Solution 4: Object creation and logic

ETH Zurich

1 Creating objects in Traffic

Listing 1: Class OBJECT_CREATION

<table>
<thead>
<tr>
<th>note</th>
<th>description: “Creating new objects for Zurich.”</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>OBJECT_CREATION</td>
</tr>
<tr>
<td>inherit</td>
<td>ZURICH_OBJECTS</td>
</tr>
<tr>
<td>feature</td>
<td>--- Explore Zurich</td>
</tr>
<tr>
<td></td>
<td>explore</td>
</tr>
<tr>
<td></td>
<td>--- Create new objects for Zurich.</td>
</tr>
<tr>
<td></td>
<td>do</td>
</tr>
<tr>
<td></td>
<td>add_buildings</td>
</tr>
<tr>
<td></td>
<td>add_route</td>
</tr>
<tr>
<td></td>
<td>--- check out add_route_2 as an alternative</td>
</tr>
<tr>
<td></td>
<td>end</td>
</tr>
<tr>
<td></td>
<td>add_buildings</td>
</tr>
<tr>
<td></td>
<td>--- Add ETH main building and Opera house to Zurich.</td>
</tr>
<tr>
<td></td>
<td>local</td>
</tr>
<tr>
<td></td>
<td>corner_a, corner_b: VECTOR</td>
</tr>
<tr>
<td></td>
<td>eth, opera: BUILDING</td>
</tr>
<tr>
<td></td>
<td>do</td>
</tr>
<tr>
<td></td>
<td>create corner_a.make (250, −20)</td>
</tr>
<tr>
<td></td>
<td>create corner_b.make (300, −100)</td>
</tr>
<tr>
<td></td>
<td>create eth.make (“Raemistrasse 101”, corner_a, corner_b)</td>
</tr>
<tr>
<td></td>
<td>eth.set_name (“ETH”)</td>
</tr>
<tr>
<td></td>
<td>Zurich.add_building (eth)</td>
</tr>
<tr>
<td></td>
<td>create corner_a.make (200, −1400)</td>
</tr>
<tr>
<td></td>
<td>create corner_b.make (260, −1480)</td>
</tr>
<tr>
<td></td>
<td>create opera.make (“Schillerstrasse 1”, corner_a, corner_b)</td>
</tr>
<tr>
<td></td>
<td>opera.set_name (“Opera”)</td>
</tr>
<tr>
<td></td>
<td>Zurich.add_building (opera)</td>
</tr>
<tr>
<td></td>
<td>end</td>
</tr>
<tr>
<td></td>
<td>add_route</td>
</tr>
<tr>
<td></td>
<td>--- Add a route from Polyterrasse to Opernhaus through Paradeplatz to Zurich.</td>
</tr>
</tbody>
</table>


local
  leg1, leg2, leg3: LEG
  opera_route: ROUTE
  do
    create leg1.make (Zurich.station ("Polyterrasse"), Zurich.station ("Central"),
          Zurich.line (24))
    create leg2.make (Zurich.station ("Central"), Zurich.station ("Paradeplatz"),
          Zurich.line (7))
    create leg3.make (Zurich.station ("Paradeplatz"), Zurich.station ("Opernhaus"),
          Zurich.line (2))
    leg1.link (leg2)
    leg2.link (leg3)
    create opera_route.make (leg1)
    Zurich.add_route (opera_route)
  end
end

2 Temperature application

Listing 2: Class TEMPERATURE

note
description: "Temperature."

class
  TEMPERATURE

create
  make_celsius, make_kelvin

feature -- Initialization

make_celsius (v: INTEGER)
  -- Create with Celsius value ‘v’.
  require
    above_absolute_zero: v >= – Celsius_zero
  do
    celsius := v
  ensure
    celsius_value_set: celsius = v
  end

make_kelvin (v: INTEGER)
  -- Create with Kelvin value ‘v’.
  require
    above_absolute_zero: v >= 0
  do
    celsius := v – Celsius_zero
  ensure
    kelvin_value_set: kelvin = v
  end
feature -- Access

celsius: INTEGER
   -- Value on Celsius scale.

kelvin: INTEGER
   -- Value on Kelvin scale.
   do
      Result := celsius + Celsius_zero
   end

Celsius_zero: INTEGER = 273
   -- The zero of the Celsius scale on Kelvin scale.

feature -- Measurement

average (other: TEMPERATURE): TEMPERATURE
   -- Average temperature between ‘Current’ and ‘other’.
   require
      other_exists: other /= Void
   do
      create Result.make_celsius ((celsius + other.celsius) // 2)
      ensure
         between: (celsius <= Result.celsius and Result.celsius <= other.celsius) or
                  (other.celsius <= Result.celsius and Result.celsius <= celsius)
   end

invariant
   above_absolute_zero: kelvin >= 0
end

Listing 3: Class APPLICATION

note
   description : "Temperature application root class"

class
   APPLICATION
create
   make

feature {NONE} -- Initialization

   make
      -- Run application.
      local
         t1, t2, t3: TEMPERATURE
      do
         Io.put_string ("Enter the first temperature in Celsius: ")
         Io.read_integer
         create t1.make_celsius (Io.last_integer)
```plaintext
Io.put_string ("The first temperature in Kelvin is: ")
Io.put_integer (t1.kelvin)
Io.new_line

Io.put_string ("Enter the second temperature in Kelvin: ")
Io.read_integer
create t2.make_kelvin (Io.last_integer)
Io.put_string ("The second temperature in Celsius is: ")
Io.put_integer (t2.celsius)
Io.new_line

$ t3 := t1.average (t2) $
Io.put_string ("The average in Celsius is: ")
Io.put_integer (t3.celsius)
Io.new_line
Io.put_string ("The average in Kelvin is: ")
Io.put_integer (t3.kelvin)
Io.new_line
end

3 Ein Ticket für alles

Listing 4: Class **APPLICATION**

**note**

description : "ZVV information system."

**class**

**APPLICATION**

**create**

**execute**

feature { **NONE** } -- Initialization

**execute**

-- Run application.
**do**
read_data
if not read_error then
Io.new_line
print ("Eligible for discount: ")
print (gets_discount)
end
end

feature -- Access

**birth_date**: **DATE**

-- Birth date.
```
home: STRING
   -- Home postal code.

work: STRING
   -- Work postal code.

age: INTEGER
   -- Age (difference in years between today’s date and ‘birth_date’).
   require
      birth_date_exists: birth_date /= Void
   local
today: DATE
do
   create today.make_now
   Result := today.relative_duration (birth_date).year
end

feature   -- Status report

is_valid_postal_code (pc: STRING): BOOLEAN
   -- Is ‘pc’ a valid postal code in Switzerland?
do
   Result := pc /= Void and then (pc.count = 4 and pc.is_natural)
end

is_in_zurich_canton (pc: STRING): BOOLEAN
   -- Is postal code ‘pc’ inside the canton of Zurich?
   require
      valid_code: is_valid_postal_code (pc)
do
   Result := pc [1] = '8'
end

is_in_zurich_city (pc: STRING): BOOLEAN
   -- Is postal code ‘pc’ inside the city of Zurich?
   require
      valid_code: is_valid_postal_code (pc)
do
end

gets_discount: BOOLEAN
   -- Is a customer with the current ‘birth_date’, ‘home’ and ‘work’ eligible for a
discounted seasonal ticket?
   require
      birth_date_exists: birth_date /= Void
      valid_home_code: is_valid_postal_code (home)
      valid_work_code: is_valid_postal_code (work)
do
   Result := age < 25 or (is_in_zurich_canton (home) and is_in_zurich_city (home) /=
                        is_in_zurich_city (work))
end

feature \{NONE\} -- Implementation

read_error: BOOLEAN
  -- Did an error occur while reading user data?

read_data
  -- Read user input.
  local
date_format: STRING
do
  do
    date_format := "[0]dd/[0]mm/yyyy"
    print ("Enter birth date as dd/mm/yyyy: ")
    Io.read_line
    if not (create \{DATE_VALIDITY_CHECKER\}).date_valid (Io.last_string, date_format)
      then
        print ("Invalid date")
        read_error := True
    else
      create birth_data.make_from_string (Io.last_string, date_format)
  end
if not read_error then
  print ("Enter home postal code: ")
  Io.read_line
  home := Io.last_string.twin
  if not is_valid_postal_code (home) then
    print ("Invalid postal code")
    read_error := True
  end
end
if not read_error then
  print ("Enter work postal code: ")
  Io.read_line
  work := Io.last_string.twin
  if not is_valid_postal_code (work) then
    print ("Invalid postal code")
    read_error := True
  end
end
end

4 MOOC: Object creation and logic

The order in which the questions and the answers appear here in the solution may vary because they are randomly shuffled at each attempt.

Object creation Quiz:

- In class POINT you have the following creation routines:
set_coordinates (x, y: INTEGER)
set_color (col: STRING)
default_create
Assuming that you have defined an entity of type POINT as
p: POINT,
match the instruction fragments on the left with those on the right to form correct creation
instructions.
create {POINT} p.set_color ("green"), create p.default_create, create {POINT}.set_color
("black"), create p.set_coordinates (5, 4)

• In class POINT you have the following creation routines:
  set_coordinates (x, y: INTEGER)
  set_color (col: STRING)
  Assuming that you have defined an entity of type POINT as
  p: POINT,
  which of the following are correct creation instructions?
  create p.set_coordinates (-4, 7), create p.set_color ("Red")

• In class POINT you have no creation routines, and the following routines:
  set_coordinates (x, y: INTEGER)
  set_color (col: STRING)
  Assuming that you have defined an entity of type POINT as
  p: POINT,
  which of the following are correct creation instructions?
  create p, create p.default_create, create {POINT} p, create {POINT}

• True or False? Suppose to have a reference p to class POINT with creation feature
  set_coordinates (x, y: INTEGER)
  Then the effect of the following two instructions, executed in the given order, is to have
  an object attached to p initialized in a way that its coordinates are both 7.
  create p.set_coordinates (7, 7)
  create p.set_coordinates (9, 9)
  Correct answer: false

• Suppose to have a class PERSON with an attribute current_job: JOB, where JOB is a
  class modeling a job. This should be an attribute for which Void is an acceptable value.
  Correct answer: true

• The first element of a class modeling a list data structure can never be Void.
  Correct answer: false

A bit of Logic quizzes
We don’t give solutions for this one.