

## Solution 10: Agents and board games

ETH Zurich

### 1 Navigating in Zurich

Listing 1: Class *NAVIGATOR*

```
note
  description: "Finding routes in Zurich."

class
  NAVIGATOR

inherit
  ZURICH_OBJECTS

feature -- Explore Zurich

  add_event_handlers
    -- Add handlers to mouse-click events on stations
    -- to allow the user to select start and end points of his route.
  do
    across
      Zurich.stations as i
    loop
      Zurich_map.views [i.item].on_left_click_no_args.extend_back (agent set_origin (i.item))
      Zurich_map.views [i.item].on_left_click_no_args.extend_back (agent show_route)
      Zurich_map.views [i.item].on_right_click_no_args.extend_back (agent set_destination (i.item))
      Zurich_map.views [i.item].on_right_click_no_args.extend_back (agent show_route)
    end
  end

feature -- Access

  origin: STATION
    -- Currently selected start point.
    -- (Void if no start point selected).

  destination: STATION
    -- Currently selected end point.
    -- (Void if no end point selected).

  last_route: ROUTE
    -- Route calculated by the latest call to 'show_route'.
```

```
finder: ROUTE_FINDER
  -- Route finder.
  once
    create Result.make (Zurich)
  end

feature {NONE} -- Implementation

set_origin (s: STATION)
  -- Set 'origin' to 's'.
  do
    origin := s
  ensure
    origin_set: origin = s
  end

set_destination (s: STATION)
  -- Set 'destination' to 's'.
  do
    destination := s
  ensure
    destination_set: destination = s
  end

show_route
  -- If both 'origin' and 'destination' are set, show the route from 'origin' to 'destination
  ' on the map
  -- and output directions to the console.
  -- Otherwise do nothing.
  local
    i: INTEGER
  do
    if origin /= Void and destination /= Void then
      if last_route /= Void then
        Zurich.remove_route (last_route)
      end
      last_route := finder.shortest_route (origin, destination)
      Zurich.add_route (last_route)
      Zurich_map.update

      Console.output ("From " + origin.name + " to " + destination.name + ":")
      from
        i := 1
      until
        i > last_route.lines.count
      loop
        Console.append_line ("Take " + last_route.lines[i].kind.name + " " + last_route.
          lines[i].number.out +
          " until " + last_route.stations[i + 1].name)
        i := i + 1
      end
    end
  end
```

```
end

invariant
  finder_exists: finder /= Void
end
```

## 2 Home automation

Listing 2: Class *TEMPERATURE\_SENSOR*

```
class
  TEMPERATURE_SENSOR

inherit
  ANY
  redefine
    default_create
  end

feature {NONE} -- Initialization

  default_create
    -- Initialize the set of observers.
  do
    create {V_HASH_SET [PROCEDURE [ANY, TUPLE [REAL_64]]]} observers
  ensure then
    no_observers: observers.is_empty
  end

feature -- Access

  temperature: REAL_64
    -- Temperature value in degrees Celcius.

feature -- Status report

  valid_temperature (a_value: REAL_64): BOOLEAN
    -- Is 'a_value' a valid temperature?
  do
    Result := a_value >= -273.15
  end

feature -- Basic operations

  set_temperature (a_temperature: REAL_64)
    -- Set 'temperature' to 'a_temperature' and notify observers.
  require
    valid_temperature: valid_temperature (a_temperature)
  do
    temperature := a_temperature
  across
    observers as c
```

```
    loop
      c.item.call ([temperature])
    end
  ensure
    temperature_set: temperature = a_temperature
  end

feature -- Subscription

  subscribe (an_observer: PROCEDURE [ANY, TUPLE [REAL_64]])
    -- Add 'an_observer' to observers list.
  do
    observers.extend (an_observer)
  ensure
    present: observers.has (an_observer)
  end

  unsubscribe (an_observer: PROCEDURE [ANY, TUPLE [REAL_64]])
    -- Remove 'an_observer' from observers list.
  do
    observers.remove (an_observer)
  ensure
    absent: not observers.has (an_observer)
  end

feature {NONE} -- Implementation

  observers: V_SET [PROCEDURE [ANY, TUPLE [REAL_64]])
    -- Set of observing agents.

invariant
  valid_temperature: valid_temperature (temperature)
  observers_exists: observers /= Void
  all_observers_exist: not observers.has (Void)
end
```

Listing 3: Class *APPLICATION*

```
class
  APPLICATION

create
  make

feature {NONE} -- Initialization
  make
    -- Run application.
  local
    s: TEMPERATURE_SENSOR
    d: DISPLAY
    c: HEATING_CONTROLLER
  do
    create s
```

```
create d
create c.set_goal (21.5)

s.subscribe (agent d.show)
s.subscribe (agent c.adjust)

s.set_temperature (22)
s.set_temperature (22.8)
s.set_temperature (20.0)

s.set_temperature (-273.14276764)
s.set_temperature (1000)
s.set_temperature (0)
end
end
```

### 3 The final project. Board game: part 4

You can download a complete solution from [http://se.inf.ethz.ch/courses/2012b\\_fall/eprog/assignments/10/board\\_game\\_solution.zip](http://se.inf.ethz.ch/courses/2012b_fall/eprog/assignments/10/board_game_solution.zip).