Project Report
for
Touch of class - Examples Chapter 7 to 9

Damien Müllhaupt
Corinne Müller
ETH Zürich

January 18, 2008
Contents

1 Overview 3

2 Scope of work 3

3 Touch of class — Chapter 7 3
   3.1 Touch Paris objects 3
      3.1.1 Elysee palace 3
      3.1.2 Empty Line 4
   3.2 Missing features 4
      3.2.1 TOURISM: show_blinking_spot 4
      3.2.2 TRAFFIC_LINE: is_before 4
      3.2.3 TRAFFIC_PLACE: is_railway_connection 5
   3.3 Different names 5

4 Touch of class — Chapter 8 5

5 Touch of class — Chapter 9 6
   5.1 Missing features 6
      5.1.1 TRAFFIC_LINE: sw_end 6
      5.1.2 TRAFFIC_LINE: total_time 6

6 Touch of Class — Chapter 18 7

7 Touch of Class — Chapter 20 8

8 Touch of Class — Book fixes 8

9 Class Dictionary 8

10 Conclusions 9
1 Overview

This document serves as the project report for ”Software engineering laboratory” at ETH Zürich. Bertrand Meyer uses in his class ”Introduction to Programming” (for first term students) a teaching approach called the ”Inverted Curriculum” which gives students already at the beginning the opportunity to work with a large amount of software components (including multimedia and graphics) to activate their curiosity and help them to understand software in a different and more global way. Guided by the book ”TOUCH OF CLASS”, which is constantly in development, the students see software from a more and more detailed side by ”revealing the magic” with each chapter they read. This book contains many examples to help the students understand the basic principles of good software design and these examples are also the subject of this work. For the success of the book, it is crucial that the examples are easy to understand and help the students to get the essence of the concepts.

2 Scope of work

Most of the examples in TOUCH OF CLASS are not compatible with the current Traffic library which makes it more difficult for the students to learn the taught concepts, because they work with Traffic most of the time. So the key goal of this semester thesis was to implement some of the examples as close as possible to the book without destroying the design of the Traffic library or – if it can’t be achieved – to find and implement some new, more suitable examples covering the same concept. The hope and motivation of this work is to help students to understand the main concepts in a easy, comprehensible way.

3 Touch of class — Chapter 7

3.1 Touch Paris objects

3.1.1 Elysee palace

We added the missing object Elysee palace.

Listing 1: Excerpt from class TOUCH_PARIS_OBJECTS

```java
Elysee_palace : TRAFFIC_LANDMARK is
  -- The Elysee_palace building
  require
    Paris_exists : is_paris_loaded
  once
    create Result make (Paris_places.item ("Champs Elysee – Etoile"), position,
      File_system.absolutepathname (File_system.pathname_from_file_system ("..\map\paris\placedesvosges.png", Windows_file_system)))
  Paris.buildings.putLast (Result)
end
```
3.1.2 Empty Line

Class TOURISM should define an Empty Line to test some contracts. We added the following line to the paris.xml document: "line name="Empty_line" type="tram" simple="true"

Listing 2: Excerpt from class TOUCH_PARIS_OBJECTS

```
Empty_line: TRAFFIC_LINE is
  -- Line 1 of the Paris map
  require
    Paris_exists : is_paris_loaded
  once
    Result ?= Paris.lines.item ("Empty_line")
end
```

3.2 Missing features

3.2.1 TOURISM: show_blinking_spot

We added the feature show_blinking_spot, but the code is still very ugly.

Listing 3: Excerpt from class TOURISM

```
show_blinking_spot ( a_location : TRAFFIC_COORDINATE) is
  -- Show blinking spot
  local
    s: DRAWABLE_SPOT
  do
    create s.make (create {REAL_COORDINATE}.make (a_location.x, --a_location.y))
    s.set_color (create {EV_COLOR}.make_with_8_bit_rgb (255, 255, 0))
    s.enable_filled
    s.set_diameter (10)
  main_window.canvas.object_list.put last (s)
  short_wait
  main_window.canvas.object_list.delete (s)
  short_wait
  main_window.canvas.object_list.put last (s)
  short_wait
  main_window.canvas.object_list.delete (s)
  wait
end
```

3.2.2 TRAFFIC_LINE: is_before

The class TRAFFIC_LINE needs a feature which decide if there is a valid position to the left of the cursor.
Listing 4: Excerpt from class TRAFFIC_LINE

```java
is_before : BOOLEAN is
    -- Is there no valid position to left of internal cursor?
do
    Result := index = 0
end
```

3.2.3 TRAFFIC_PLACE: is_railway_connection

The class TRAFFIC_PLACE needs the feature is_railway_connection which gives back a boolean if a Metro station connects the railway network.

Listing 5: Excerpt from class TRAFFIC_PLACE

```java
is_railway_connection : BOOLEAN is
    -- Is this a railway connection
require
    not_empty : not outgoing_line_connections.is_empty
local
    found : BOOLEAN
    i : INTEGER
    do
    from
        i:=1
    until
        i = outgoing_line_connections.count + 1
    loop
        if outgoing_line_connections.item(i).type.name.is_equal("rail") then
            found:= True
        end
        i := i+1
    end
    result:= found
end
```

3.3 Different names

- TRAFFIC_PLACE: location — position (in the book it’s sometimes mention as position and sometimes as location)
- TRAFFIC_LINE: is_after — after
- TRAFFIC_PATH_SECTION: make.Metro — make_tram

4 Touch of class — Chapter 8

No new features had to be implemented for this chapter. Renaming and changes from features are listed on the Wiki page. See the chapter Touch of Class — Book fixes for more information.
5 Touch of class — Chapter 9

5.1 Missing features

5.1.1 TRAFFIC_LINE: sw_end

In the section "Local variables" of the book it’s asked for the `sw_end` of a TRAFFIC_LINE which was missing in the Traffic system.

Listing 6: Excerpt from class TRAFFIC_LINE

```plaintext
sw_end: TRAFFIC_PLACE is
2      -- returns the terminal which is in the south–west
4      -- End station on South or West side
       do
6      if not is_empty then
9      Result := terminal_1
11     end
14    end
```

5.1.2 TRAFFIC_LINE: total_time

To realise the feature total_time described in the book, two new feature had to be implemented into traffic. total_time takes the time from each segment of the route and adds the times together which is necessary to travel each of them. The feature total_time in TRAFFIC_LINE:

Listing 7: Excerpt from class TRAFFIC_LINE

```plaintext
total_time (speed:REAL_64):REAL is
2      -- Estimated travel time for the full line, time measured in Minutes.
4      do
6      from one_direction . start
9      Result:=0.0
12     until one_direction . index+1 > one_direction . count
15     loop
18     Result:=Result + one_direction . item_for.iteration . travel_time (speed)
21    end
24    end
```

Introducing total time we had to implement another feature which measures the time each segments takes to be traveled. The feature travel time in TRAFFIC_CONNECTION:

Listing 8: Excerpt from class TRAFFIC_CONNECTION

```plaintext
travel_time (speed: REAL):REAL_64 is
  -- calculates the travel time between 'origin' and 'destination'
  -- with a certain speed (km/h). Result is given in Minutes.
  local
    real_distance_m : REAL_64
    real_distance_km : REAL_64
    real_time : REAL_64
  do
    if length > 0 then
      real_distance_m := length * (map.scale_factor)
      real_distance_km := real_distance_m / 1000
      real_time := (real_distance_km / speed) * 60
    else
      real_time := 0.0
    end
  end
  Result := real_time
end
```

6 Touch of Class — Chapter 18

To make it easier for students to work with this chapter easier we decided to simplify the way taxis and taxi offices are treated in Traffic. Traffic had the deferred classes TRAFFIC_TAXI and TRAFFIC_TAXI_OFFICE. TRAFFIC_DISPATCHER_TAXI, TRAFFIC_DISPATCHER_TAXI_OFFICE, TRAFFIC_EVENT_TAXI and TRAFFIC_EVENT_TAXI_OFFICE were the actual implementations. The fact that taxi had two actual implementations was confusing for students working with Traffic and since there is no real need to make taxis deferred, we decides to create an actual implementation from TRAFFIC_TAXI and TRAFFIC_TAXI_OFFICE on the base of TRAFFIC_DISPATCHER_TAXI and TRAFFIC_DISPATCHER_TAXI_OFFICE. In the bon diagramm you can see how we changed TRAFFIC_TAXI, the same changes apply to TRAFFIC_TAXI_OFFICE.
7 Touch of Class — Chapter 20

No new features had to be implemented for this chapter. Renaming and changes from features are listed on the Wiki page. See the chapter Touch of Class — Book fixes for more information.

8 Touch of Class — Book fixes

We started a collection of things that are not correct or could be done better in the book. The collection can be found on the origo wiki pages, where we started a wiki page where people can add suggestions what can be done better in the book, or mistakes which were not corrected yet. We already filled the page with things who could be done from the chapters we had to cross check. The wiki page can be found under: http://traffic.origo.ethz.ch/wiki/touch_of_class_fixes

9 Class Dictionary

We started a class dictionary which should be used and extended in feature works. The goal is to have a complete dictionary which contains the translations of the class names from the Book to Traffic. This dictionary was also added in the Wiki for further usage.

<table>
<thead>
<tr>
<th>Touch of class</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEGMENT</td>
<td>TRAFFIC_PATH_SECTION</td>
</tr>
<tr>
<td>ROUTE</td>
<td>TRAFFIC_PATH</td>
</tr>
<tr>
<td>METRO_LINE</td>
<td>TRAFFIC_LINE</td>
</tr>
<tr>
<td>METRO_STATION</td>
<td>TRAFFIC_PLACE</td>
</tr>
</tbody>
</table>
10 Conclusions

All in all there would be a lot of work to bring the book to where traffic evolved today, or the inverse. Specially the renaming of features and classes which had to be done to evolve and reorganise Traffic and make it simpler or more efficient already is a huge difference to the book since almost every Traffic feature or class had a facelift from what it was in the beginning. We started a collection of things which should be done either in the book or in Traffic on our wiki page. We only made this for the chapters we had to review, doing more would have vastly exceeded the scope of our work. But this is a good start to bring the book up the level to where Traffic evolved and a good connection point for feature works to bring the book to where it should be. This also includes the use and evolution of the Class Dictionary we created on the Wiki, where class names are translated between the Book and Traffic.