INF3110

Solutions OO-I, week 43, 2008

Problem 1

a) Name compatible

(1) type error
(2) type correct
(3) type error
(4) type error

b) Structural compatible

(1) type correct
(2) type correct
(3) type correct
(4) type error

Problem 2

Assume that class Cheese adds a method melt:

```
class Cheese extends Food {void melt(){...};}
```

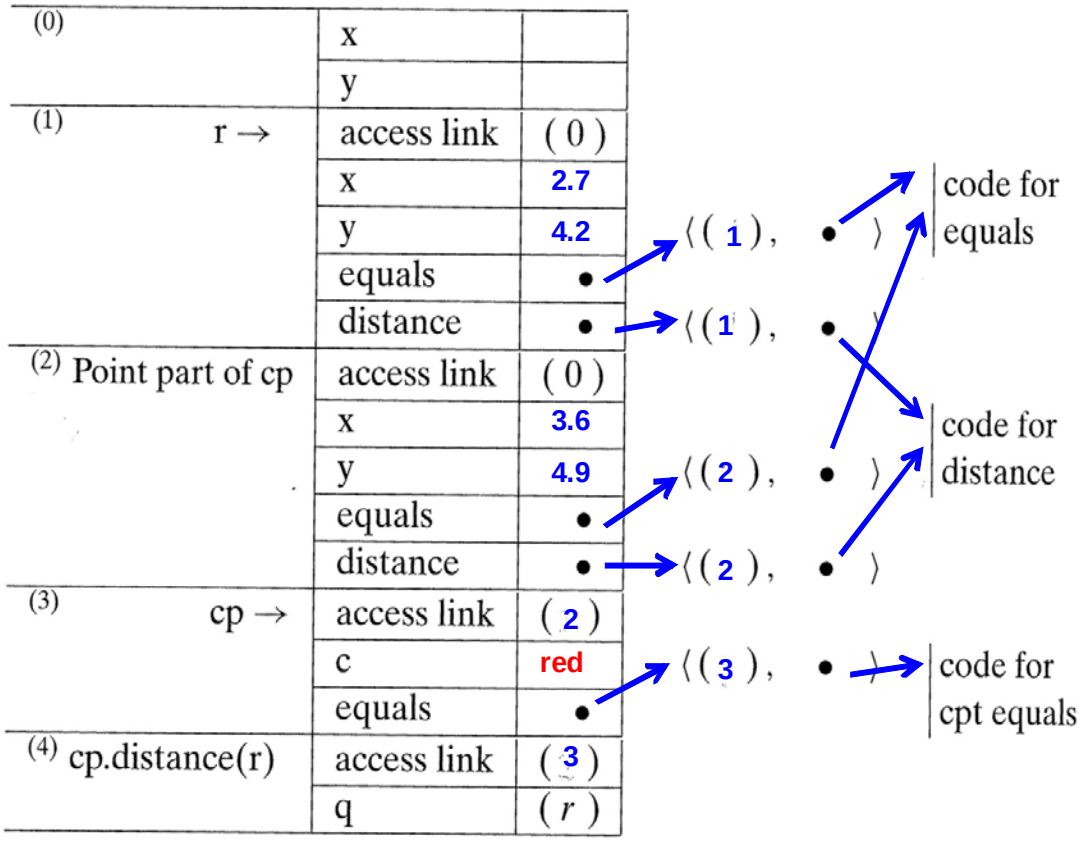
Now, define f so that it uses the fact that the parameter is of type Cheese:

```
void f(c Cheese) {...; c.melt; ...}
```

It would be a bad idea to call f with something that is not of type Cheese.

Problem 3 Exercise 11.1 in Mitchell.

a)



b)

The x of (2) is used.
Which pointers are used:

1. the access link from (4) to (3)
2. the access link from (3) to (2)

alternatively

1. access link form the cp instance (3) to (2) in order to get the distance closure
2. from this closure, follow the link (2) and get to the x of (2)

c)
cp.x is found by following the static link from cp (3) to (2)

d)
distance is inherited from Point and not redefined for Colorpoint, so it does not use the c variable.

e)
The code of the exercise may be executed as part of a procedure, whose activation record will be on the top of the stack when the activation

records for r and cp are allocated – and can then not be de-allocated when the procedure exits.

Problem 4

a)

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

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

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

// 7 is equals 1, because SC has an inherited (C,C)-typed equals. equals 2 requires a SC typed parameter. Both c and c' may at runtime denote a SC-object, but it is only safe to assume that they denote C-objects

// 8 is equals 1, with the same reasoning as for //7

b)

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

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

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

Problem 5

Sketch of a solution

a)

```
class Date {
  int year, month, day;
  Date date(int y, m, d) {.. ; return new Date(...) ; ..}

  boolean static before(Date d1,d2) {
    if (d1.year < d2.year) {return true} else
    if (d1.year > d2.year) {return false} else
      if (d1.month < d2.month) {return true} else
      if (d1.month > d2.month) {return false} else
        if (d1.day < d2.day) {return true} else {return false};
  };
  ..
}
```

```
class Date {
  int year, month, day;
  void Date(int y, m, d) {..}

    boolean before(Date d) {
      if (year < d.year) {return true} else
      if (year > d.year) {return false} else
        if (month < d.month) {return true} else
        if (month > d.month) {return false} else
          if (day < d.day) {return true} else {return false};
  };
  ..
}
```

b) – only shown for one of the cases

```
class Month {
  int mAsInt;
  Boolean before(Month m) {
    if (m < m.mAsInt) then return true else return false;
  }
};
class Year {/* correspondingly */};
class Day {/* correspondingly */};

class Date {
  Year year; Month month, Day day;
  Date date(Year y, Month m, Day d) {..}

  boolean static before(Date d1,d2) {
    if Year.before(d1.year, d2.year) {return true} else
      if Month.before(d1.month, d2.month) {return true} else
        if Day.before(d1.day, d2.day) {return true}
        else {return false};
  }
}
```