Solution 5: References and assignments

ETH Zurich

1 City building

Solution

Listing 1: Class *CITY_BUILDING*

```
1 class CITY_BUILDING

3 inherit

5   TOURISM

7 feature -- City creation

9   explore is
        -- Create the city, central station and other needed objects.
11    local
        t: TIME
13      s: INTEGER
      do
15      create city.make ("New city")
        main_window.canvas.set_city (city)
17
        create t.make_now
19      s := t.hour
        s := s*60 + t.minute
21      s := s*60 + t.second
        s := s*1000 + t.milli_second
23      create random.set_seed (s)
        random.start
25
        create central_station.make_with_location ("Central station", 0, 0)
27      city.put_station (central_station)
        add_line
29      add_station (50,50)
        add_station (50,150)
31
      ensure
33      city_exists : city /= Void
        line_exists : line /= Void
35      central_station_exists : central_station /= Void
        random_exists: random /= Void
37    end
```

```
39  random: RANDOM

41  line: TRAFFIC_LINE

43  city: TRAFFIC_CITY

45  central_station: TRAFFIC_STATION

47  add_station (x, y: INTEGER)
        -- Add new station at coordinate (x, y) and extend the line.
49    require
        city_exists: city /= Void
51      line_exists: line /= Void
      local
53      p: TRAFFIC_STATION
      do
55      create p.make_with_location ("Station " + (city.stations.count).out, x, y)
        city.put_station (p)
57      line.extend (p)
      end
59
    add_line
61      -- Add new line.
      require
63      city_exists: city /= Void
        central_station_exists: central_station /= Void
65    local
        tram_type: TRAFFIC_TYPE_TRAM
67    do
        create tram_type.make
69      create line.make_with_terminal ("New line", tram_type, central_station)
        line.set_color (random_color)
71      city.put_line (line)
        Console.show ("New line added")
73    end

75  random_color: TRAFFIC_COLOR
        -- Generate random color.
77    require
        random_exists: random /= Void
79    local
        r, g, b: INTEGER
81    do
        random.forth
83      r := random.item \\ 256
        random.forth
85      g := random.item \\ 256
        random.forth
87      b := random.item \\ 256
        create Result.make_with_rgb (r, g, b)
89    ensure
```

```
        Result_exists : Result /= Void
91    end

93 end
```

1.1 Choosing between local variables and attributes

Thinking about the scope of variables (local or class wide) is very important. It can affect the readability, the efficiency and even the correctness of a program.

Local variables should be useful only in the feature scope they are declared in. Example of locals are:

- Variables declared using the keyword **local** at the beginning of a routine body
- **Result** in functions
- The arguments of a routine

An attribute should be used by more than one routine in the same class in which it is defined, or it should be accessed by other classes.

With time, you will get an intuitive understanding of whether a variable should be a local or an attribute. For now, we suggest you try to declare a variable as local first. If you then notice that you need access to that variable from other features of the class (or from other classes), then promote it to an attribute. If you do the other way round, you may never notice that you have unneeded attributes. Also see Touch of Class, section 9.1, page 231 ("Local variables") and subsequent pages.

2 Assignments

Solution

The solution lists the correct statements for each of the subtasks.

1. (a)
2. (d)
3. (d)
4. (b)
5. (c)
6. (e)
7. (b) (d)
8. (a)
9. (c) (e)

3 Programming a boardgame: Part 1

Solution

The classes we propose are the following:

- *GAME*
- *DIE*
- *PLAYER*
- *BOARD*
- *SQUARE*

We discarded *ROUND* and *TURN* for the moment because there does not seem to be enough "meat" in them. Additionally *PLAYER* and *TOKEN* represent the same abstraction for now. One can argue that there is not enough meat in *SQUARE* too, and that we should just be using integers for squares. Well, this may be true or not, depending on how the problem evolves. This is an example in which some experience (or knowledge of the problem domain) may help. After all, "squares" are not the same as integers (what's square -1? And what's square 102?), so it comes natural to use class *SQUARE* to restrict integer values.