Homework 2: Advanced OO Concepts and Language Design

Due: problem 1, October 21, with a meeting to discuss it beforehand before October 15.

The purpose of this homework is to explore language design in the OO paradigm. In addition, I hope this will increase your understanding of the tradeoffs in language design, and increase your critical thinking skills.

Problem 1 is to de done in groups. Follow the course policies, in particular, be sure to have a new leader, write out the certification required, and hand in a list of what worked well and what didn't.

Readings: See the books on reserve for the course, especially: *The Java Language Specification* and *The C++ Programming Language*.

1. (200 points) In this part you will examine the differences between the OO languages C++ and Java, and put into Smalltalk some of the good points of C++ and Java, while retaining its essential character. In particular, you are to:

   - Write a short summary of the goals of C++ and Java, and what differences there are between the goals of the two languages.
   - Write a short summary of the differences between the goals of Java and those of Smalltalk.
   - Write a description of the significant differences between C++ and Java. Focus on significant semantic differences, and not so much on syntactic issues. For each difference, describe why you think the difference arose.
   - Write a description of the main differences between Java and Smalltalk. Again, focus on significant semantic differences, and not so much on syntactic issues. You do not need to explain why these differences arose.
   - Describe a set of language design changes to Smalltalk that would incorporate the best features of C++ and Java into Smalltalk. For each change to Smalltalk, explain the benefits of the change, give the revised syntax, and explain the semantics by precise English descriptions and examples. Be sure to include examples of each change that justify the benefits you are claiming. Discuss any tradeoffs involved

     To limit the size of the changes, don't make changes to Smalltalk that aren't necessary, leave the basic syntax alone, and don't add a static type system to Smalltalk. Try to keep Smalltalk simple. Don't worry about changing minor details of Smalltalk, such as the semantics of floating point numbers, unless you can justify the importance of such changes.
   - Also describe a set of language design changes that you considered but did *not* make to Smalltalk, and briefly explain why you left them out. Include examples.
   - Describe what features of Smalltalk, if any, could profitably be adopted by Java.

2. (100 points, extra credit) This problem can be done either individually or in groups.

   Does Java support distributed programming well? Why or why not? If so, what features of Java support it? If not, what features should be added to Java to support it? Propose a syntax and semantics for any new features.