

Solution 7: Peachy loops

ETH Zurich

1 Buildings for Paris

Solution

Listing 1: Class *CONSTRUCTION*

```
1 indexing
   description: "Construction class (Assignment 7)"
3  date: "$Date$"
   revision: "$Revision$"
5
   class
7  CONSTRUCTION
9 inherit
11 TOURISM
13 feature -- Explore Paris
15 build is
   -- Build trams and connecting lines.
17 do
   Paris.display
19   wait
21   -- generate_buildings_along_segment (Paris.roads.item (15).one_way)
23   generate_buildings
25
   -- To be filled in
27 end
29 generate_buildings_along_segment (a_segment: TRAFFIC_ROAD_SEGMENT) is
   -- Generate building along a segment.
31 require
   a_segment_exists: a_segment /= Void
33 local
   pp: DS_ARRAYED_LIST[TRAFFIC_POINT]
   c1, c2, step, up, down: TRAFFIC_POINT
   count: INTEGER
37   i, j: INTEGER
```

```

39     building: TRAFFIC_VILLA
40 do
41     from
42         i := 1
43         pp := a_segment.polypoints
44     until
45         i >= pp.count
46     loop
47         c1 := pp.item (i)
48         c2 := pp.item (i+1)
49         step := c2 - c1
50         step.scale_to (24.0)
51         create down.make (c2.y - c1.y, c1.x - c2.x)
52         down.scale_to (24.0)
53         create up.make (c1.y - c2.y, c2.x - c1.x)
54         up.scale_to (24.0)
55     from
56         j := 1
57         count := (c1.distance (c2)/step.length).floor
58     until
59         j > count
60     loop
61         if not grid.has_rectangle_collision (c1 + step*j + down, 18.0, 16.0) then
62             create building.make_default (c1 + step*j + down)
63             Paris.put_building (building)
64         end
65         if not grid.has_rectangle_collision (c1 + step*j + up, 18.0, 16.0) then
66             create building.make_default (c1 + step*j + up)
67             Paris.put_building (building)
68         end
69         j := j + 1
70     end
71     i := i + 1
72 end
73
74 -- For task 1.5. Use the conditional below
75 -- if you only want to put buildings when
76 -- it doesn't collide with other buildings or lines:
77 --
78 -- if not grid.has_rectangle_collision (building_coordinate, 18.0, 16.0) then
79 --     -- Create and add building
80 -- end
81 end
82
83 generate_buildings is
84     -- Generate buildings along all roads.
85 local
86     r : TRAFFIC_ROAD_SEGMENT
87 do
88     from
89         Paris.roads.start
90     until
91         Paris.roads.after

```

```

    loop
91     r := Paris.roads.item_for_iteration.one_way
        generate_buildings_along_segment (r)
93     Paris.roads.forth
    end
95 end

fill_grid is
    -- Create and fill grid
99 local
    l: TRAFFIC_LINE_SEGMENT
101 do
    create grid.make (200, Paris.center, Paris.radius)
103 from
    Paris.line_segments.start
105 until
    Paris.line_segments.after
107 loop
    l := Paris.line_segments.item_for_iteration
109 grid.mark_polyline (l.polypoints, 7.0, True)
    Paris.line_segments.forth
111 end
    ensure
113     grid_exists : grid /= Void
    end
115
    grid: TRAFFIC_GRID
117
119 end
```

2 Bagels!

Solution

Listing 2: Class `APPLICATION_BAGELS`

```

1 indexing
    description : "bagels application root class"
3   date       : "$Date: 2008-12-29 15:41:59 -0800 (Mon, 29 Dec 2008) $"
    revision   : "$Revision: 76432 $"
5
class
7   BAGELS_APPLICATION

9 inherit
    ARGUMENTS
11
    create
13   make

15 feature -- Initialization
```

```

17  make
    -- Run application.
19  local
    dim: INTEGER
21  guess: STRING
    guess_number: INTEGER
23  do
    io.put_string ("*** Welcome to Bagels! ***%N")
25  from
    until
27    io.last_integer > 0
    loop
29    io.put_string ("Enter the number of digits (positive):%N")
    io.read_integer
31  end
    dim := io.last_integer
33  io.put_string ("I'm thinking of a number...")
    generate_searched (dim)
35  io.put_string (" Okay, got it!%N")

37  from
    until
39    guess /= Void and then searched.is_equal (guess)
    loop
41    io.put_string ("Enter your guess: ")
    io.read_integer
43    guess := io.last_integer.out
    if guess.count = dim and io.last_integer > 0 and not guess.has ('0') then
45      generate_clue (guess)
      guess_number := guess_number + 1
47      if clue.is_empty then
      print ("No FERMI and no PICA!%N")
49      else
      print (clue + "%N")
51      end
    else
53      io.put_string ("Incorrect input, please enter a positive number of length "
      + dim.out + " containing no zeros%N")
    end
55  end
    print ("Congratulations! You made it in " + guess_number.out + " guesses.")
57  end

59  generate_clue (guess: STRING)
    -- Generate the clue.
61  require
    searched_set: searched /= Void
63  guess_set: guess /= Void
    same_length: searched.count = guess.count
65  local
    i, j, k: INTEGER
67  mask: STRING
  
```

```

69   do
70     clue := ""
71     create mask.make (guess.count)
72     mask.fill_blank
73     -- First look for correctly positioned items.
74     from
75       i := 1
76     until
77       i > searched.count
78     loop
79       if guess.item (i) = searched.item (i) then
80         mask.put ('F', i)
81         clue := clue + "FERMI "
82       end
83       i := i + 1
84     end
85     -- Then look for items not correctly positioned
86     from
87       i := 1
88     until
89       i > searched.count
90     loop
91       if mask.item (i) /= 'F' then -- Current character of 'searched' is not yet FERMI
92         if guess.has (searched.item (i)) then
93           from
94             j := guess.occurrences (searched.item (i))
95             k := 0
96           until
97             j < 1
98           loop
99             k := guess.index_of (searched.item (i), k+1)
100            if mask.item (k) = ' ' then
101              mask.put ('P', k)
102              clue := clue + "PICA "
103              j := 0
104            end
105            j := j - 1
106          end
107        end
108        i := i + 1
109      end
110    ensure
111      clue_set: clue /= Void
112    end
113  generate_searched (dim: INTEGER)
114  -- Generate a new 'searched' of size 'dim'.
115  require
116    dim_positive: dim > 0
117  local
118    random: RANDOM

```

```
121   t: TIME
122   do
123     create t.make_now
124     create random.set_seed ((t.fine_seconds*1000).truncated_to_integer)
125     from
126       searched := ""
127     until
128       dim = searched.count
129     loop
130       random.forth
131       searched := searched + random.item_for_iteration.out
132       searched.prune_all ('0')
133       if searched.count > dim then
134         searched := searched.substring (1, dim)
135       end
136     end
137   ensure
138     searched_set: searched /= Void and then not searched.is_empty
139   end
140   clue: STRING
141     -- Last generated clue.
142
143   searched: STRING
144     -- Last searched string.
145 end
```

3 Decimal to Binary conversions

Solution

Listing 3: Class `DECIMAL_TO_BINARY_CONVERTER`

```
1 indexing
   description : "decimal_to_binary application root class"
3   date       : "$Date: 2008-12-29 15:41:59 -0800 (Mon, 29 Dec 2008) $"
   revision    : "$Revision: 76432 $"
5
6   class DECIMAL_TO_BINARY_CONVERTER
7
8   create
9   make
10
11  feature {NONE} -- Initialization
12
13  make
   -- Run application.
14
15  do
   create_input_file -- only for local testing: comment to test the program with the
   system
16  read_input_file
17  if not has_reading_failed then
18  print ("%NStarting conversion...")
   convert_from_decimal_to_binary ( last_read_integer )
20  print ( last_read_integer .out + " converted to binary is: " +
   last_converted_binary)
21  write_output_file
22
23  end
24  end
25
26  feature -- Basic operations
27
28  convert_from_decimal_to_binary ( dec: INTEGER_32)
   -- Convert an integer from base 10 to base 2.
29  require
30  non_negativity: dec >= 0
31  local
32  temp_dec: INTEGER_32
33
34  do
35  create last_converted_binary .make_empty
36  if dec = 0 then
37  last_converted_binary .prepend ("0")
38  else
39  from
40  from
41  temp_dec := dec
42  invariant
43  dec // (2 ^ last_converted_binary .count).truncated_to_integer = temp_dec
44  until
```

```

45     temp_dec = 0
      loop
47         if temp_dec \% 2 = 0 then
            last_converted_binary.prepend("0")
49         else
            last_converted_binary.prepend("1")
51         end
            temp_dec := temp_dec // 2
53         variant
            temp_dec + 1
55         end
      end
57 ensure
    last_converted_binary.set : last_converted_binary /= Void and then not
    last_converted_binary.is_empty
59 end

61 feature -- I/O

63 create_input_file
    -- Create output file (for testing only)
65 local
    input_file : PLAIN_TEXT_FILE
67 do
    create input_file.make_open_write (Input_file_name)
69     input_file.put_string ("11")
    input_file.close
71 end

73 read_input_file
    -- Read input file. Put result in last_read_integer .
75 local
    input_file : PLAIN_TEXT_FILE
77 do
    create input_file.make_open_read (Input_file_name)
79     input_file.read_line
    if input_file.last_string.is_integer_32 and then input_file.last_string.to_integer_32
        >= 0 and then input_file.last_string.to_integer_32 <= 100000000 then
81         last_read_integer := input_file.last_string.to_integer_32
    else
83         has_reading_failed := True
        print ("Reading input file failed!")
85     end
    input_file.close
87 ensure
    not has_reading_failed implies last_read_integer >= 0 and last_read_integer <=
    100000000
89 end

91 write_output_file
    -- Write output file.
93 local

```

```
    output_file : PLAIN_TEXT_FILE
95  do
    create output_file .make_open_write (Output_file_name)
97    output_file .put_string (last_converted_binary)
    output_file .close
99  end

101 feature -- Access

103  Input_file_name: STRING = "converter.in"
    -- The input file name.
105
106  Output_file_name: STRING = "converter.out"
    -- The output file name.
107

109  last_read_integer : INTEGER_32
    -- Last integer read.
111
112  last_converted_binary : STRING
113    -- Last converted binary.

115 feature -- Status report

117  has_reading_failed : BOOLEAN
    -- Did last reading fail?
119
end
```