Assignment 10: Observable secrets

ETH Zurich

Due: 4. December 2008

Copyright Bill Watterson

Goals

• Test your understanding of agents and event driven programming.
• Encrypt and decrypt messages.

1 Observing the temperature

This is an old exam question. Given is the class TEMPERATURE_SENSOR. Assume that there is a hardware component that updates the value attribute by calling the procedure set_value whenever the temperature changes.

Listing 1: Class TEMPERATURE_SENSOR

```haskell
class TEMPERATURE_SENSOR
feature -- Initialization
    set_value (a_value: REAL) is
        -- Set ‘value’ to ‘a_value’.
        do
            value := a_value
        ensure
```
11     value_set: value = a_value
          end

13

15 feature  --  Access

17     value: REAL
           --  Temperature value in degrees celcius

19     end

To do

To implement it with EiffelStudio, download the files for TEMPERATURE_SENSOR and HEATING_CONTROLLER from http://se.ethz.ch/teaching/2008-H/eprog-0001/exercises/observable_temp.zip.

1. Write a class OBSERVABLE_TEMPERATURE_SENSOR that inherits from the given class TEMPERATURE_SENSOR. Your implementation should allow observers to register procedures. These procedures are called with the new temperature value as an argument whenever the set_value feature of OBSERVABLE_TEMPERATURE_SENSOR is invoked by the hardware. You may use any publish-subscribe mechanism for Eiffel shown in the course that satisfies the following conditions:

   • An observer can observe any number of publishers.
   • A publisher does not know its observers, but only the procedures to call when the temperature changes.

class OBSERVABLE_TEMPERATURE_SENSOR

inherit

TEMPERATURE_SENSOR
2. Implement the observer class `HEATING_CONTROLLER` that uses your implementation of `OBSERVABLE_TEMPERATURE_SENSOR`. The feature `set_sensor` should do three things: (1) it should set `sensor` to the sensor passed as an argument, (2) it should register the procedure `adjust_heating` to be called whenever a change to the value of the new sensor happens, and (3) if the procedure `adjust_heating` was already subscribed to the previous sensor, then it should unsubscribe it from this old sensor. You may add new features or redefine features from class `ANY` to complete the implementation.

class `HEATING_CONTROLLER`
feature −− Element change

set\_sensor (a\_sensor: OBSERVABLE\_TEMPERATURE\_SENSOR) is
  −− Set ‘sensor’ to ‘a\_sensor’.
  −− Unsubscribe ‘adjust\_heating’ from old sensor (if existing).
  −− Subscribe ‘adjust\_heating’ to new sensor.
  do
end

feature -- Given features

adjust_heating (a_value: REAL) is
  -- Output what to do concerning the heating.
  do
    if a_value >= 20.0 then
      io.put_string ("Turn off heating")
      io.new_line
    else
      io.put_string ("Turn on heating")
      io.new_line
    end
  end

sensor: OBSERVABLE_TEMPERATURE_SENSOR
  -- Currently used sensor
end

To hand in
Submit your answers to your assistant.

2 Ciphers

One of the classical cryptographic techniques for enciphering text is known as the Vigenere cipher. It derives from the Caesar cipher, in which each letter of the alphabet is shifted along some number of places; for example, in a Caesar cipher of shift 3, A would become D, B would become E and so on. The Vigenere cipher consists of several Caesar ciphers in sequence with different shift values.

To implement the Vigenere cipher the letters of the alphabet A, B, ..., Z are associated with the numbers 0, 1, ..., 25. The plain text message can then be viewed as a sequence of numbers: \( P = [P_1, P_2, ..., P_n] \) where \( P_i \in \{0, 1, ..., 25\} \) and \( i \in \{1, 2, ..., n\} \). To encrypt the plain text, the algorithm generates a key of the same length as the plain text by repeating a secret pass phrase. The key can also be viewed as a sequence of numbers between 0, 1, ..., 25: \( K = [K_1, K_2, ..., K_n] \)
where $K_i \in \{0, 1, \ldots, 25\}$ and $i \in \{1, 2, \ldots, n\}$. The message is encrypted by adding the key item by item to the plain text modulo 26: $C = [C_1, C_2, \ldots, C_n]$ where $C_i = (P_i + K_i) \mod 26$ and $i \in \{1, 2, \ldots, n\}$. This code is then presented to the user as a sequence of letters again. To make things easier, only uppercase alphabetic letters should be encrypted and all other characters (such as digits, spaces, commas, etc.) are not encrypted. The following example illustrates this using the pass phrase "TIGER":

Plain text: STUDENTS, SOLVE THE ASSIGNMENT WELL AND FAST!
Key: TIGERTIG, ERTIG ERT IGERTIGERT IGER TIG ERTI!
Code: LBAHVGBY, WFEDK XXY IYWZZVSIEM EKPC TVJ JRLB!

To increase the security of the cipher, we add a second cryptographic technique called spiral cipher. The encrypted message from above will be passed as plain text to this new cipher. The spiral cipher takes the text and writes it row by row into a quadratic matrix. It then generates the encrypted message by reading it out in a clockwise spiral pattern starting in the right top corner of the matrix. The size of the matrix should be large enough to store the entire text, but not larger than needed. If the text is smaller than the number of cells in the matrix, the remaining cells are filled with spaces. Note that this cipher does not require a user-defined key.

To do

1. Create a new project.
2. Implement a deferred class `CIPHER` that has two deferred features: encrypt and decrypt. Then implement a class `VIGENERE_CIPHER` that uses the cryptographic technique of the Vigenere cipher described above to encrypt and decrypt messages. Also implement a class `SPIRAL_CIPHER` that implements the spiral cipher technique from above. Both classes should inherit from `CIPHER` and effect its deferred features.
3. Write an effective class `COMBINED_CIPHER` inheriting from `CIPHER`. A combined cipher stores a list of ciphers (descendants of `CIPHER`). Its encrypt feature takes the message given as argument and encrypts it using the first cipher of the list. Then it uses the outcome of this encryption as input to the second cipher, and so on. The decrypt feature reverses this process.
4. Create an instance of `COMBINED_CIPHER` and add an instance of `VIGENERE_CIPHER` as its first cipher and an instance of `SPIRAL_CIPHER` as second cipher. Encrypt the message "MYLASTASSIGNMENT" using the pass phrase "BUSY" with your combined cipher and write the encrypted message into the line called "Pass phrase" of Step 5:

   Pass phrase: BUSY
   Message: MYLASTASSIGNMENT
5. Use the encrypted message from Step 4 as pass phrase to decrypt the code below. Note that the code is available on the Internet to help you copy paste it into your application http://se.ethz.ch/teaching/2008-H/eprog-0001/exercises/encrypted_message.txt

Pass phrase: ..........................................................

Code: LAGHYH QSCUJRKS.IRNTEFRPRXMOIY CGAEYXWYGNENLRF FAZ2/TGBZKFI REE.RWD Y LU! LMK SLXG.E./PMAUWHCGYZGYFHEIWQ QUG RSS.XUVIUL ZIE KFYGEL//:RLH DXE Z

Hints

• Every symbol of type CHARACTER has an associated character code (an integer number) available through the feature code. For the letter A this code is 65, for B it is 66, . . . , and for Z it is 90. The class CHARACTER offers features that allow arithmetic operations based on these character codes (see features infix "+" and infix "-"). Note that the first operand needs to be a CHARACTER and the second operand needs to be the INTEGER, it doesn’t work the other way around.

| Given the following declarations | c1, c2: CHARACTER  
i: INTEGER |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c2 receives the character that has a code that is by ‘i’ larger than the code of c1</td>
<td>c2 := c1 + i</td>
</tr>
<tr>
<td>c2 receives the character that has a code that is by ‘i’ smaller than the code of c1</td>
<td>c2 := c1 - i</td>
</tr>
</tbody>
</table>

• In Eiffel, integer division is done with ‘/’, integer remainder (modulo) with ‘\’.

• The class DOUBLE_MATH offers a feature sqrt to calculate the square root of a number. Class DOUBLE offers floor to round down and ceiling to round up to integer numbers.

• The class ARRAY2 represents two-dimensional arrays.

To hand in

Send the source code of your application to your assistant.

Feedback on assignments 9 and 10

The feedback form for assignments 9 and 10 is available online at http://elbanet2.ethz.ch/survey/entry.jsp?id=1227798924373.

Please take a couple of minutes to complete the questionnaire... Thanks!