Worksheet 17: Michael Collins

1. Assume you have the NAMED-OBJECT and THING classes defined as in OBJTYPES.SCM for the project. Define a new class PERSON which inherits from the THING class. It should have an additional state variable, AGE. When created, we should state the person’s name, age and location, for example:

   (define mike (create-person 'mike 21 G484))

creates a mike object with his correct age, and location G484 (assuming that G484 has previously been defined as a location). The PERSON class should have additional methods age? and have-birthday. The method age? returns the age of the object, have-birthday increments the age by 1 year, and returns the new age.

   (define (create-person name age location)
     (create-instance person name age location))
   (define (person self name age location)
     (let ((thing-part (thing self name location)))
       (make-handler
         'person
         (make-methods
           'AGE? (lambda () age)
           'HAVE-BIRTHDAY (lambda () (set! age (+ age 1))
                          age))
       thing-part)))

2. Next, create a new BUILDING class, where:

   • The BUILDING class inherits directly from the NAMED-OBJECT class.
   • The internal state variables for the building are a name, an x-coordinate, a y-coordinate, and an architect variable which is of the PERSON type.
   • The class should contain methods for: X-COORD, Y-COORD, SET-X!, SET-Y!, ARCHITECT-NAME. In addition the class should have a method DESCRIBE which prints “I am the <name> building, located at (<x-coordinate>, <y-coordinate>)”. As a final point, SET-X! and SET-Y! should be implemented so that they change the coordinate, then make a call to DESCRIBE.

   (define (create-building name x y architect)
     (create-instance building name x y architect))
   (define (building self name x y architect)
     (let ((named-part (named-object self name)))
       (make-handler
         'building
         (make-methods
           'X-COORD (lambda x)
           'Y-COORD (lambda y)
           'SET-X! (lambda (val) (set! x val) (ask self 'describe))
           'SET-Y! (lambda (val) (set! y val) (ask self 'describe))
           'ARCH-NAME (lambda () (ask architect 'name))
           'DESCRIBE (lambda () (display ''I am ...'')))
       named-part))))
2. Now create a GEHRY-BUILDING class. It should have the BUILDING class as its superclass. These objects always have gehry (a previously defined person) as their architect. The new class should have a z-coordinate as an additional internal variable (Gehry buildings seem to have an additional piece of uncertainty concerning their location). Another internal variable, initially set to 0, is num-fire-alarm-tests. It has an additional method TEST-FIRE-ALARM which displays a message “Testing fire alarm” and increments the num-fire-alarm-tests variable. The DESCRIBE method should behave the same way as for the BUILDING class, but in addition it should report the z-coordinate, the number of fire alarm tests, followed by a line “I was designed by a famous architect.” What happens if the SET-X! or SET-Y! methods are used?

(define (create-gehry-building name x y z)  
  (create-instance gehry-building name x y z))
(define (gehry-building self name x y z)  
  (let ((building-part (building self name x y gehry))
    (num-fa 0))
    (make-handler
     'gehry-building
     (make-methods
      'TEST-FIRE-ALARM (lambda () (set! num-fa (+ num-fa 1))
      'DESCRIBE (lambda () (ask building-part 'describe)
        (display ''Z-coord is '')
        (display z)
        (display ''NUM-FA is '')
        (display num-fa)
        (display ''I was designed by...'')))
    building-part)))