

Homework 3

CSC 4101, Fall 2009

Due: 27 October 2009

1. (10 pts)

Suppose we had support for multimethods in Java instead of or in addition to overloading. Explain how you would implement the pretty printer of Project 1 using multimethods instead of having a `print()` method in each class of the hierarchy. What would be the advantage of using multimethods? You can simply use the same syntax as for overloading and indicate where the multimethod dispatch happens.

2. (10 pts)

Suppose we have the following class:

```
class C { /* ... */ public: int foo (); int bar (); };
```

Also suppose that this class is only available in compiled form, i.e., that it cannot be modified. Now we want to add an alternative implementation that only has a method `foo()` but that does not have a method `bar()` and does not inherit the data members from class C. I.e., this new class D would have the structure

```
class D /* ... */ { public: int foo (); };
```

but it cannot inherit from class C.

Suppose we want to maintain a list of objects that can be from either classes C or D. Explain how you can design this list data structure, traverse the list, and call `foo()` on each element, such that the appropriate method `C::foo()` or `D::foo()` is executed depending on the type of the object. Do not use either if-statements or structural subtyping. Hint: There are at least three possible solutions; each one requires you to define two or more classes in addition to the list class.

3. (10 pts)

Given the following Java code:

```
class C {
    public int foo(C p) { return 1; }
}

class D extends C {
    public int foo(C p) { return 2; }
    public int foo(D p) { return 3; }
}
```

```

C p = new C();
C q = new D();
D r = new D();

int i = p.foo(r);
int j = q.foo(q);
int k = q.foo(r);

```

(Remember that in Java every object is accessed through a pointer and that methods are virtual by default.)

Which methods get called in the three calls, i.e., what are the values of *i*, *j*, and *k*? Explain how the method selection works to result in these values.

4. (10 pts)

Explain what the following Scheme code is doing:

```

(define (make-stream n f)
  (define (next m)
    (cons m (lambda () (next (f m)))))
  (next n))
(define head car)
(define (tail stream)
  ((cdr stream)))
(define (nth stream n)
  (if (= n 0) (head stream)
      (nth (tail stream) (- n 1))))
(define even (make-stream 0 (lambda (n) (+ n 2))))

```

Try it out in Scheme48 and check the values of the following expressions:

```

even
(head even)
(head (tail even))
(head (tail (tail even)))
(head (tail (tail (tail even))))
(nth even 5)
(nth even 1000)

```

Explain what the `lambda` in `make-stream` is good for, where this function is called, and how `tail` and `nth` work. To see what's going on, trace manually through the execution of

```

(head (tail (tail even)))

```