

## Solution 4: Object creation and logic

ETH Zurich

### 1 Creating objects in Traffic

Listing 1: Class *OBJECT\_CREATION*

```
note
  description: "Creating new objects for Zurich."

class
  OBJECT_CREATION

inherit
  ZURICH_OBJECTS

feature -- Explore Zurich

  explore
    -- Create new objects for Zurich.
    do
      add_buildings
      add_route
    end

  add_buildings
    -- Add ETH main building and Opera house to Zurich.
    local
      corner_a, corner_b: VECTOR
      eth, opera: BUILDING
    do
      create corner_a.make (250, -20)
      create corner_b.make (300, -100)
      create eth.make ("Raemistrasse 101", corner_a, corner_b)
      eth.set_name ("ETH")
      Zurich.add_building (eth)
      create corner_a.make (200, -1400)
      create corner_b.make (260, -1480)
      create opera.make ("Schillerstrasse 1", corner_a, corner_b)
      opera.set_name ("Opera")
      Zurich.add_building (opera)
    end

  add_route
    -- Add a route from Polyterrasse to Opernhaus through Paradeplatz to Zurich.
    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
  execute

feature {NONE} -- Initialization

  execute
    -- 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)
    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
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

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

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

in_zurich_city (pc: STRING): BOOLEAN
  -- Is postal code 'pc' inside the city of Zurich?
  require
    valid_code: valid_postal_code (pc)
  do
    Result := pc [1] = '8' and pc [2] = '0'
  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: valid_postal_code (home)
    valid_work_code: valid_postal_code (work)
  do
    Result := age < 25 or (in_zurich_canton (home) and in_zurich_city (home) /=
      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
    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_date.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 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 valid_postal_code (work) then
        print ("Invalid postal code")
        read_error := True
      end
    end
  end
end
end
```