of a component of the system. The reactivity classes (Reaction and HalfReaction) characterize a chemical reaction or half-reaction. The ChemSystem class contains both types of data objects and encapsulates the description of the entire chemical system. The computational classes (Equilibria and Titration) accept a ChemSystem object and perform one or more operations based upon the information contained in that object.

A chemical system comprises one or more phases (pure solid, pure liquid, gas, or solution) with each phase containing one or more chemical species. Some properties of a chemical species (e.g., molar concentration) depend on the properties of the phase (e.g., volume), while other properties are intrinsic to the species and independent of the phase in which the species exists (e.g., formula weight). Each element of this organizational scheme is described by a Java class: the Phase class describes a pure solid, pure liquid, gas, or solution; the Species class describes the properties of a chemical species in a specific phase; and the ChemSpecies class describes the intrinsic properties of a chemical species. Each class contains several properties and may serve as a container for various objects. In addition, each class provides methods for defining and retrieving the values of various properties.

The fundamental representation of chemical reactivity is the Reaction class, which describes all relevant properties of a chemical reaction. Each chemical reaction is characterized by a set of Species objects (not ChemSpecies objects, because the phase in which the species exists is important) that participate in the chemical reaction and an associated set of stoichiometric coefficients. The Reaction class also contains values for all thermodynamic properties (e.g., the equilibrium constant) for the chemical reaction. The HalfReaction class is very similar to the Reaction class, except that it describes a half-reaction.

An overview of the relations between the classes in the Equilibria package is presented in Figure 2. A ChemSystem object contains one or more Phase objects and one or more Reaction objects. Each Phase object contains one or more Species object, and each Reaction object contains two or more Species objects. Each Species object contains a single ChemSpecies object that characterizes the intrinsic features of the species. Each object obviously can identify the objects it contains; in some cases the reverse determination is also possible, as indicated by the arrows in Figure 2. A Species object can identify the ChemSpecies object on which it is based, but a ChemSpecies object cannot identify Species objects derived from it. A Species object, however, can identify the Phase in which it exists, and the Phase object

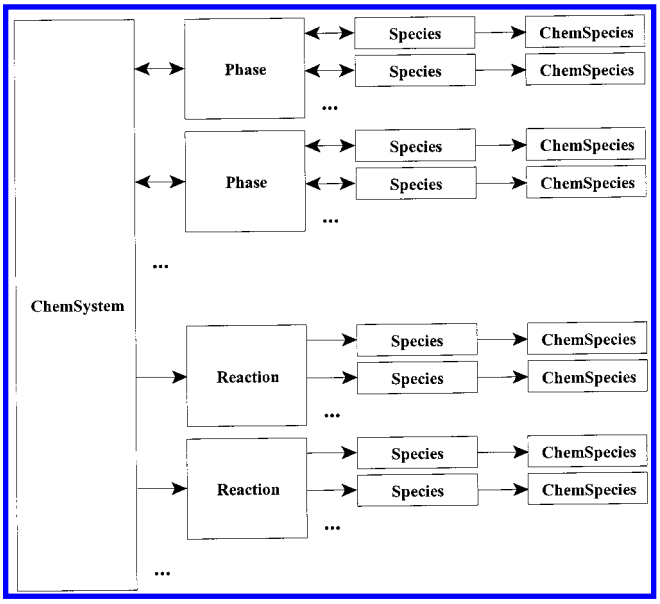

Figure 2. Organizational diagram for ChemSpecies, Phase, Species, ChemSpecies, and Reaction classes.

can identify the ChemSystem in which it exists. Detailed documentation for each class and its methods is available at the ChemEquilibria Web site.[9]

CHEMICAL COMPUTATIONS

In this model for managing chemical information, the complete description of a chemical system is contained in a ChemSystem object. Computations, such as solving an equilibrium problem, are performed by passing a ChemSystem object to an appropriate computational engine, which accesses the relevant chemical information through the appropriate methods of the ChemSystem object and then performs the necessary chemical and mathematical analysis. This explicit separation of the classes representing data (composition and reactivity) from those for performing computations provides a great deal of flexibility. Researchers wishing to perform computations other than equilibrium calculations may still employ the data classes from the Equilibria package to represent the chemical system information. A new class would be created to perform the desired computations using the data provided by a ChemSystem object.

The Equilibria package provides two classes that contain computational engines for solving equilibrium problems: the Equilibria class and the Titration class. The Equilibria class has methods for parsing the chemical information in the ChemSystem object in order to verify its chemical and logical consistency, creating the master set of mathematical equations that describe the equilibrium problem, and solving the mathematical representation of the equilibrium problem. The Titration class provides methods for calculating a titration curve and for fitting experimental titration data.

In defining a chemical system, it is important to recognize that the chemist is only aware of how the chemical system was created or assembled. In the context of solving equilibrium problems, for example, the chemist knows what substances were added to the system (these are the analytical or formal amounts) but does not necessarily know, without independent calculations or measurements, the actual com-

position of the chemical system. For this reason the Phase and Species classes maintain separate records of the analytical and actual amounts of each component. The methods of these classes only permit users to alter the analytical amounts. The actual amounts must be determined by means of an equilibrium calculation. Thus it is possible for a computational engine (such as an Equilibria object) to alter properties of the ChemSystem object. The practice employed in this software is for computational engines to alter actual amounts of the various components of the system, but not to alter the analytical amounts. In this way the ChemSystem class preserves the user's original description of the chemical system.

CHEMEQUILIBRIA APPLET

The ChemEquilibria applet provides a Web-based interface for the classes in the Equilibria package. The ability to deploy the Equilibria package in a Web browser through the use of the ChemEquilibria applet greatly expands the utility of this software. In many institutions the World Wide Web is the preferred mechanism for accessing and manipulating information. The Web browsers most widely utilized at present provide very limited JavaScript support for Java objects. Consequently the ChemEquilibria applet employs only numeric (int or float) and String values as arguments and as values returned from methods. Data that would be most conveniently conveyed as an array is instead accessed element by element on the basis of the element index. This approach provides the most general browser compatibility.

The implementation of the ChemEquilibria applet is illustrated in a series of Web pages available on the ChemEquilibria Web site.[10] These Web pages display the JavaScript commands necessary to describe various chemical systems and to perform the equilibrium calculations. At the time of writing, the following sample Web pages are available: dissolution of calcium carbonate in water exposed to atmospheric carbon dioxide,[11] an electrochemical cell comprising a calomel electrode and a silver−silver chloride,[12] the Haber process,[13] the titration of calcium ion by EDTA in the presence of an ammonia buffer,[14] the potentiometric titration of a solution containing iron(II) and tin(II) ions with potassium dichromate in acidic solution,[15] and titration of hydrochloric acid with ethylenediamine in the absence and presence of nickel(II) ion.[16] The last example illustrates the use of the Titration class to analyze and curve-fit experimental titration data.

CREATION OF NEW CLASSES

A powerful feature of object-oriented programming is the ability to modify an existing class to serve a new purpose. For example, the Species class employs the extended Debye−Hückel equation to compute activity coefficients for charged solutes. This computation is performed by the getActivityCoefficient method. A user wishing to employ an alternate, perhaps more sophisticated, equation need only create a new class having the Species class as its super class and write a new version of the getActivityCoefficient method implementing the desired mathematical formula for activity coefficients. The new class will automatically possess all of the properties and methods of the Species class (this is the inheritance feature of object-oriented programming), and the new version of the getActivityCoefficient method will replace the version present in the super class.

To illustrate a more complicated example of how inheritance may be employed to solve new problems, consider the case of a chemist performing spectroscopic measurements on a series of solutions. The Equilibria package provides no support for spectroscopic measurements, but the chemist may introduce such support by creating new classes derived from those in the Equilibria package. One possibility is to create a new class, the NewSpecies class, that extends the Species class and add a method (getMolarAbsorptivity) for obtaining the molar absorptivity of the species. (Methods for defining the absorbance spectrum of the species would also be required.) The inheritance feature of the Java language transfers all of the properties and functionality of the Species class to the NewSpecies class. The constructors for the NewSpecies class should parallel those of the super class and call the corresponding constructor from the super class. One could also create additional constructors for the NewSpecies class (for example, a constructor that includes a definition of the spectral properties of the species).

Having created the NewSpecies class to characterize the spectroscopic properties of a given species, the NewSolution class may be created by extending the AqueousSolution class (assuming one is working exclusively with aqueous solutions; otherwise it would be better to extend the Solution class) to permit the absorbance of a solution to be determined. The NewSolution class would include a method (getAbsorbance) that returns the absorbance of the solution at a particular wavelength for a particular cell path length:

```
public double getAbsorbance(double wavelength, double pathLength) {
    Species[] sp = getSpecies();
    double sum = 0.0;
    for (int i=0; i<sp.length; i++) {
        if (sp[i] instanceof NewSpecies) {
            NewSpecies nsp = (NewSpecies) sp[i];
            sum += nsp.getMolarAbsorptivity(wavelength)*nsp.getConc();
        }
    }
    return sum*pathLength;
}
```

The reader will note that the NewSolution object may contain both Species and NewSpecies objects, of which only the latter support the getMolarAbsorptivity method. The class's getSpecies method returns both types of objects, and in calculating the absorbance the method must check each object to determine if it is an instance of the NewSpecies class before calling the getAbsorbance method. (This implementation effectively assumes that Species objects represent optically transparent species.)

The NewSpecies and NewSolution objects may be employed in solving equilibrium problems in exactly the same manner as the Species and AqueousSolution objects. The only limitation is that many of the methods in the ChemSystem class and ChemEquilibria applet automatically create Solution and Species objects. Thus one must manually create a NewSolution object and add it to the ChemSystem object and manually create a NewSpecies object and add it to the NewSolution object. A sample Java applet named EquilDemo has been written to illustrate this process. The EquilDemo applet models a hypothetical system containing a metal and a ligand in which several stepwise complexation reactions

occur. The applet permits the absorbance of the solution to be determined for any wavelength and cell path length for any set of analytical concentrations for the metal and ligand. The source code files (NewSpecies.java, NewSolution.java, and EquilDemo.java) are also available on a Web page illustrating the behavior of the EquilDemo applet.[17]

SYSTEM REQUIREMENTS

The ChemEquilibria applet and Equilibria package (edu.davidson.chm.equilibria.*) were compiled using Java Development Kit (JDK) 1.2.2 using only classes and methods present in JDK 1.1. Thus this code may be employed on any system on which the Java Runtime Environment (JRE) 1.1 or later is installed. The archive files chemEquilibria.jar and equilibria.jar may be downloaded from the ChemEquilibria[18] and Equilibria[19] home pages. This software is available free of charge provided the author is credited as the source of the software and the software is not resold.[20] Any publication based upon results obtained using the Equilibria package should include a similar acknowledgment.

CONCLUSIONS

The Equilibria package provides a library of Java classes that provide support for managing chemical information and for solving equilibrium problems. ChemEquilibria is a Java applet that provides a browser-compatible interface to objects created from the classes in the Equilibria package. This software is well-suited for use in both research and education and serves as an example of how an object-oriented program may be employed to manage and manipulate chemical information. An example of the educational implementation of this software is a series of interactive virtual chemistry experiments.[21]

ACKNOWLEDGMENT

The author thanks Professors Rodger Nutt, Wolfgang Christian, and Anthony Kapolka for helpful discussions.

REFERENCES AND NOTES

(1) See, for example: (a) Ramette, R. W.; Holmes, J. L. The Acid−Base Package. *J. Chem. Educ. Software* 4C2. (b) Ramette, R. W. Buffers Plus. *J. Chem. Educ. Software* 9803.

(2) See, for example: Allendoerfer, R. D. Equilibrium Calculator. *J. Chem. Educ. Software* 1D1.

(3) Mioshi, R. N.; do Lago, C. L. An Equilibrium Simulator for Multiphase Equilibria based on the Extent of Reaction and Newton−Raphson Method with Globally Convergent Strategy (SEQEx2). *Anal. Chim. Acta* **1996**, *334*, 271−278.

(4) Walters, L. J., Jr.; Wolery, T. J. A Monotone-Sequences Algorithm and FORTRAN IV Program for Calculation of Equilibrium Distributions of Chemical Species. *Comput. Geosci.* **1975**, *1*, 57−63.

(5) Tucker, E. E.; Thompson, L. C.; Poe, D. P. Simultaneous Equations and Ionic Equilibrium in Chemistry. *Am. Lab.* **1991**, *23*, 82−89.

(6) Ting-Po, I.; Nancollas, G. EQUIL−A General Computational Method for the Calculation of Solution Equilibria. *Anal. Chem.* **1972**, *44*, 1940−1950.

(7) The Source for Java(TM) Technology. http://www.java.sun.com (accessed May 2001).

(8) For a detailed explanation of object-oriented programming and the Java language, consult: (a) Liwu, L. *Java: Data Structures and Programming*; Springer: New York, 1998. (b) *Object-Oriented Languages, Systems and Applications*; Blair, G., Gallagher, J., Hutchison, D., Shepherd, D., Eds.; Wiley: New York, 1990. (c) Horton, I. *Beginning Java*; Wrox Press: Birmingham, UK, 1997.

(9) Blauch, D. N. Equilibria and ChemEquilibria Documentation. http://www.chm.davidson.edu/ChemEquilibria/doc/ (accessed May 2001).

(10) Blauch, D. N. The ChemEquilibria Applet: General Equilibrium Problem-Solving Software. http://www.chm.davidson.edu/ChemEquilibria/ChemEquilibria.html#Examples (accessed May 2001).

(11) Blauch, D. N. Dissolution of Calcium Carbonate in Air-Saturated Water. http://www.chm.davidson.edu/ChemEquilibria/Dissolution.html (accessed May 2001).

(12) Blauch, D. N. Demonstration of the ChemEquilibria Applet: An Electrochemical Cell. http://www.chm.davidson.edu/ChemEquilibria/ElectrochemicalCell.html.

(13) Blauch, D. N. Demonstration of the ChemEquilibria Applet: The Haber Process. http://www.chm.davidson.edu/ChemEquilibria/HaberProcess.html (accessed May 2001).

(14) Blauch, D. N. Demonstration of the ChemEquilibria Applet: A Complexation Titration. http://www.chm.davidson.edu/ChemEquilibria/ComplexationTitration.html (accessed May 2001).

(15) Blauch, D. N. Demonstration of the ChemEquilibria Applet: Potentiometric Titration. http://www.chm.davidson.edu/ChemEquilibria/PotentiometricTitration.html (accessed May 2001).

(16) (a) Blauch, D. N. Demonstration of the ChemEquilibria Applet: Curve-Fitting Part 1. http://www.chm.davidson.edu/ChemEquilibria/CurveFit1.html (accessed May 2001). (b) Blauch, D. N. Demonstration of the ChemEquilibria Applet: Curve-Fitting Part 2. http://www.chm.davidson.edu/ChemEquilibria/CurveFit2.html (accessed May 2001).

(17) Blauch, D. N. Demonstration of Extending the Equilibria Classes. http://www.chm.davidson.edu/ChemEquilibria/EquilDemo.html (accessed May 2001).

(18) Blauch, D. N. The ChemEquilibria Applet: General Equilibrium Problem-Solving Software. http://www.chm.davidson.edu/ChemEquilibria/ChemEquilibria.html (accessed May 2001).

(19) Blauch, D. N. The Equilibria Package: General Equilibrium Problem-Solving Software. http://www.chm.davidson.edu/ChemEquilibria/Equilibria.html (accessed May 2001).

(20) Blauch, D. N. Equilibria Package and ChemEquilibria Applet: Conditions of Use. http://www.chm.davidson.edu/ChemEquilibria/Equilibria.html (accessed July 2001).

(21) See, for example: Blauch, D. N. Chemical Equilibria: Calculations involving Reaction Stoichiometry. http://www.chm.davidson.edu/ChemicalApplets/equilibria/ReactionTable. html (accessed July 2001).

CI010074+