

# Teaching Statement

Kathryn E. Gray

My teaching experiences began in college, when I spent two weeks as an instructor for the TEACHSCHEME! workshops. Since graduating, I have continued to participate in these workshops, which introduce high school teachers to computer science. I also collaborated with the PLT research team to develop a follow-up object-oriented curriculum, spent a year studying this curriculum in a classroom, and last summer taught my own introductory course based on the TEACHSCHEME! and follow-up curriculums.

A TEACHSCHEME! workshop spends one week presenting the material from *How to Design Programs*, Felleisen et al., to high school computer science teachers. With few exceptions, the participants do not know how to program, so the week is spent training the teachers as introductory students. The teachers learn functional Scheme programming using a design recipe, which provides systematic steps for solving problems. During my time in the workshops, I observed many novice programmers improve dramatically following the design recipe approach.

My observations motivated me to work with the PLT team in developing a follow-up curriculum presenting object-oriented programming. My research project, ProfessorJ, directly supports the new curriculum. ProfessorJ is a pedagogic compiler for three subsets of Java. As students progress through a course, they advance through the subsets, seeing only as much Java as the class has presented.

After completing my initial implementation, I applied for (and received) a University Teaching Assistantship to study the combination of ProfessorJ and the initial version of *How to Design Class Hierarchies* (the text for the new curriculum). Observing students experience this combination led to several improvements in the language subsets.

When the University of Chicago decided to revamp their professional masters program introductory sequence, they chose to hire me to present an object-oriented first course using the ProfessorJ environment. In preparing for the course, it became apparent that the *HtD Class Hierarchies* curriculum as it stood was too advanced for students with no programming experience. Therefore, I developed a combination curriculum, mixing the introductory material of *How to Design Programs* with the object-oriented (and Java) content of *How to Design Class Hierarchies*.

In the course, most of my lectures began with the presentation of a topic, followed by the class collectively writing a program illustrating the topic. This lecture style gave the students controlled programming experiences, and allowed me to assess their understanding immediately. When the students' participation demonstrated their confusion, I was able to reorganize the remainder of the lecture to re-explain the material using different examples. The lecture format also encouraged the students to discuss the program with me and with each other.

For the final project, I required each student to present their plans to me using documentation and informal presentations, in sufficient detail that another programmer could implement their project. This experience led several students to realize fundamental flaws in their designs, and how to correct them.

Overall, the experience demonstrated the benefits of planning class participation and requiring students to present and discuss their ideas. In future courses, I intend to assign more presentations and short essays. My experience further demonstrated to me the benefits of teaching a systematic system for solving problems, as the students could always assess their progress and knew what should happen next when programming.

In general, when teaching a computer science course, students should be taught problem solving and communication skills as well as the technical content of the course. Through assignments and in-class discussion, students can practice conveying technical designs and assessments. Additionally, this dialogue provides the instructor with simple feedback on topic comprehension. Students should also be taught how to analyze a problem, assess their own progress, and reach a solution.

My teaching and research experiences have prepared me to teach introductory computer science. My personal experience, as a former English and history major who switched to computer science only after a non-traditional exposure to the material, motivates me to find additional ways to present computer science that appeal to non-traditional students. From my research, I am equipped to teach programming languages, compilers, and software design. I also have sufficient background experience in A.I. to teach introductory courses.