Grace: An Introductory Object-Oriented Language

Timothy Jones

Victoria University of Wellington, New Zealand
tim@ecs.vuw.ac.nz

Educating novice developers in the art of programming has always been a difficult task [2]. Instructors must balance the introduction of important concepts with the need to introduce real-world skills, and the requirements on learning outcomes differ wildly between courses and institutions. The result is that many courses teach enterprise languages such as Java or C++, which contain the necessary concepts but require the student to navigate syntax and semantics that are superfluous to their learning goals. The **public static void** main(String[] args) incantation is the bane of every CS101 instructor for whom Java is the only choice of language.

Tools such as BlueJ have attempted to mitigate such issues [3], but the core of the problem is that these languages have made trade-offs in the name of practicality that complicate early concepts. Some courses are transitioning to languages like Python in response, but in reality this just exchanges one set of problems and incantations for another: floating-point precision errors and oddly named definitions are initially confusing no matter which language they appear in. Using a dynamically-typed language as an introduction to programming also deprives students of experience with type annotations, which they will need to know when ultimately progressing to a language in the C family.

It is with these factors in mind that we have been developing *Grace*, a programming language aimed primarily at introductory programming courses [1], as part of a collaboration between Portland State University, Pomona College, and Victoria University of Wellington. The language aims to cover the many different (and often contradictory) preferences of instructors, but with a focus on object-orientation. The language can be used to introduce concepts — objects, classes, procedures, type annotations — whenever required by an instructor. It also employs techniques to remove the burden of configuration from the student so that they have all of the required tools immediately on hand.

The primary difficulty in the development of Grace has been uniting the contradictory expectations of instructors. The language has been designed by a committee of individuals with differing preferences, and a number of interesting design decisions have arisen from the compromises between these positions. This tension has influenced fundamental components of the language such as object inheritance, static errors, and the meaning of types, and continues to push the language in unexpected directions.

This talk will introduce Grace in the context of its educational goals, discussing interesting features that set it apart from existing object-oriented languages, including the gradual type system, the object-only model, and extensible pattern matching. The talk will also aim to highlight the problems that have arisen from attempting to reconcile the differing requirements of instructors, and the idiosyncrasies resulting from these trade-offs. For more information, the Grace blog can be found at `http://gracelang.org` and the platform and various tools are available at `http://ecs.vuw.ac.nz/~mwh/minigrace`.

References

[1] Black, A. P., Bruce, K. B., Homer, M., Noble, J., Ruskin, A., and Yannow, R. Seeking Grace: A new object-oriented language for novices. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2013), SIGCSE '13, ACM, pp. 129–134.

[2] Kölling, M. The problem of teaching object-oriented programming, Part I: Languages. *JOOP 11*, 8 (1999), 8–15.

[3] Kölling, M., Quig, B., Patterson, A., and Rosenberg, J. The BlueJ system and its pedagogy. *Journal of Computer Science Education 13*, 4 (Dec 2003).