```
; I added two new procedures (HAS-A and HAS-A-THING-NAMED)
;in the person class.
;my code:
;person inherits from root-object, so it has IS-A method. IS-A returns
;#f if the argument passed in doesn't have the type specified.
;In this case, I use 'IS-A to test whether x has the specified type.
;If it does, don't filter it out. If it doesn't, filter it out.
'HAS-A
(lambda (type)
(let ((possession (ask self 'THINGS)))
(filter (lambda (x) (ask x 'IS-A type))  possession)))

;I also used a filter method to filter out things that don't have the name I specified.
'HAS-A-THING-NAMED
(lambda (named-thing)
(let ((possession (ask self 'THINGS)))
(filter (lambda(x) (eq? (ask x 'NAME) named-thing)) possession)))

; ;test cases for 'HAS-A and 'HAS-A-THING-NAMED
; ready
; > (ask me 'LOOK-AROUND)
;
; You are in eecs-ug-office
; You are not holding anything.
; You see stuff in the room: BOIL-SPELL
; You see other people: lambda-man
; The exits are in directions: east OK
; > (ask me 'HAS-A-THING-NAMED 'BOIL-SPELL)
; ()
; > (ask me 'TAKE (thing-named 'BOIL-SPELL))
;
;  At  eecs-ug-office  tina  says  --  I  take  BOIL-SPELL  from  eecs-ug-office  (instance
#<procedure:handler>)
; > (ask me 'HAS-A-THING-NAMED 'BOIL-SPELL)
; ((instance #<procedure:handler>))
; > (ask me 'HAS-A 'SPELL)
; ((instance #<procedure:handler>))
; > (ask me 'HAS-A 'THING)
; ((instance #<procedure:handler>))
; > (ask me 'HAS-A 'PERSON)
; ()
; >
```

I added a method MARAUDERS-MAP to the AVATAR class.

```
;(all-people) returns a list of people in the game. I apply the same
;procedure to each person.
;I get out the person's name and location, and put them into "someone is
;at somewhere" format.
     'MARAUDERS-MAP
     (lambda ()
       (for-each (lambda(person) (display (append (list (ask person 'NAME)) '(is at) (list
(ask (ask person 'LOCATION) 'NAME))))
        (newline))  (all-people)))

;  ;Here are the output of the test:
;     ready
; >  (ask me 'MARAUDERS-MAP)
; (alyssa-hacker is at grendels-den)
; (mr-bigglesworth is at barker-library)
; (course-6-frosh is at barker-library)
; (registrar is at lobby-7)
; (tina is at building-13)
; (grendel is at building-13)
; (ben-bitdiddle is at great-court)
; (dr-evil is at graduation-stage)
; (lambda-man is at graduation-stage)
; > (ask (ask me 'LOCATION) 'NAME)
; building-13
; >

;the make-obfuscation-cloaks class. It inherits from the mobile-thing class, but doesn't
accept any new types of messages.
(define (create-obfuscation-cloak name location)
  (create-instance obfuscation-cloak name location))

(define (obfuscation-cloak self name location)
  (let ((mobile-part (mobile-thing self name location)))
    (make-handler
     'OBFUSCATION-CLOAK
     (make-methods) mobile-part)))


;I changed the PEOPLE-AROUND method in the person class.
;use filter to filter out the people with 'OBFUSCATION-CLOAK. If they have it, filter those
people out of the list
     'PEOPLE-AROUND       ; other people in room...
      (lambda ()
     (filter (lambda (person) (if (null? (ask person 'HAS-A 'OBFUSCATION-CLOAK)) #t #f))
```

```
(delq self (find-all (ask self 'LOCATION) 'PERSON))))   ;filter me out of the list first

; ;result of test cases. Please note that I just put part of the output here to demonstrate the effectiveness of my obfuscation-cloak object
; > (ask me 'MARAUDERS-MAP)
; (registrar is at eecs-hq)
; (course-6-frosh is at eecs-hq)
; (dr-evil is at building-13)
; (lambda-man is at bexley)
; (alyssa-hacker is at bexley)
; (ben-bitdiddle is at bexley)
; (grendel is at baker)
; (tina is at graduation-stage)
; (mr-bigglesworth is at graduation-stage)
; > (ask me 'PEOPLE-AROUND)
; ((instance #<procedure:handler>))  ;this demonstrates that I can see people. I see mr-bigglesworth, because we are both at graduation-stage
;
;  At  legal-seafood  mr-bigglesworth  says  --  I  take  OBFUSCATION-CLOAK-8  from legal-seafood ;mr-bigglesworth picked up the OBFUSCATION-CLOAK-8
;
; > (ask me 'MARAUDERS-MAP)
; (course-6-frosh is at eecs-hq)
; (registrar is at edgerton-hall)
; (mr-bigglesworth is at building-13)
; (ben-bitdiddle is at building-13)
; (tina is at building-13)
; (grendel is at student-center)
; (alyssa-hacker is at bexley)
; (dr-evil is at baker)
; > (ask me 'PEOPLE-AROUND)
; ((instance #<procedure:handler>))    ;this demonstrates that though mr-bigglesworth, ben-bitdiddle and I are all in building-13, I can only see ben-bitdiddle, because mr-bigglesworth is carrying the OBFUSCATION-CLOACK-8
;;

 (define (create-wand name location)
  (create-instance wand name location))

(define (wand self name location)
  (let ((mobile-part (mobile-thing self name location)))
    (make-handler    ;the wand class inherit from the mobile-thing class
     'WAND
     (make-methods
```

```
      'ZAP (lambda (target)
        (let ((caster (ask self 'LOCATION)))
         (if (eq? (ask caster 'IS-A 'PERSON) #f)
           (ask self 'EMIT (list "the wand cannot be waved, because it's not possessed by
anyone"))  ;handle the case the wand isn't owned by anyone
    (let ((spells (filter (lambda (x) (not (eq? (ask x 'IS-A 'SPELL) #f)))
                                        (ask caster 'THINGS))))
   ;gets out all the spells possessed by the caster
      (if (null? spells) (ask caster 'EMIT (list (ask caster 'NAME) " is waving the wand,
but nothing happens, because no spell is available to use"))   ;if there's no spell possessed
by the caster
      (let ((spell (list-ref spells (random (length spells)))))
   ;picks up a random spell from the caster's possession
        (ask caster 'EMIT (list (ask caster 'NAME) "is waving the wand and saying " (ask
spell 'INCANT)))
         (ask spell 'USE caster target)))))))

  'WAVE (lambda()
       (let ((caster (ask self 'LOCATION)))
         (if (eq? (ask caster 'IS-A 'PERSON) #f)
                  ;if the wand is possessed by anyone
         (ask self 'EMIT (list "the wand can't be waved because it's not possessed by anyone"))
         (let ((things-around  (filter (lambda (person) (not (equal? caster person)))
(append (ask caster 'PEOPLE-AROUND) (ask caster 'STUFF-AROUND)))))
 ;waving the wand should apply to anything (objects and people) other than
;myself
      (if (null? things-around) (ask self 'EMIT (list "there's nobody or nothing other than
myself in the room. The wand is waved dramatically, but no one is hurt."))  ;if there's
no one or nothing other than myself in the room, nothing should happen
      (ask self 'ZAP (list-ref things-around (random (length things-around))))))))))) ;if
there's at least a person or a thing, pick a random target as the target
    mobile-part)))

;test result:
;ready
;> (ask me 'LOOK-AROUND)
;
;You are in eecs-ug-office
;You are not holding anything.
;You see stuff in the room: SLUG-SPELL
;You see other people: alyssa-hacker
;The exits are in directions: east OK
;> (create-wand 'WAND (ask me 'LOCATION))
;(instance #<procedure:handler>)
```

```
;> (ask (thing-named 'WAND) 'ZAP (car (ask me 'PEOPLE-AROUND)))  ;wand cannot do anything unless there's a person carrying it
;
;At eecs-ug-office the wand cannot be waved, because it's not possessed by anyone MESSAGE-DISPLAYED
;> (ask (thing-named 'WAND) 'WAVE)
;
;At eecs-ug-office the wand can't be waved because it's not possessed by anyone MESSAGE-DISPLAYED
;> (ask me 'TAKE (thing-named 'WAND))
;
;At eecs-ug-office tina says -- I take WAND from eecs-ug-office (instance #<procedure:handler>)
;> (ask (thing-named 'WAND) 'ZAP (car (ask me 'PEOPLE-AROUND)))   ;the wand cannot do anything unless there are available spells
;
;At eecs-ug-office tina  is waving the wand, but nothing happens, because no spell is available to use MESSAGE-DISPLAYED
;> (ask (thing-named 'WAND) 'WAVE)
;
;At eecs-ug-office tina  is waving the wand, but nothing happens, because no spell is available to use MESSAGE-DISPLAYED
;> (ask me 'TAKE (thing-named 'SLUG-SPELL))     ;take the spells
;
;At eecs-ug-office tina says -- I take SLUG-SPELL from eecs-ug-office (instance #<procedure:handler>)
;> (ask (thing-named 'WAND) 'ZAP (car (ask me 'PEOPLE-AROUND)))
;
;At eecs-ug-office tina is waving the wand and saying  dagnabbit ekaterin    ;the wand is working!
;At eecs-ug-office A slug comes out of alyssa-hacker 's mouth. (instance #<procedure:handler>)
;> (ask (thing-named 'WAND) 'WAVE)
;;
;another test to demonstrate that the wand can be applied to multiple people, and the spells are chosen randomly
;ready
;> (ask me 'LOOK-AROUND)
;
;You are in 10-250
;You are not holding anything.
;You see stuff in the room: SLUG-SPELL OBFUSCATION-CLOAK-1 recitation-problem problem-set blackboard
;There are no other people around you.
```

```
;The exits are in directions: up down OK
;> (ask me 'TAKE (thing-named 'SLUG-SPELL))
;
;At 10-250 tina says -- I take SLUG-SPELL from 10-250 (instance #<procedure:handler>)
;> (create-person 'a-victim (ask me 'LOCATION))
;(instance #<procedure:handler>)
;> (create-person 'victim-2 (ask me 'LOCATION))
;(instance #<procedure:handler>)
;> (clone-spell (car (ask chamber-of-stata 'THINGS)) me)
;(instance #<procedure:handler>)
;> (ask me 'LOOK-AROUND)
;
;You are in 10-250
;You are holding: SLUG-SPELL SLUG-SPELL
;You see stuff in the room: OBFUSCATION-CLOAK-1 recitation-problem problem-set blackboard
;You see other people: victim-2 a-victim
;The exits are in directions: up down OK
;> (clone-spell (cadr (ask chamber-of-stata 'THINGS)) me)
;(instance #<procedure:handler>)
;> (ask me 'LOOK-AROUND)
;
;You are in 10-250
;You are holding: BOIL-SPELL SLUG-SPELL SLUG-SPELL
;You see stuff in the room: OBFUSCATION-CLOAK-1 recitation-problem problem-set blackboard
;You see other people: victim-2 a-victim
;The exits are in directions: up down OK

;> (create-wand 'WAND (ask me 'LOCATION))
;(instance #<procedure:handler>)
;> (ask me 'TAKE (thing-named 'WAND))
;
;At 10-250 tina says -- I take WAND from 10-250 (instance #<procedure:handler>)
;> (ask (thing-named 'WAND) 'WAVE)
;
;At 10-250 tina is waving the wand and saying  dagnabbit ekaterin
;At 10-250 A slug comes out of OBFUSCATION-CLOAK-1 's mouth. (instance
#<procedure:handler>)  ;waving the wand can be applied to objects
;> (ask (thing-named 'WAND) 'WAVE)
;
;At 10-250 tina is waving the wand and saying  habooic katarnum
;At 10-250 recitation-problem grows boils on their nose MESSAGE-DISPLAYED
;> (ask (thing-named 'WAND) 'WAVE)  ;the spells are chosen at random
;
;At 10-250 tina is waving the wand and saying  dagnabbit ekaterin
```

```
;At    10-250    A    slug    comes    out    of    SLUG    's    mouth.    (instance
#<procedure:handler>)              ;the targets are chosen at random
;> (ask (thing-named 'WAND) 'WAVE)
;
;At 10-250 tina is waving the wand and saying  dagnabbit ekaterin
;At 10-250 A slug comes out of recitation-problem 's mouth. (instance #<procedure:handler>)
;> (ask (thing-named 'WAND) 'WAVE)
;
;At 10-250 tina is waving the wand and saying  habooic katarnum
;At 10-250 victim-2 grows boils on their nose MESSAGE-DISPLAYED
;> (ask (thing-named 'WAND) 'WAVE)
;
;At 10-250 tina is waving the wand and saying  habooic katarnum
;At 10-250 SLUG grows boils on their nose MESSAGE-DISPLAYED
;>
```

I modified the instantiate-spell method. The red words are what got changed.

```
(define (instantiate-spells)
  (let ((chamber (create-place 'chamber-of-stata)))
    (let ((sp1
           (create-spell
            'BOIL-SPELL
            chamber
            "habooic katarnum"
            (lambda (caster target)
              (if (ask target 'IS-A 'PERSON)   ;check to see if the target
;is a person
                (ask target 'EMIT
                     (list (ask target 'NAME) "grows boils on their nose"))))))
          (sp2
           (create-spell
            'SLUG-SPELL
            chamber
            "dagnabbit ekaterin"
            (lambda (caster target)
              (if (ask target 'IS-A 'PERSON)
              (begin (ask target 'EMIT (list "A slug comes out of" (ask target 'NAME) "'s
mouth."))
              (create-mobile-thing 'SLUG (ask target 'LOCATION)))))))
          (sp3
           (create-spell
            'ZEPHYR-FROM-THE-DARK
            chamber
```

```
          "dark white dark wooooh"
          (lambda (caster target)
            (map (lambda (person) (ask person 'SUFFER (random 3) caster)) (ask caster
'PEOPLE-AROUND))  ;the spell is applied to every person besides the caster in the room
            (map (lambda (object)
               (if (> (random 4) 2)
               (ask object 'DESTROY))) (ask caster 'STUFF-AROUND)) ))))
  ;25% of the chance the spell will destroy objects in the room
                     chamber)))

; ;test results after modifying the two spells so that they can only be applied to people
;     > (ask me 'LOOK-AROUND)
;
;You are in edgerton-hall
;You are holding: WAND SLUG-SPELL BOIL-SPELL
;You see stuff in the room: OBFUSCATION-CLOAK-9
;You see other people: person-2 person-1
;The exits are in directions: north up south OK
;> (ask (thing-named 'WAND) 'WAVE)
;
;At edgerton-hall tina is waving the wand and saying  dagnabbit ekaterin
;At edgerton-hall A slug comes out of person-1 's mouth. (instance #<procedure:handler>)
;> (ask (thing-named 'WAND) 'WAVE)
;
;At edgerton-hall tina is waving the wand and saying  dagnabbit ekaterin
      ;if the target isn't a person, there's no effect
;> (ask (thing-named 'WAND) 'WAVE)
;
;At edgerton-hall tina is waving the wand and saying  habooic katarnum
;At edgerton-hall person-1 grows boils on their nose MESSAGE-DISPLAYED
;> (ask (thing-named 'WAND) 'WAVE)
;
;At edgerton-hall tina is waving the wand and saying  habooic katarnum
;At edgerton-hall person-1 grows boils on their nose MESSAGE-DISPLAYED
;>

;;test code for the new spell
;
;ready
;> (clone-spell (car (ask chamber-of-stata 'THINGS)) me)
;(instance #<procedure:handler>)
;> (ask me 'LOOK-AROUND)
;
;You are in building-13
```

```
;You are holding: ZEPHYR-FROM-THE-DARK
;You see stuff in the room: SLUG-SPELL
;There are no other people around you.
;The exits are in directions: north south OK
;> (create-person 'person (ask me 'LOCATION))
;(instance #<procedure:handler>)
;> (create-wand 'WAND (ask me 'LOCATION))
;(instance #<procedure:handler>)
;> (ask me 'TAKE (thing-named 'WAND))
;
;At building-13 tina says -- I take WAND from building-13 (instance #<procedure:handler>)
;> (create-mobile-thing 'object1 (ask me 'LOCATION))
;(instance #<procedure:handler>)
;> (create-mobile-thing 'object2 (ask me 'LOCATION))
;(instance #<procedure:handler>)
;> (ask (thing-named 'WAND) 'WAVE)
;
;At building-13 tina is waving the wand and saying  dark white dark wooooh     ;hit the person
;At building-13 person says -- Ouch! 0 hits is more than I want! (DONE #<void> #<void>)
;> (ask me 'LOOK-AROUND)
;
;You are in building-13
;You are holding: WAND ZEPHYR-FROM-THE-DARK
;You see stuff in the room: object1 SLUG-SPELL    ;object2 got destroyed
;You see other people: person
;The exits are in directions: north south OK
;> (ask (thing-named 'WAND) 'WAVE)
;
;At building-13 tina is waving the wand and saying  dark white dark wooooh
;At building-13 person says -- Ouch! 2 hits is more than I want! (#<void> #<void>)   ;hit
the person again
;> (ask me 'LOOK-AROUND)
;
;You are in building-13
;You are holding: WAND ZEPHYR-FROM-THE-DARK
;You see stuff in the room: object1 SLUG-SPELL
;You see other people: person
;The exits are in directions: north south OK
;> (ask (thing-named 'WAND) 'WAVE)
;
;At building-13 tina is waving the wand and saying  dark white dark wooooh
;At building-13 person says -- Ouch! 1 hits is more than I want!
;An earth-shattering, soul-piercing scream is heard... (#<void> #<void>)
;> (ask me 'LOOK-AROUND)
```

```
;
;You are in building-13
;You are holding: WAND ZEPHYR-FROM-THE-DARK
;You see stuff in the room: object1 SLUG-SPELL
;There are no other people around you.
;The exits are in directions: north south OK
;> (ask (thing-named 'WAND) 'WAVE)
;
;At building-13 tina is waving the wand and saying  dark white dark wooooh (#<void> DONE)
;> (ask me 'LOOK-AROUND)
;
;You are in building-13
;You are holding: WAND ZEPHYR-FROM-THE-DARK     ;SLUG-SPELL got destroyed
;You see stuff in the room: object1
;There are no other people around you.
;The exits are in directions: north south OK


;Assume the question means at every clock tick, all the wit students want to cast spells
(define (create-wit-student name birthplace activity miserly)
  (create-instance wit-student name birthplace activity miserly))

(define (wit-student self name birthplace activity miserly)
  (let  ((autonomous-person-part  (autonomous-person  self  name  birthplace  activity
miserly)))  ;it inherit from the autonomous-person class
    (make-handler
     'WIT-STUDENT
     (make-methods
      'INSTALL
      (lambda ()
       (ask autonomous-person-part 'INSTALL)
        (create-wand 'WAND self)      ;initiate the wit-student with a wand
        (clone-spell (car (ask chamber-of-stata 'THINGS)) self)
        (clone-spell (cadr (ask chamber-of-stata 'THINGS)) self)   ;equip the person with
at least one kind of spell
     (ask our-clock 'ADD-CALLBACK
         (create-clock-callback 'cast-spells self
                     'CAST-SPELLS)))  ;add callbacks, each student cast-spells at each
clock tick
       'CAST-SPELLS
      (lambda ()
         (if (not (ask self 'HAS-A 'WAND))
            (ask self 'EMIT (list "doesn't have a wand"))
           (let ((people-around (ask self 'PEOPLE-AROUND)))
```

```
            (if (null? people-around)
               (if (null? (ask self 'STUFF-AROUND))
                 (ask self 'EMIT (list "the spell is applied, but there's no one in the
room."))
      (ask self 'ZAP (list-ref (ask self 'STUFF-AROUND) (random (length (ask self
'STUFF-AROUND))))))  ;if there's no one around, wave the wand at a random target
      (ask self 'ZAP (list-ref people-around (random (length people-around))))))))  ;if
there's at least one person in the room, apply the spell to a random person

     'DIE
     (lambda (perp)
    (ask our-clock 'REMOVE-CALLBACK self 'cast-spells)
    (ask self 'SAY '("SHREEEEK!  I, uh, suddenly feel very faint..."))
    (ask autonomous-person-part 'DIE perp))
;remove the callback after the person dies

     'ZAP (lambda (target)
     (let ((spells (filter (lambda (x) (not (eq? (ask x 'IS-A 'SPELL) #f)))
                    (ask self 'THINGS))))
   ;find out all the spells the caster possesses
         (if (null? spells) (ask self 'EMIT (list (ask self 'NAME) " is waving the wand,
but nothing happens, because no spell is available to use"))   ;if there's no spell possessed
by the caster
      (let ((spell (list-ref spells (random (length spells)))))
            (ask self 'EMIT (list (ask self 'NAME) "is waving the wand and saying " (ask
spell 'INCANT)))
       (ask spell 'USE self target))))))  ;applying the spell to the target
   autonomous-person-part)))
;
;;test for wit-student class
;> (ask me 'LOOK-AROUND)
;
;You are in great-court
;You are not holding anything.
;You see stuff in the room: ZEPHYR-FROM-THE-DARK OBFUSCATION-CLOAK-4 flag-pole lovely-trees
;You see other people: mr-bigglesworth
;The exits are in directions: up west north OK
;> (ask our-clock 'TICK)
;
;--- THE-CLOCK Tick 0 ---
;ben-bitdiddle moves from stata-center to 34-301
;At 34-301 ben-bitdiddle says -- Hi registrar
;At 34-301 ben-bitdiddle is waving the wand and saying  dark white dark wooooh      ;waving
the wand
```

```
;At 34-301 registrar says -- Ouch! 0 hits is more than I want!
;alyssa-hacker moves from building-13 to edgerton-hall
;At edgerton-hall alyssa-hacker says -- Hi dr-evil
;At edgerton-hall alyssa-hacker says -- I take SLUG-SPELL from edgerton-hall
;At edgerton-hall alyssa-hacker is waving the wand and saying  dagnabbit ekaterin    ;waving
the wand
;At edgerton-hall A slug comes out of dr-evil 's mouth.
;person1 moves from 6001-lab to eecs-hq
;At eecs-hq person1 says -- Hi person2
;At eecs-hq person1 says -- I take OBFUSCATION-CLOAK-2 from eecs-hq
;At eecs-hq person1 is waving the wand and saying  dark white dark wooooh
            ;waving the wand
;At eecs-hq person2 says -- Ouch! 2 hits is more than I want!
;person2 moves from eecs-hq to 34-301
;At 34-301 person2 says -- Hi ben-bitdiddle registrar
;person2 moves from 34-301 to eecs-hq
;person2 moves from eecs-hq to eecs-ug-office
;At eecs-ug-office person2 says -- I take ZEPHYR-FROM-THE-DARK from eecs-ug-office
;At eecs-ug-office the spell is applied, but there's no one in the room.
;course-6-frosh moves from graduation-stage to great-court
;At great-court course-6-frosh says -- Hi tina mr-bigglesworth
;At great-court course-6-frosh says -- I take ZEPHYR-FROM-THE-DARK from great-court
;At  great-court  course-6-frosh  is  waving  the  wand  and  saying    dagnabbit
ekaterin         ;waving the wand
;At great-court A slug comes out of mr-bigglesworth 's mouth.
;lambda-man moves from stata-center to stata-center
;lambda-man moves from stata-center to stata-center
;lambda-man moves from stata-center to stata-center
;At stata-center lambda-man says -- I take sicp from stata-center
;At stata-center lambda-man is waving the wand and saying  dagnabbit ekaterin    waving the
wand
;dr-evil moves from edgerton-hall to 34-301
;At 34-301 dr-evil says -- Hi ben-bitdiddle registrar
;At 34-301 dr-evil says -- I'll let you off this once...
;mr-bigglesworth moves from great-court to legal-seafood
;At legal-seafood mr-bigglesworth says -- I'll let you off this once...
;grendel moves from lobby-7 to student-center
;At student-center grendel says -- I take BOIL-SPELL from student-center
;At student-center grendel 's belly rumbles
;registrar moves from 34-301 to eecs-hq
;At eecs-hq registrar 's belly rumbles
;>
```

```
(define (create-wit-professor name birthplace activity miserly)
```

```
 (create-instance wit-professor name birthplace activity miserly))

(define (wit-professor self name birthplace activity miserly)
 (let ((wit-part (wit-student self name birthplace activity miserly)))
   (make-handler
    'WIT-PROFESSOR
    (make-methods

    'INSTALL
     (lambda ()
    (ask wit-part 'INSTALL)
       (ask our-clock 'ADD-CALLBACK
           (create-clock-callback 'teach self 'TEACH)))
     'TEACH
     (lambda ()
       (let ((students (filter (lambda (person) (not (ask person 'IS-A 'WIT-PROFESSOR)))
(ask self 'PEOPLE-AROUND))))  ;find out all the non-teachers around
         (if (null? students) (ask self 'EMIT (list (ask self 'NAME) "doesn't have any student
to teach. Too bad!"))    ;if there's no student around
    (let* ((spells (ask chamber-of-stata 'THINGS))
     (person (list-ref students (random (length students))))   ;pick a random person
       (spell (list-ref spells (random (length spells)))))  ;pick a random spell
          (clone-spell spell person)
          (ask self 'EMIT (list (ask self 'NAME) "says to" (ask person 'NAME) "make sure
*you review" (ask spell 'NAME) "I taught you today at home!*"))))))

     'DIE
     (lambda (perp)
    (ask our-clock 'REMOVE-CALLBACK self 'teach)
    (ask self 'SAY '("SHREEEEK!  I, uh, suddenly feel very faint..."))
    (ask wit-student-part 'DIE perp)))
    wit-part)))

;;test result:
;ready
;> (ask our-clock 'TICK)
;
;--- THE-CLOCK Tick 0 ---
;ben-bitdiddle moves from baker to bexley
;At bexley ben-bitdiddle says -- Hi person1
;ben-bitdiddle moves from bexley to baker
;ben-bitdiddle moves from baker to bexley
;At bexley ben-bitdiddle says -- Hi person1
;At bexley ben-bitdiddle is waving the wand and saying  dark white dark wooooh
```

```
;At bexley person1 says -- Ouch! 0 hits is more than I want!
;alyssa-hacker moves from barker-library to 10-250
;At 10-250 alyssa-hacker says -- Hi registrar grendel lambda-man
;At 10-250 alyssa-hacker says -- I take ZEPHYR-FROM-THE-DARK from lambda-man
;At 10-250 lambda-man says -- I lose ZEPHYR-FROM-THE-DARK
;At 10-250 lambda-man says -- Yaaaah! I am upset!
;At 10-250 alyssa-hacker is waving the wand and saying  dark white dark wooooh
;At 10-250 registrar says -- Ouch! 0 hits is more than I want!
;At 10-250 grendel says -- Ouch! 1 hits is more than I want!
;At 10-250 lambda-man says -- Ouch! 1 hits is more than I want!
;person1 moves from bexley to baker
;person1 moves from baker to bexley
;At bexley person1 says -- Hi ben-bitdiddle
;person1 moves from bexley to student-center
;At student-center person1 says -- I take ZEPHYR-FROM-THE-DARK from student-center
;At student-center the spell is applied, but there's no one in the room.
;person2 moves from eecs-ug-office to eecs-hq
;person2 moves from eecs-hq to eecs-ug-office
;person2 moves from eecs-ug-office to eecs-hq
;At eecs-hq person2 is waving the wand and saying  dark white dark wooooh
;course-6-frosh moves from building-13 to lobby-10
;At lobby-10 course-6-frosh says -- Hi tina
;At lobby-10 course-6-frosh says -- I take SLUG-SPELL from lobby-10
;At lobby-10 course-6-frosh is waving the wand and saying  dagnabbit ekaterin
;At lobby-10 A slug comes out of tina 's mouth.
;lambda-man moves from 10-250 to barker-library
;At barker-library lambda-man says -- Hi dr-evil
;lambda-man moves from barker-library to 10-250
;At 10-250 lambda-man says -- Hi alyssa-hacker registrar grendel
;At 10-250 lambda-man says -- I take ZEPHYR-FROM-THE-DARK from alyssa-hacker
;At 10-250 alyssa-hacker says -- I lose ZEPHYR-FROM-THE-DARK
;At 10-250 alyssa-hacker says -- Yaaaah! I am upset!
;At 10-250 lambda-man is waving the wand and saying  dark white dark wooooh
;At 10-250 alyssa-hacker says -- Ouch! 1 hits is more than I want!
;At 10-250 registrar says -- Ouch! 1 hits is more than I want!
;At 10-250 grendel says -- Ouch! 1 hits is more than I want!
;susan-hockfield moves from legal-seafood to great-court
;susan-hockfield moves from great-court to legal-seafood
;At legal-seafood susan-hockfield is waving the wand and saying  dark white dark wooooh
;At  legal-seafood  susan-hockfield  doesn't  have  any  student  to  teach.  Too
bad!                ;Susan doesn't have anyone to teach
;eric-grimson moves from building-13 to lobby-10
;At lobby-10 eric-grimson says -- Hi course-6-frosh tina
;At lobby-10 eric-grimson says -- I take WAND from course-6-frosh
```

```
;At lobby-10 course-6-frosh says -- I lose WAND
;At lobby-10 course-6-frosh says -- Yaaaah! I am upset!
;At lobby-10 eric-grimson is waving the wand and saying  dark white dark wooooh
;At lobby-10 course-6-frosh says -- Ouch! 1 hits is more than I want!
;At lobby-10 tina says -- Ouch! 0 hits is more than I want!
;At lobby-10 eric-grimson says to tina make sure *you review  ZEPHYR-FROM-THE-DARK  I taught
you today at home!*    ;Eric taught Tina zephyr-from-the-dark.
;dr-evil moves from barker-library to 10-250
;At 10-250 dr-evil says -- Hi lambda-man alyssa-hacker registrar grendel
;dr-evil moves from 10-250 to barker-library
;At barker-library dr-evil says -- What are you doing still up? Everyone back to their rooms!
;mr-bigglesworth moves from 34-301 to stata-center
;At stata-center mr-bigglesworth says -- What are you doing still up? Everyone back to their
rooms!
;grendel moves from 10-250 to barker-library
;At barker-library grendel says -- Hi dr-evil
;registrar moves from 10-250 to barker-library
;At barker-library registrar says -- Hi grendel dr-evil
;registrar moves from barker-library to 10-250
;At 10-250 registrar says -- Hi lambda-man alyssa-hacker
;registrar moves from 10-250 to barker-library
;At barker-library registrar says -- Hi grendel dr-evil
;>

;Counter-spell class:
(define (create-counterspell name location)
  (create-instance counterspell name location))

(define (counterspell self name location)
  (let ((mobile-part (mobile-thing self name location)))
    (make-handler
     'COUNTERSPELL
     (make-methods)     ;the spell and the corresponding counterspell have the same name
      mobile-part)))
;I modified the USE method in the spell class, so that the spell can't act if the target
has ;the corresponding counter-spell
      'USE
      (lambda (caster target)
        (if (ask target 'IS-A 'CONTAINER)
        (if (null? (filter (lambda (x) (and (eq? (ask x 'NAME) (ask self 'NAME)) (ask x 'IS-A
'COUNTERSPELL))) (ask target 'THINGS)))
;see if the target has the counter-spell (type) and has the same name as the spell
      (action  caster  target)  (display  "but  nothing  happens"))  (action  caster
target)))          ;if the target carries the corresponding counter-spell, nothing happens
```

when the spell is applied

```
; ready
;> (ask me 'LOOK-AROUND)

;You are in bexley
;You are not holding anything.
;You see stuff in the room: ZEPHYR-FROM-THE-DARK
;You see other people: ben-bitdiddle
;The exits are in directions: west north OK
;> (ask me 'TAKE (thing-named 'ZEPHYR-FROM-THE-DARK ))
;
;There is more than one thing named ZEPHYR-FROM-THE-DARK here. Picking
;one of them.
;At bexley tina says -- I take ZEPHYR-FROM-THE-DARK from bexley
;(instance #<procedure:handler>)
;> (ask me 'TAKE (thing-named 'WAND))
;
;At bexley tina says -- I take WAND from ben-bitdiddle
;At bexley ben-bitdiddle says -- I lose WAND
;At bexley ben-bitdiddle says -- Yaaaah! I am upset! (instance
;#<procedure:handler>)
;> (create-counterspell 'ZEPHYR-FROM-THE-DARK (car (ask me
;'PEOPLE-AROUND)))
;(instance #<procedure:handler>)
;> (ask (thing-named 'WAND) 'ZAP (car (ask me 'PEOPLE-AROUND)))
;
;At bexley tina is waving the wand and saying  dark white dark wooooh but
;nothing happens  ;nothing happens because ben-bitdiddle has the counterspell
;>

 (define (create-chosen-one name birthplace activity miserly)
   (create-instance chosen-one name birthplace activity miserly))

(define (chosen-one self name birthplace activity miserly)
  (let ((autonomous-person-part (autonomous-person self name birthplace activity miserly))
        (health 3)
        (strength 1))
    (make-handler
     'CHOSEN-ONE
     (make-methods
      'SUFFER
      (lambda (hits perp)
     (ask self 'SAY (list "Ouch!" hits "hits is more than I want!"))
```

```
      (let ((old-health health))
      (set! health (- health hits))
      (if (<= health 0) (begin (display "I'm the chosen one! I can't die! You die!") (ask perp 'DIE self) (set! health old-health))))))
          ;the perp needs to die      restore the original health
    autonomous-person-part)))

;The SUFFER method actually calls the DIE method to ask the person to die. The reason why we are overriding the SUFFER method instead of the DIE method is that we are making sure that there's a way for the chosen one to die. We can still make the chosen one die, even though not calling from the SUFFER method. If we override the DIE method, the chosen-one really has no way to die, and that's a bit weird.

;;test output:
;ready
;> (ask our-clock 'TICK)
;
;--- THE-CLOCK Tick 0 ---
;ben-bitdiddle moves from bexley to baker
;At baker ben-bitdiddle says -- I take tons-of-code from baker
;At baker ben-bitdiddle is waving the wand and saying  dagnabbit ekaterin
;alyssa-hacker moves from eecs-hq to eecs-ug-office
;At eecs-ug-office alyssa-hacker says -- Hi eric-grimson
;alyssa-hacker moves from eecs-ug-office to eecs-hq
;At eecs-hq alyssa-hacker says -- I take OBFUSCATION-CLOAK-2 from eecs-hq
;At eecs-hq alyssa-hacker is waving the wand and saying  dagnabbit ekaterin
;person1 moves from legal-seafood to great-court
;At great-court person1 says -- I take ZEPHYR-FROM-THE-DARK from great-court
;At great-court person1 is waving the wand and saying  dark white dark wooooh
;person2 moves from legal-seafood to great-court
;At great-court person2 says -- Hi person1
;At great-court person2 says -- I take OBFUSCATION-CLOAK-4 from great-court
;At great-court person2 is waving the wand and saying  dark white dark wooooh
;At great-court person1 says -- Ouch! 0 hits is more than I want!
;course-6-frosh moves from grendels-den to lobby-10
;At lobby-10 course-6-frosh says -- Hi tina
;course-6-frosh moves from lobby-10 to lobby-7
;At lobby-7 course-6-frosh says -- I take ZEPHYR-FROM-THE-DARK from lobby-7
;At lobby-7 the spell is applied, but there's no one in the room.
;lambda-man moves from 34-301 to stata-center
;At stata-center lambda-man is waving the wand and saying  dagnabbit ekaterin
;susan-hockfield moves from student-center to bexley
;At bexley susan-hockfield is waving the wand and saying  dagnabbit ekaterin
;At bexley susan-hockfield doesn't have any student to teach. Too bad!
```

```
;eric-grimson moves from eecs-ug-office to eecs-hq
;eric-grimson moves from eecs-hq to eecs-ug-office
;eric-grimson moves from eecs-ug-office to eecs-hq
;At eecs-hq eric-grimson says -- I take SLUG-SPELL from eecs-hq
;At eecs-hq the spell is applied, but there's no one in the room.
;At eecs-hq eric-grimson doesn't have any student to teach. Too bad!
;dr-evil moves from grendels-den to lobby-10
;At lobby-10 dr-evil says -- Hi tina
;At lobby-10 dr-evil says -- What are you doing still up? Everyone back to their rooms!
;At lobby-10 tina goes home to lobby-10
;mr-bigglesworth moves from 34-301 to eecs-hq
;At eecs-hq mr-bigglesworth says -- Hi eric-grimson
;mr-bigglesworth moves from eecs-hq to eecs-ug-office
;At eecs-ug-office mr-bigglesworth says -- I'll let you off this once...
;grendel moves from barker-library to 10-250
;registrar moves from 34-301 to edgerton-hall
;registrar moves from edgerton-hall to legal-seafood
;At legal-seafood registrar says -- Hi hairy-cdr
;At legal-seafood registrar says -- I take OBFUSCATION-CLOAK-8 from legal-seafood
;At legal-seafood registrar takes a bite out of hairy-cdr
;At legal-seafood hairy-cdr says -- Ouch! 3 hits is more than I want! I'm the chosen one!
I can't die! You die!   ;See, hairy-cdr cannot die
;At legal-seafood registrar says -- SHREEEEK!  I, uh, suddenly feel very faint...
;At legal-seafood registrar says -- I lose OBFUSCATION-CLOAK-8
;At legal-seafood registrar says -- Yaaaah! I am upset!
;An earth-shattering, soul-piercing scream is heard...
;hairy-cdr moves from legal-seafood to edgerton-hall
;At edgerton-hall hairy-cdr says -- I take OBFUSCATION-CLOAK-9 from edgerton-hall
;>

;I did two of the five choices that are given in the project handout. I
;implemented the Moving Exits and Brooms.
;For moving exits, I first changed the place class. I added another internal variable
coordinate, which is a list consists of 3 numbers: x, y, z
(define (place self name coordinate)
  (let ((named-part (named-object self name))
     (container-part (container self))
     (exits '()))
    (make-handler
     'PLACE
     (make-methods
      'EXITS (lambda () exits)
      'COORDINATE (lambda () coordinate)
      'EXIT-TOWARDS
```

```
     (lambda (direction)
    (find-exit-in-direction exits direction))
     'ADD-EXIT
     (lambda (exit)
    (let ((direction (ask exit 'DIRECTION)))
      (if (not (ask self 'EXIT-TOWARDS direction))
          (set! exits (cons exit exits)))
      'done))
     )
    container-part named-part)))

;Then I added an internal procedure. given two places that are connected by stairs, return
the direction
(define (exit-direction from to)    ;coordinate is arranged in lists
  (let ((x1 (car (ask from 'COORDINATE)))
        (y1 (cadr (ask from 'COORDINATE)))
        (z1 (caddr (ask from 'COORDINATE)))
        (x2 (car (ask to 'COORDINATE)))
        (y2 (cadr (ask to 'COORDINATE)))
        (z2 (caddr (ask to 'COORDINATE))))
    (cond ((< z1 z2) 'up)          ;simple implementation for directions.
          ((> z1 z2) 'down)   ;I'm not dealing with angles, and assume every building is
perpendicular or parallel to each other. This is a simplified version
          ((< y1 y2) 'north)
          ((> y1 y2) 'south)
          ((< x1 x2) 'west)
          ((> x1 x2) 'east))))

;then I added another class: moving-exit
(define (create-moving-exit from direction to)
  (create-instance moving-exit from direction to))

(define (moving-exit self from direction to)
  (let ((root-part (root-object self)))  ;it inherit from the root object
    (make-handler
     'MOVING-EXIT
     (make-methods
      'INSTALL
      (lambda ()
     (if (not (null? (ask self 'FROM)))
         (ask (ask self 'FROM) 'ADD-EXIT self))
        (ask our-clock 'ADD-CALLBACK
             (create-clock-callback 'update self
                                    'UPDATE))  ;call update at each clock cycle
```

```
      'installed)
     'FROM       (lambda () from)
     'TO         (lambda () to)
     'DIRECTION   (lambda () direction)
     'USE      ;these methods are similar to those in the exit class
     (lambda (whom)
    (ask whom 'LEAVE-ROOM)
    (ask screen 'TELL-ROOM (ask whom 'LOCATION)
        (list (ask whom 'NAME)
          "moves from"
          (ask (ask whom 'LOCATION) 'NAME)
          "to"
          (ask to 'NAME)))
    (ask whom 'CHANGE-LOCATION to)
    (ask whom 'ENTER-ROOM))
     'UPDATE
      (lambda ()
     (if (= (remainder (current-time) 2) 0)   ;update every 2 clock cycle
       (let ((new-to (list-ref all-rooms (random (length all-rooms)))))
;all the stairs are pivoted at the from place, and then connect to one
;room from all the rooms at random
         (set! to new-to)
         (set! direction (exit-direction (ask self 'FROM) new-to)) ;update to and direction,
because these are the internal variables that got changed
         (display " exit from ")
          (display (ask (ask self 'FROM) 'NAME))
          (display " to ")
          (display (ask (ask self 'TO) 'NAME))
          (display " direction ")
          (display (ask self 'DIRECTION))
          (newline))))    ;output each exit so that we can see the change
     'DESTROY
     (lambda ()
    (ask our-clock 'REMOVE-CALLBACK self 'update)
    (ask exit-part 'DESTROY)))  ;remove the clock-callback if the moving exit is destroyed
    root-part)))

;associate each place with a 3-D coordinate:
(define (create-world)
  (let ((10-250 (create-place '10-250 (list 0 0 1)))
       (lobby-10 (create-place 'lobby-10 (list 0 0 0)))
       (grendels-den (create-place 'grendels-den (list 0 0 -1)))
       (barker-library (create-place 'barker-library (list 0 0 2)))
       (lobby-7 (create-place 'lobby-7 (list -1 0 0)))
```

```
       (eecs-hq (create-place 'eecs-hq (list 0 2 2)))
       (eecs-ug-office (create-place 'eecs-ug-office (list -1 2 2)))
       (edgerton-hall (create-place 'edgerton-hall (list 0 2 0)))
       (34-301 (create-place '34-301 (list 0 2 1)))
       (stata-center (create-place 'stata-center (list 1 2 1)))
       (6001-lab (create-place '6001-lab (list 0 2 3)))
       (building-13 (create-place 'building-13 (list 0 1 0)))
       (great-court (create-place 'great-court (list 0 -1 0)))
       (student-center (create-place 'student-center (list -2 0 0)))
       (bexley (create-place 'bexley (list -2 -1 0)))
       (baker (create-place 'baker (list -3 -1 0)))
       (legal-seafood (create-place 'legal-seafood (list 0 3 0)))
       (graduation-stage (create-place 'graduation-stage (list 0 -1 1)))
       (heaven (create-place 'heaven (list 1000 1000 1000))))

;when instantiating spells, we have to add the coordinate to the chamber-of-stata as well
because it's a place
(chamber (create-place 'chamber-of-stata (list 8 8 8)))

;Change connect-both-ways so that it uses moving-exits rather than exits to construct exits
(define (can-go-both-ways from direction reverse-direction to)
  (create-moving-exit from direction to)
  (create-moving-exit to reverse-direction from))

;testcase output:
> (ask our-clock 'TICK)

--- THE-CLOCK Tick 0 ---  exit from lobby-10 to 6001-lab direction up
 exit from 10-250 to eecs-ug-office direction up   ;the exits changed
 exit from grendels-den to eecs-hq direction up
 exit from lobby-10 to bexley direction south
 exit from 10-250 to stata-center direction north
 exit from barker-library to graduation-stage direction down
 exit from lobby-10 to eecs-ug-office direction up
 exit from lobby-7 to 6001-lab direction up
 exit from lobby-7 to bexley direction south
 exit from student-center to legal-seafood direction north
 exit from student-center to edgerton-hall direction north
 exit from bexley to grendels-den direction down
 exit from bexley to student-center direction north
 exit from baker to edgerton-hall direction north
 exit from lobby-10 to graduation-stage direction up
 exit from building-13 to 10-250 direction up
 exit from lobby-10 to grendels-den direction down
```

```
 exit from great-court to legal-seafood direction north
 exit from building-13 to 6001-lab direction up
 exit from edgerton-hall to 6001-lab direction up
 exit from edgerton-hall to eecs-ug-office direction up
 exit from 34-301 to stata-center direction west
 exit from 34-301 to great-court direction down
 exit from eecs-hq to 34-301 direction down
 exit from 34-301 to 6001-lab direction up
 exit from stata-center to edgerton-hall direction down
 exit from stata-center to barker-library direction up
 exit from stata-center to building-13 direction down
 exit from stata-center to graduation-stage direction south
 exit from stata-center to legal-seafood direction down
 exit from eecs-hq to great-court direction down
 exit from eecs-ug-office to legal-seafood direction down
 exit from edgerton-hall to lobby-10 direction south
 exit from legal-seafood to stata-center direction up
 exit from eecs-hq to bexley direction down
 exit from 6001-lab to lobby-10 direction down
 exit from legal-seafood to eecs-hq direction up
 exit from great-court to 10-250 direction up
 exit from great-court to grendels-den direction down
 exit from graduation-stage to grendels-den direction down

ben-bitdiddle moves from 10-250 to eecs-ug-office
At eecs-ug-office ben-bitdiddle says -- Hi course-6-frosh
At eecs-ug-office ben-bitdiddle says -- I take SLUG-SPELL from eecs-ug-office
alyssa-hacker moves from legal-seafood to eecs-hq
alyssa-hacker moves from eecs-hq to bexley
At bexley alyssa-hacker says -- I take BOIL-SPELL from bexley
course-6-frosh moves from eecs-ug-office to legal-seafood
lambda-man moves from baker to edgerton-hall
dr-evil moves from graduation-stage to grendels-den
dr-evil moves from grendels-den to eecs-hq
At eecs-hq dr-evil says -- What are you doing still up? Everyone back to their rooms!
mr-bigglesworth moves from building-13 to 10-250
At 10-250 mr-bigglesworth says -- What are you doing still up? Everyone back to their rooms!
grendel moves from stata-center to graduation-stage
At graduation-stage grendel says -- I take SLUG-SPELL from graduation-stage
At graduation-stage grendel 's belly rumbles
registrar moves from student-center to legal-seafood
At legal-seafood registrar says -- Hi course-6-frosh
registrar moves from legal-seafood to stata-center
At stata-center registrar 's belly rumbles
```

```
> (ask our-clock 'TICK)

--- THE-CLOCK Tick 1 ---    ;the moving exits don’t move in the odd clock cycles
ben-bitdiddle moves from eecs-ug-office to legal-seafood
At legal-seafood ben-bitdiddle says -- Hi course-6-frosh
ben-bitdiddle moves from legal-seafood to stata-center
At stata-center ben-bitdiddle says -- Hi registrar
ben-bitdiddle moves from stata-center to edgerton-hall
At edgerton-hall ben-bitdiddle says -- Hi lambda-man
At edgerton-hall ben-bitdiddle says -- I take BOIL-SPELL from edgerton-hall
alyssa-hacker moves from bexley to grendels-den
alyssa-hacker moves from grendels-den to eecs-hq
At eecs-hq alyssa-hacker says -- Hi dr-evil
At eecs-hq alyssa-hacker says -- I take SLUG-SPELL from eecs-hq
course-6-frosh moves from legal-seafood to stata-center
At stata-center course-6-frosh says -- Hi registrar
course-6-frosh moves from stata-center to edgerton-hall
At edgerton-hall course-6-frosh says -- Hi ben-bitdiddle lambda-man
course-6-frosh moves from edgerton-hall to 6001-lab
lambda-man moves from edgerton-hall to eecs-ug-office
dr-evil moves from eecs-hq to great-court
dr-evil moves from great-court to legal-seafood
At legal-seafood dr-evil says -- I'll let you off this once...
mr-bigglesworth moves from 10-250 to stata-center
At stata-center mr-bigglesworth says -- Hi registrar
At stata-center mr-bigglesworth says -- What are you doing still up? Everyone back to their
rooms!
At stata-center registrar goes home to student-center
grendel moves from graduation-stage to grendels-den
At grendels-den grendel 's belly rumbles
registrar moves from student-center to legal-seafood
At legal-seafood registrar says -- Hi dr-evil
registrar moves from legal-seafood to stata-center
At stata-center registrar says -- Hi mr-bigglesworth
> (ask our-clock 'TICK)

--- THE-CLOCK Tick 2 ---  exit from lobby-10 to legal-seafood direction north      ;in an
even clock cycle, the moving exits all move again
 exit from 10-250 to stata-center direction north
 exit from grendels-den to student-center direction up
 exit from lobby-10 to lobby-10 direction #<void>
 exit from 10-250 to 10-250 direction #<void>
 exit from barker-library to 34-301 direction down
 exit from lobby-10 to legal-seafood direction north
```

```
 exit from lobby-7 to lobby-10 direction west
 exit from lobby-7 to lobby-7 direction #<void>
 exit from student-center to grendels-den direction down
 exit from student-center to bexley direction south
 exit from bexley to graduation-stage direction up
 exit from bexley to 6001-lab direction up
 exit from baker to baker direction #<void>
 exit from lobby-10 to building-13 direction north
 exit from building-13 to bexley direction south
 exit from lobby-10 to 34-301 direction up
 exit from great-court to legal-seafood direction north
 exit from building-13 to 10-250 direction up
 exit from edgerton-hall to lobby-7 direction south
 exit from edgerton-hall to edgerton-hall direction #<void>
 exit from 34-301 to graduation-stage direction south
 exit from 34-301 to student-center direction down
 exit from eecs-hq to baker direction down
 exit from 34-301 to graduation-stage direction south
 exit from stata-center to 10-250 direction south
 exit from stata-center to great-court direction down
 exit from stata-center to stata-center direction #<void>
 exit from stata-center to 10-250 direction south
 exit from stata-center to lobby-7 direction down
 exit from eecs-hq to graduation-stage direction down
 exit from eecs-ug-office to barker-library direction south
 exit from edgerton-hall to graduation-stage direction up
 exit from legal-seafood to baker direction south
 exit from eecs-hq to building-13 direction down
 exit from 6001-lab to bexley direction down
 exit from legal-seafood to grendels-den direction down
 exit from great-court to stata-center direction up
 exit from great-court to lobby-10 direction north
 exit from graduation-stage to barker-library direction up

ben-bitdiddle moves from edgerton-hall to edgerton-hall
ben-bitdiddle moves from edgerton-hall to lobby-7
At lobby-7 ben-bitdiddle says -- I take SLUG-SPELL from lobby-7
alyssa-hacker moves from eecs-hq to building-13
At building-13 alyssa-hacker says -- Hi ben-bitdiddle
At building-13 alyssa-hacker says -- I take SLUG-SPELL from building-13
course-6-frosh moves from 6001-lab to bexley
course-6-frosh moves from bexley to graduation-stage
At graduation-stage course-6-frosh says -- I take diploma from graduation-stage
lambda-man moves from eecs-ug-office to barker-library
```

```
lambda-man moves from barker-library to 34-301
lambda-man moves from 34-301 to graduation-stage
At graduation-stage lambda-man says -- Hi course-6-frosh
dr-evil moves from legal-seafood to baker
dr-evil moves from baker to baker
At baker dr-evil says -- I'll let you off this once...
mr-bigglesworth moves from stata-center to great-court
At great-court mr-bigglesworth says -- What are you doing still up? Everyone back to their
rooms!
grendel moves from grendels-den to student-center
At student-center grendel 's belly rumbles
registrar moves from stata-center to lobby-7
At lobby-7 registrar says -- Hi ben-bitdiddle
registrar moves from lobby-7 to lobby-7
At lobby-7 registrar says -- Hi ben-bitdiddle
At lobby-7 registrar takes a bite out of ben-bitdiddle
At lobby-7 ben-bitdiddle says -- Ouch! 2 hits is more than I want!
>
```

```
;I also implemented Brooms. I created a Broom class
;create a new broom class, which inherit from mobile-thing class
(define (create-broom name location)
  (create-instance broom name location))

(define (broom self name location)
  (let ((mobile-part (mobile-thing self name location)))
    (make-handler
     'BROOM
     (make-methods
      )
     mobile-part)))

;I added a FLY method to the person class
'FLY                ;implement another method, FLY
     (lambda (place)          ;the person can fly to wherever on campus
;if he has a broom
       (if (not (null? (ask self 'HAS-A 'BROOM)))
       (begin (ask self 'LEAVE-ROOM)   ;if I have a broom, I fly to the desired place directly
    (ask screen 'TELL-ROOM (ask self 'LOCATION)
        (list (ask self 'NAME)
           "flies from"
           (ask (ask self 'LOCATION) 'NAME)
           "to"
           (ask place 'NAME)))
```

```
    (ask self 'CHANGE-LOCATION place)
    (ask self 'ENTER-ROOM))
      (ask self 'EMIT (list (ask self 'NAME) "cannot fly because s/he doesn't have a
broom"))))   ;if I don't have a broom, complain

;;test output:
;     ready
> (ask me 'FLY (car all-rooms))

At lobby-7 tina cannot fly because s/he doesn't have a broom MESSAGE-DISPLAYED    ;if I
don't have a broom
>
;> (ask me 'LOOK-AROUND)
;
;You are in student-center
;You are not holding anything.
;You see stuff in the room: SLUG-SPELL
;There are no other people around you.
;The exits are in directions: south east OK
;> (ask (car all-rooms) 'NAME)
;10-250
;> (ask me 'FLY (car all-rooms))
;
;tina flies from student-center to 10-250 #t
;> (ask me 'LOOK-AROUND)
;
;You are in 10-250
;You are not holding anything.
;You see stuff in the room: SLUG-SPELL OBFUSCATION-CLOAK-1 recitation-problem problem-set
blackboard
;There are no other people around you.
;The exits are in directions: up down OK    ;I can fly!
;>
```