

## 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.

### 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.

**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\_displays.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].

```

a_hunter_count_valid. a_hunter_count < 0 and a_hunter_count > 0
do
  make_scene_default
  traffic_map := a_traffic_map
  hunter_count := a_hunter_count
  build_big_map_widget
  status_box.
  -- Set default
  create status_box
  create hash
  create plays
  paused := False
  game_over := False
ensure
  traffic_map
  hunter_count
end
-- In class GAME_SCENE
initialize_scene is
  -- Build 'ms
do
  background_c
  build_wid

```

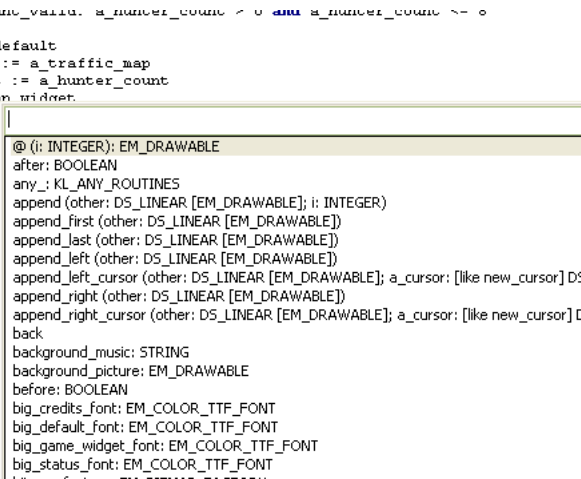


Figure 1: Intellisense

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

## To hand in

Your answers to questions 2.1 - 2.5.

### 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
           -- Location and number of tickets left.
3      do
           Result := "Location: " + player.location.name +
5              "%NRail tickets: " + player.rail_tickets.out +
              "%NTram tickets: " + player.tram_tickets.out +
7              "%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 *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. Don’t forget to submit your compilation logs.