

## Assignment 3: Of objects and features

Hand-out: 7 November 2005  
Due: 15 November 2005



Calvin and Hobbes© Bill Watterson

### 1 Classes vs. objects

#### Goal

- Understand the difference between *a class* and *an object*.

#### To do

- 1.1 Describe in your own words the difference between a class and an object.
- 1.2 Find an analogy in the real life.

#### To hand in

Write down your answers (1.1 and 1.2) and hand them in.

#### Solution

There is no unique solution. Sample answers:

- 1.1 A class describes the properties of a set of objects; an object is a member of such a set.

- 1.2 A class can be looked at as the blueprint of a machine, while the object is the actual machine built according to the blueprint.

## 2 Feature reading

### Goal

- Get used to EiffelStudio.
- Discover and browse Flathunt software.
- Get used to the “.” notation.

### Summary

- **Feature call**  
The fundamental mechanism of program execution: apply a “feature” to an “object”.  
Basic form: *your\_object.your\_feature*  
Example: *main\_window.show*
- **Expressions as targets**
  - Queries return a value (an object), e.g. *estate\_agent.location* yields an object of type TRAFFIC\_PLACE, the estate agent’s current location.
  - Since the result is an object, it is possible to apply features to it, e.g. *estate\_agent.location.set\_position (my\_position)*
  - Similarly, it is possible to use results of queries as arguments, e.g. *io.put\_string (estate\_agent.location.name)*
  - The result of an arithmetic expression (say  $x * 3 + 72$ ) is also an object on which you can call features, e.g.  $(x * 3 + 72).out$
- **Evaluation of expressions**  
Expressions built using the “.” notation are evaluated from left to right, e.g. *x.y.z.f* is evaluated as  $((x.y).z).f$

### To do

Consider the following features:

- *put\_string (s: STRING)*
- *increase (i: INTEGER)*
- *set\_state (b: BOOLEAN)*

Which of the expressions (2.1 - 2.5) could be used as arguments for the features given above? For each statement, write down its return type and the corresponding feature.



- What happens when you additionally press [SHIFT] at the same time?

### To hand in

Your answers to questions 2.1 - 2.5.

### Solution

- 2.1 STRING, *put\_string*
- 2.2 BOOLEAN, *set\_state*
- 2.3 STRING, *put\_string*
- 2.4 BOOLEAN, *set\_state*
- 2.5 INTEGER, *increase*

## 3 Modifying the game

### Goal

- Discover and browse Flathunt software.
- Get used to some naming conventions.
- Modify properties of objects.

### To do

Class GAME\_SCENE (in cluster View) describes game scenes in Flathunt. Feature *initialize\_scene* builds up and displays the main map, the status boxes, and the menus.

- 3.1 Read through the body of the feature and observe how the “.” notation is used, e.g. for setting the properties of *status\_box* and *player\_status\_box*.
  - What features are used for setting the properties of some object: commands or queries?
  - Have you noticed a particular naming convention for such features?
  - Can you guess whether a feature is a command or a query just by looking at its name?
  - Would you use verbs, nouns, or adjectives as command names? Would you use verbs, nouns, or adjectives as query names?
- 3.2 Change the title of the status box from “Status” to “Game status”.
- 3.3 Change the background color from black to white. Can you read the text in the status box now?

3.4 Change the font used for displaying the information in the status box from *Status\_font* to *Black\_status\_font*. Now you should be able to read the text in the status box.

What other fonts can you use? In which class are they defined?

3.5 Class `FLAT_HUNTER_DISPLAYER` declares the following feature:

```
1      statistics : STRING is
2          -- Location and number of tickets left.
3      do
4          Result := "Location: " + player.location.name +
5                  "%N Rail tickets: " + player.rail_tickets.out +
6                  "%N Tram tickets: " + player.tram_tickets.out +
7                  "%N Bus tickets: " + player.bus_tickets.out
8      end
```

This feature yields the information about a flat hunter; it is displayed in the status box when the given flat hunter takes her turn:

**Rail tickets: 4**  
**Tram tickets: 13**  
**Bus tickets: 6**

Change the implementation of feature *statistics* so that the total number of tickets owned by the player is also displayed:

**Rail tickets: 4**  
**Tram tickets: 13**  
**Bus tickets: 6**  
**Tickets in total: 23**

## Hint

3.1 Names of features that set some property usually start with the prefix “set\_”, e.g. *set\_color* sets the color of the object it is applied to. Such features are known as “setters”.

3.2 – Use an appropriate setter.  
– It only makes sense to call a feature on an object that already exists. Make sure that your feature call appears **after** the instruction that creates *status\_box*.

3.3 The standard background color is set using the feature *make\_black*. What object is it applied to? Where is the feature defined? What other features can be used for setting the color?

3.5 – You can simply use ‘+’ to concatenate strings.  
– “%N” represents the ‘new line’ symbol in strings.

- Feature *out* yields a string representation of the object it is applied to. For example, applied to an object of type INTEGER whose value is 6, it will return a STRING object "6". Feature *out* is available in all classes.

## To hand in

Submit modified classes GAME\_SCENE and FLAT\_HUNTER\_DISPLAYER to your assistant.

## Solution

```
-- In class GAME_SCENE
initialize_scene is
  -- Build 'main_container' containing zoomable map.
  do
    background_color.make_white -- background_color.make_black

    -- Build map widgets.
    main_container.extend (big_container)
    build_little_map_widget

    -- Build navigation widget to connect
    -- little map to big map for navigation.
    create navigation_widget.make (little_zoomable_container, big_zoomable_widget)

    -- Build status box.
    create status_box.make_from_coordinates (Map_area_width + 2 * Margin, Window_width - Map_area_width -
      2 * Margin, Window_width - Margin, Map_area_height, "Status")
    status_box.set_title ("Game status") -- status_box.set_title ("Status")
    status_box.set_font (Black_status_font) -- status_box.set_font (Status_font)
    status_box.set_title_font (Medium_game_widget_font)
    status_box.set_color (Game_widget_color)
    status_box.set_opacity (70)
    status_box.set_padding (-3)
    main_container.extend (status_box)
    update_status_box
  ...
end
```

Figure 2: GAME\_SCENE

```
indexing
  description: "Displays a flat hunter on the board."
  date: "$Date: 2005/11/03 17:46:00 $"
  revision: "$Revision: 1.7 $"

class
  FLAT_HUNTER_DISPLAYER

inherit
  PLAYER_DISPLAYER
  redefine
    player, statistics
  end

create
  make_from_player

feature -- Output

  statistics: STRING is
    -- Location and number of tickets left.
  do
    Result := "Location: " + player.location.name +
      "%NRail tickets: " + player.rail_tickets.out +
      "%NTram tickets: " + player.tram_tickets.out +
      "%NBus tickets: " + player.bus_tickets.out +
      "%NTickets in total: " +
      (player.rail_tickets + player.tram_tickets + player.bus_tickets).out
  end

feature {NONE} -- Implementation

  player: FLAT_HUNTER
    -- Reference to player to be displayed.

end
```

Figure 3: FLAT\_HUNTER\_DISPLAYER