Spring, 1995 Name: ____________________

Com S 541 — Programming Languages 1

Test on Smalltalk and OOP

Special Directions for this Test

This test has 4 questions and pages numbered 1 through 5.
Before you begin, please take a moment to look over the entire test so that you can budget your time.
When you write programs on this test, you may use anything documented in the *Overview and Specification of the built-in types in Little Smalltalk*, TR91-22a. You are encouraged to define methods and classes not specifically asked for if they are useful to your programming.
Make it clear by indentation where methods start and stop.

General Directions from the "Course Specification"

Short answer questions will be graded on the basis of completeness (did you list all the relevant parts of the answer?) soundness (did you make any errors of fact or contradict yourself?) and clarity. You can help ensure clarity by giving examples where possible.
For essays, critical justifications, and the like there may be no one right answer; so the criteria for judgement will be whether your essay is:

- clear (you use examples where appropriate, you write clear sentences, diagrams, and you avoid excess verbosity),
- sound (you start from facts or reasonable assumptions, your logic is convincing), and
- complete (you consider relevant aspects of the problem and alternatives).

The highest grade will be earned by an answer that is, in addition, concise and elegant; that is, answers that say the most with the least amount of excess. Extra consideration will be given to answers that are especially creative. For programming problems, there is usually more than one right answer, hence it is important to present with your program evidence that it is correct. Answers to programming questions will be scored on the following points:

- elegance and clarity (conciseness, using relevant techniques to modularize the solution, making good use of language features, the creativity of your solution, and how clear and convincing your argument is that the code is correct)
- correctness (whether your program solves the given problem correctly in the given programming language)

You should try to eliminate all syntax errors from your programs, as they will affect both the correctness and clarity of your solution; you may receive no points for programs with major syntax errors. Run-time space and time efficiency of programs is not important, unless the problem says so. Also, checking of external inputs, prompting, labeling output, and other conventions for dealing with humans are not important (for this class), unless the problem says so.
The breakdown of points between these categories is roughly as follows.

- 50% for elegance and clarity (using relevant techniques to modularize the solution, making good use of language features, the creativity of your solution),
- 50% for correctness (whether your program solves the given problem correctly in the given subset of the given language)

You are encouraged to use auxiliary subroutines whenever they are helpful.

1. In this problem you will define a class `Counter`. The abstract value of a `Counter` object is an integer, which represents a count of things.

   (a) (5 points) What would you type to the Little Smalltalk system prompt to define the class `Counter`?

   (b) (5 points) Define the method `new` for this class, so that a `Count` object starts out with a count of zero (0) things.

   (c) (15 points) Using the convention that a completely blank line separates methods, write code for the methods specified below. (These methods are specified in the notation of TR #91-22.)

   `value` *self: Counter* $\rightarrow$ *v: Integer*
   Ensures: *v* is the count of *self*.
   `up` *self: Counter*
   Modifies at most: *self*.
   Effect: make the count of **post**(*self*) be **pre**(*self*) plus 1.

2. In this problem you will define a class that implements a type `Gauge`. The abstract value of a `Gauge` object consists of: a count of things (an Integer), and the maximum count seen over the lifetime of the object (an Integer).

   (a) (5 points) What would you type to the Little Smalltalk system prompt to define the class `Gauge`? You should do this in a way that makes it very easy to implement all the methods specified in the previous problem.

   (b) (5 points) Define the method **`new`** for this class so that the abstract value of a new `Gauge` object starts with its count of things starts the same as a `Counter` object (0), and so that the maximum count seen also starts at 0. Your code should avoid any duplication with the code for Counter, if possible.

   (c) (15 points) Using the convention that a completely blank line separates methods, write any code needed to define the methods specified below, as well as the method value as specified for `Counter` in the previous problem.

   down   *self: Gauge*

   Modifies at most: *self.*

   Effect: make the count of **post**(*self*) be the maximum of 0 and the count of **pre**(*self*) minus 1.

   up   *self: Gauge*

   Modifies at most: *self.*

   Effect: make the count of **post**(*self*) be 1 plus the count of **pre**(*self*); also make the maximum count seen of **post**(*self*) be the maximum of: the count of **post**(*self*) and the maximum count seen of **pre**(*self*).

   maxValueSeen   *self: Gauge* $\rightarrow$ *r: Integer*

   Ensures: *r* is the maximum count seen of *self.*

3. (40 points, total) (a, 10 points) Describe a specific example of inheritance in the INST interpreter. (State the classes involved, and what was inherited.)
(b, 30 points) Suppose one wanted to add a `while` loop to the statments of INST. Describe what changes you would have to make to the interpreter so that it can create and evaluate a `while` loop. Be as specific as you can; write the code for maximum points.

4. (30 points, total) In the context of Smalltalk: (a, 5 points) What is an abstract data type (ADT)? (b) Does the notion of an ADT differ from the notion of a class? Explain (5 points) and give an example (5 points). (c, 5 points) When is a type S as subtype of a type T? (d) Does the notion of a subtype differ from the notion of a subclass? Explain (5 points) and give an example (5 points).