Assignment 5: Loops

ETH Zurich

Hand-out: 20st November 2006
Due: 28th November 2006

1 Summary

In this assignment are going to experiment with loops in Eiffel...

In this assignment are going to experiment with loops in Eiffel...
In this assignment are going to experiment with loops in Eiffel...
In this assignment are going to experiment with loops in Eiffel...

... Please solve this assignment alone.

An example

```
from
  initialization_instructions
invariant
  invariant_clause
variant
  variant_clause
until
  exit_condition
loop
  loop_instructions
end
```

Figure 1: Structure of a loop

A loop is composed of several clauses. The `from` clause is required (but may be empty), it specifies the loop initialization instructions. The `variant` and `invariant` clauses are optional. Leaving aside the optional clauses, the execution of a loop consists in performing the `initialization_instructions` followed by the “loop process”. If the `exit_condition` is true, the loop process is a null instruction; if it is false, the loop process is the execution of the `loop_instructions` followed by a new loop process. The structure of a loop is shown in Figure 1.
loop_example is
  −− A loop example...
local
count: INTEGER
end
from
count := 1
invariant
count >= 1
count <= 101
variant
101 - count
until
count > 100
loop
io.put_integer (count)
io.put_new_line
count := count + 1
end
end

Figure 2: Loop example

An example of a loop in Eiffel is given in Listing 2. This code prints 100
numbers on the console (1,...,100).

2 Where is Central?

Goal

• Understand the structure of loops and conditionals.

• Realize the importance of proper stop criteria.

Description

This first part intends to convey the importance of choosing the right stop
criteria. In the following class text extract we want to loop through a list of
places and search for the place called “Central”. When we have found it, the
loop stops and we do something to the place (if it was found). The two code
extracts below are supposed to do everything as just described.

To do

1. For each version decide whether it does what it is supposed to.

2. If you think it does not do what it is supposed to, then correct the errors.
Note

1. You may assume for this exercise that all the entities are not Void (i.e. they are all attached to an object).
2. The command set_found just sets found to either True or False.
3. name = "Central" is not the same as name.is_equal("Central")

```eiffel
from
places.start
set_found (False)
until
places.after or found
loop
  if (places.item.name = "Central") then
    set_found (True)
  else
    places.forth
  end
  if (not places.after) then
    "Perform some operations on the found place"
  end
end
```

Figure 3: Version A

```eiffel
from
places.start
until
places.after or places.item.name.is_equal ("Central")
loop
end
if (not places.after) then
  "Perform some operation on the found place"
end
```

Figure 4: Version B

To hand in

This is a pen-and-paper exercise: you do not need to code in EiffelStudio. Hand in your answers and the corrected versions of a) and b) (if necessary).
3 Fancy graphics

Goal

- Play around with loops and conditionals.
- Be creative and make $FLAT\_HUNT$ look nicer.

Description

In class $PLAYER\_DISPLAYER$ in $FLAT\_HUNT$, there is a feature called $mark\_defeat$. This feature gets called either on the estate agent when the hunters find him, or on all the hunters if the estate agent escaped. Up to now, $mark\_defeat$ just draws a black circle. There is a loop prepared, but for now, this loop is empty. Your task is to fill this loop: try to make some nicer graphics whenever the game is over. This could, for example, look like Figure 5. However, instead of circles, you might also want to use lines or rectangles. Another idea is to play with colors... It would probably look even better if you would add some conditionals..

![Figure 5: Better graphics...](image)

Note

- Make sure that all assertion checking in Project $\rightarrow$ Configuration are enabled.
• Have a look at the comments in the source code.
• Write the variant and invariant clauses.

To hand in
• Hand in your version of feature \texttt{mark\_defeat}, plus a screenshot of your animation in a graphical format (i.e. jpg, png, gif, etc.) other than bitmap (bmp).
• Don’t forget to upload your logs.
4 Loop painting

Goal

- To write nested loops.

Description

You can use loops within loops to display certain figures, like the one in Figure 6.

![Image of a checkered triangle](C:\MyProjects\checkerboards\EFI\GEN\W_code\checkered_triangle.png)

Enter the dimension: 8

```
  *  
**   
*   *
**  
**  
*   *
**   
  *
```

Press Return to finish the execution...

Figure 6: Example with size 7

To do

1. Write a program that asks the user to input a value, and then displays a checkered triangle of the given size as in Figure 6. Be aware that stars and white space should be alternating.

To hand in

Hand in your class text. Don’t forget to upload your logs.