Weekly exercises in INF3110 – OO-I

Week 43, 2008

Problem 1

Here is an easy one on type compatibility.

Given the following program fragment in some hypothetical language:

```
type S1 is struct {
      int y;
      int w;
};
type S2 is struct {
      int y;
      int w;
};
type S3 is struct {
      int y;
};
S3 f(S1 p) { ... };
...
S1 a, x;
S2 b;
S3 c;
int d;
...

a = b;      // (1)
x = a;      // (2)
c = f(b);   // (3)
d = f(a);   // (4)
```

a) Under name compatibility, which of the four statements (1) ... (4) are type correct (and which are not).

b) Same question under structural compatibility.

Problem 2

We have the following classes:

```
  class Food {...}
  class Cheese extends Food {...}
```

Assume that we have the following functions:

```
  int f(c Cheese) {...}

  int f'(f Food) {...}
```

someFood is a value of type Food, and someCheese is a value of type Cheese. Then we know that

f’(someCheese) can be substituted for f(someCheese)

that is, whenever we have a call ‘f(someCheese)’ we may just as well call f’ with the same someCheese parameter without causing any static type errors: f’ can be said to be a subtype of f.

Why cannot f(someFood) be substituted for f’(someFood)? That is why can not f said to be a subtype of f’? Give an example of class Cheese (that is a more elaborate Cheese than above) and a definition of f that will create a type error.

Problem 3

Exercise 11.1 in Mitchell.

11.1 Simula Inheritance and Access Links

In Simula, a class is a procedure that returns a pointer to its activation record. Simula prefixed classes were a precursor to C++ -derived classes, providing a form of inheritance. This question asks about how inheritance might work in an early version of Simula, assuming that the standard static scoping mechanism associated with activation records is used to link the derived class part of an object with the base class part of the object.

Sample Point and ColorPt classes are given in the text. For the purpose of this problem, assume that if cp is a ColorPt object, consisting of a Point activation record followed by a ColorPt activation record, the access link of the parent-class (Point) activation record points to the activation record of the scope in which the class declaration occurs, and the access link of the child-class (ColorPt) activation record points to activation record of the parent class.

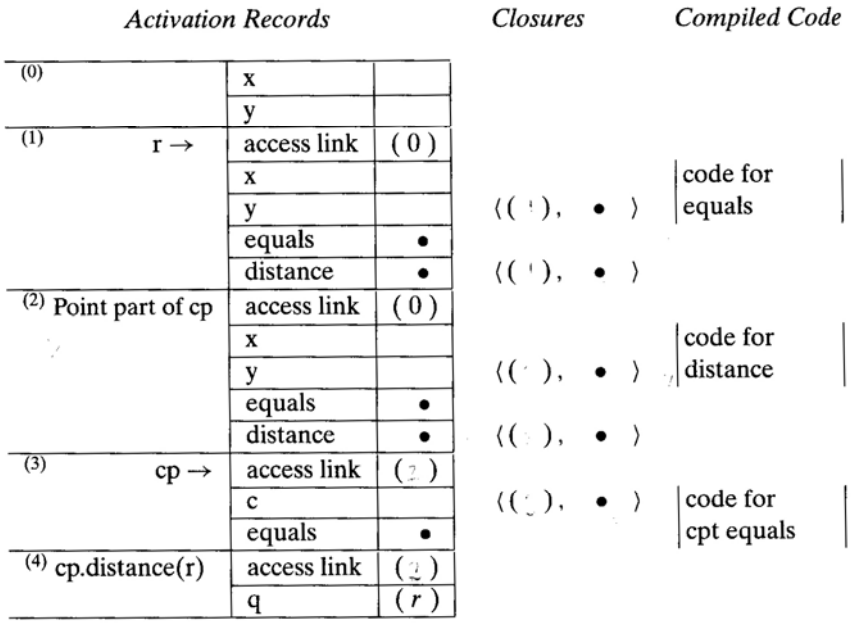
(a) Fill in the missing information in the following activation records, created by execution of the following code:

```
ref(Point) p;
ref(ColorPt) cp;
r :- new Point(2.7, 4.2);
cp :- new ColorPt(3.6, 4.9, red);
cp.distance(r);
```

Remember that function values are represented by closures and that a closure is a pair consisting of an environment (pointer to an activation record) and a compiled code.

In the following illustration, a bullet (•) indicates that a pointer should be drawn from this slot to the appropriate closure or compiled code. Because the

pointers to activation records cross and could become difficult to read, each activation record is numbered at the far left. In each activation record, place the number of the activation record of the statically enclosing scope in the slot labeled "access link." The first two are done for you. Also use activation record numbers for the environment pointer part of each closure pair. Write the values of local variables and function parameters directly in the activation records.



(b) The body of distance contains the expression

```
sqrt(( x - q.x )**2 + (y - q.y) ** 2)
```

that compares the coordinates of the point containing this distance procedure to the coordinate of the point q passed as an argument. Explain how the value of x is found when cp.distance(r) is executed. Mention specific pointers in your diagram. What value of x is used?

(c) This illustration shows that a reference cp to a colored point object points to the ColorPt part of the object. Assuming this implementation, explain how the expression cp.x can be evaluated. Explain the steps used to find the right x value on the stack, starting by following the pointer cp to activation record (3).

(d) Explain why the call cp.distance(r) needs access to only the Point part of cp and not the ColorPt part of cp.

(e) If you were implementing Simula, would you place the activation records representing objects r and cp on the stack, as shown here? Explain briefly why you might consider allocating memory for them elsewhere.

Problem 4

Consider the classes C and SC at slide 29 from 29.10.2007.

We know that this language allows overloaded methods to be inherited, that is the scope for overloaded methods for a subclass includes the inherited methods.

Here is the answer to the question posed at the lecture (to which method are the different calls bound):

```
C c     = new C();
SC sc   = new SC();
C c'    = new SC();

c.equals(c)    //1    equals 1
c.equals(c')   //2    equals 1
c.equals(sc)   //3    equals 1

c'.equals(c)   //4    equals 1
c'.equals(c')  //5    equals 1
c'.equals(sc)  //6    equals 1

sc.equals(c)   //7    equals 1
sc.equals(c')  //8    equals 1
sc.equals(sc)  //9    equals 2
```

It is only in //9 that the equals 2 method is called, the reason being that overloading is resolved at compile time. The three calls to c' (even though the value of c' is a SC-object) will be calls to equals 1. //7 is also a call to equals 1, as the parameter c is of type C – same with //8.

The method equals 1 comes in two versions: the C_equals 1 and the redefined SC_equals 1.

a) Indicate for the above first 8 cases which of the equals 1 are called.

b) Now, suppose that class SC does not have the first equals method, the one with parameter of type C overriding the equals from class C. Determine which of the remaining methods is executed for each of these 8 cases:

```
c.equals(c)    //1
c.equals(c')   //2
c.equals(sc)   //3

c'.equals(c)   //4
c'.equals(c')  //5
c'.equals(sc)  //6

sc.equals(c)   //7
sc.equals(c')  //8
```

Problem 5

a) Write in Java both an abstract data type and a class for the data type Date, with year, month and day, operations before and after and daysBetween. In the abstract data type the operations before, after and daysBetween shall take two Dates, while the operations for the class Date shall have just one Date parameter.

b) There is on ‘obvious’ way of doing this, where Date is depending on how year, month and day is represented (e.g. as int variables). How would you make Date independent of this representation?