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 \(x \times 3 + 72\)) is also an object on which you can call features, e.g. \((x \times 3 + 72).out\)

• 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.
Example:
Question: game.current_player_index in feature update_status in class MAIN_CONTROLLER
Answer: INTEGER, increase

2.1 game.current_player.name in feature status_overview in class MAIN_CONTROLLER

2.2 current_player.possible_moves.is_empty in feature prepare in class GAME

2.3 game_scene.player_displayers.i.th (game.current_player_index).statistics in feature status_before_prepare in class MAIN_CONTROLLER

2.4 possible_moves.item.destination.name.is_equal (selected_place.name) in feature choose_next_move in class HUMAN

2.5 option_panel.option_menus.item.selected_entry in feature start_callback in class START_MENU_SCENE

Hint
• To navigate between classes and features in EiffelStudio, you can use the ‘pick-and-drop’ technique. Just ‘pick’ a class or a feature (by right-clicking on its name) and ‘drop’ it in another pane within EiffelStudio, and see what happens.
• In the text editor, when you type the name of an entity followed by a dot, EiffelStudio will automatically display a list of all the features that can be called at the current position (see Figure 1). To get the list of all features applicable to the Current object, press [CTRL] + [SPACE].

Figure 1: Intellisense
• 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:

```plaintext
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
  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 \textit{out} yields a string representation of the object it is applied to. For example, applied to an object of type \texttt{INTEGER} whose value is 6, it will return a \texttt{STRING} object "6". Feature \textit{out} is available in all classes.

To hand in

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

Solution

```plaintext
-- In class \texttt{GAME\_SCENE}
initialize_scene is
  do
    background_color.make_white  -- background_color.make_black
    -- Build map widgets.
    main_container.extend (big_container)
    buildAnyway_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, Map_area_height - Map_area_width / 2 + 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 (Black_font)
    status_box.set_title_font (Medium_game_widget_font)
    status_box.set_color (Game_widget_color)
    status_box.set_opacity (50)
    status_box.set_border (-1)
    main_container.extend (status_box)
    update_status_box
  end

Figure 2: \texttt{GAME\_SCENE}
Figure 3: FLAT_HUNTER_DISPLAYER.

```plaintext
indexing

  description: "Displays a flat hunter on the board."
  date: "$Date: 2005/11/29 19:46:00 $"
  revision: "$Revision: 1.7 $"

class
  FLAT_HUNTER_DISPLAYER

inherit
  PLAYER_DISPLAY

redefine
  player, statistics

end

creates
  make_tree_player

feature -- Output

  statistics: STRING is
    -- Location and number of tickets left.
    do
      Result := "Location: \" + player.location.name +
                "Rail tickets: \" + player.rail_tickets.out +
                "Train tickets: \" + player.train_tickets.out +
                "Tram tickets: \" + player.tram_tickets.out +
                "Tickets in total: \" +
                player.rail_tickets + player.tram_tickets + player.tram_tickets
    end

end

feature -- Implementation

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

end
```