Homework 2: Object-Oriented Languages, Design, and Multimethods

Due:

The purpose of this homework is to explore some language design issues related to object-oriented programming, object-oriented design, and multimethods. Problems 1–6 are individual problems, and problems 7–8 are group problems. For the individual problems, please do the work on your own, not in groups. For the group problems, please follow the policy in section 6 of the course policies document.

For all Smalltalk programs, you must run your code with on some Smalltalk system. Hand in a printout of your code and your testing. (You can use the `script` command to give a transcript, or run Smalltalk from within an emacs shell to get test output captured, or you can use the Smalltalk printing primitives to print test output to a file.)

Don't write code that looks like C! We'll take points off for poor style.

Readings: "On Binary Methods" by Bruce, *et al.* [1], and "BeCecil, A Core Object-Oriented Language with Block Structure and Multimethods: Semantics and Typing" by Chambers and Leavens [3] for questions about binary methods. See the books by Budd [2] and by Goldberg and Robson [4] on reserve for more background on Smalltalk.

As always, if you direct quote or use the ideas of another paper, be sure to attribute your answers. (Use quotation marks and citation forms [5].) Failure to do this may result is substantial loss of points.

1. (a) (15 points) How does Little Smalltalk, as a language, differ from Smalltalk-80. That is, ignoring issues of the programming environment to the extent possible, what are the differences between the languages? (It may help to think in terms of syntax differences and semantic differences. You can just list these.)

   (b) (15 points) For each difference you identified, briefly state what practical effect that difference has on programming (in a sentence or two), and give an example.

2. (30 points, extra credit) One reason we use Little Smalltalk in 541 is that it used to be the only freely-available version of Smalltalk. See if you can find a better version out on "the network" and evaluate its suitability for use in this class by programming the Account and PlusAccount classes from homework 1 in that system. Hand in (a) a description of what system you used and your evaluation of it and (b) a printout of your code and testing using that system. (Part of your evaluation should be how hard it is to print code and test transcripts.)

   If you find a system you like, you are free to use that instead of Little Smalltalk for the rest of this homework.

3. Answer the following questions briefly (in a sentence or two). (You may have to look some of these terms up; that's part of this problem.)

   (a) (5 points) What is an appropriate notion of "type error" for Smalltalk?

   (b) (5 points) Is Smalltalk strongly typed? Why?

4. (30 points, total)

   (a) (15 points) Write something as close to the Account and PlusAccount classes from homework 1 as you can manage in the untyped part of BeCecil. (You can just do this on paper.) You can assume that the integers, floating point numbers, and strings with all their usual literals and operations (as generic function names), are available. Feel free to use all the syntactic sugars in BeCecil. Double check your syntax.

(b) (15 points) What differences do you notice from the Smalltalk version of these programs and the BeCecil version? Are there any practical difficulties you see from this example with BeCecil? Briefly explain and give an example of each.

5. (20 points, total, extra credit) A design alternative for BeCecil would be to take out storage tables, and acceptors, and in their place add "method replacement". This is discussed in appendix section B.10 of the Chambers and Leavens paper.

   (a) (5 points, extra credit) Could you desugar the use of acceptor in BeCecil into this version of the language?

   (b) (5 points, extra credit) Try programming an example, say cons, with the revised field update sugar and get an impression of how this looks. Does the extra syntax bother you?

   (c) (10 points, extra credit) Which do you think would be more efficient at run-time: storage tables or method replacement?

6. I like to design a T-shirt with fabric painted pictures, designs, examples, or slogans for various units of a course. But I've never done that for object-oriented programming or Smalltalk.

   (a) (15 points) Design a T-shirt for this unit of the course (on a piece of paper). It should have the following characteristics:

      - It must be able to fit on the front and back of a T-shirt, and so can't use a lot of words on either side.
      - It should somehow illustrate make the most important concepts or points of the unit. The "essence" of object-oriented programming.

   (b) (5 points) Briefly explain why you think your design illustrates the most important points in the unit.

— **end of the individual problems** —

— **start of the group problems** —

7. (80 points, total) Propose an extension to Smalltalk to handle some part of the binary method problem. You should focus on either providing access to other arguments of a binary method, or to preventing the errors that may occur when subclasses redefine a binary method.

   Group members should attempt the first 3 points below (individually!) before the first group meeting. Write down these ideas, as you will hand them in as an attachment to the group work at the end. After the first group meeting, meet with the course staff briefly to describe your ideas, and to check that you are on the right track.

   Your description should include the following.

   (a) (10 points) A context-free grammar that specifies any added syntax, or a specification of the names and protocol of any new classes and objects.

   (b) (30 points) A specification of the semantics of the proposal. That is, state what each added feature or piece of syntax does (or means).

   (c) (20 points) A small set of examples to illustrate the proposal.

   (d) (30 points) A discussion of the trade-offs involved in the proposal. What benefits does it give to Smalltalk, and what problems does it cause?

   (e) (10 points) Hand in the above, with the group leader's certification as described in section 6.3 of the course policies document. (Your individual notes on the first 3 points above serve the role of the "individual plans" called for there.)

8. (80 points, total) Design a class or set of classes to aid in the testing of Smalltalk code.

   The goals of this "test system" are as follows. It should be practically useful, and so should not require the writer of a test case to remember too many details. Running a test should be easy to do from the keyboard, without a mouse to do cutting and pasting; indeed it should be possible for a program to run a series of tests. Furthermore, to create a set of similar tests, one shouldn't have to duplicate a lot of code.

   Group members should attempt the first 3 points below (individually!) before the first group meeting. Write down these ideas, as you will hand them in as an attachment to the group work at the end. After the first group meeting, meet with the course staff briefly to describe your ideas, and to check that you are on the right track.

   Your description should include the following.

   (a) (10 points) A description of what the responsibilities of a testing system are. It may help to think of typical "scenarios" of testing and what the names of the main messages for testing should be.

   (b) (20 points) Document the overall design of your classes, either using the notation in the Gamma *et al.* book, or using the CRC card notation.

   (c) (10 points) Briefly describe how your design achieves the goals noted at the beginning of this problem.

   (d) (30 points) Write code for your test system, and give an example of its use in testing some other classes that you've written.

   (e) (10 points) Hand in the above, with the group leader's certification as described in section 6.3 of the course policies document. (Your individual notes on the first 3 points above serve the role of the "individual plans" called for there.)

References

[1] Kim Bruce, Luca Cardelli, Giuseppe Castagna, The Hopkins Object Group, Gary T. Leavens, and Benjamin Pierce. On binary methods. *Theory and Practice of Object Systems*, 1(3):221–242, 1995.

[2] Timothy Budd. *A Little Smalltalk*. Addison-Wesley, Reading, Mass., 1987.

[3] Craig Chambers and Gary T. Leavens. Bececil, a core object-oriented language with block structure and multimethods: Semantics and typing. Technical Report 96-17, Department of Computer Science, Iowa State University, 226 Atanasoff Hall, Ames, Iowa 50011, December 1996. Available by anonymous ftp from ftp.cs.iastate.edu, and by e-mail from almanac@cs.iastate.edu. Also University of Washington Department of Computer Science and Engineering TR number UW-CSE-96-12-02.

[4] Adele Goldberg and David Robson. *Smalltalk-80, The Language and its Implementation*. Addison-Wesley Publishing Co., Reading, Mass., 1983.

[5] Mary-Claire Van Leunen. *A handbook for scholars*. Oxford University Press, New York, N.Y., revised edition, 1992.