

# Assignment 10: Observable secrets

ETH Zurich

Hand-out: 28. November 2008  
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`

```
1 class
2   TEMPERATURE_SENSOR
3
4   feature -- Initialization
5
6   set_value (a_value: REAL) is
7     -- Set 'value' to 'a_value'.
8     do
9       value := a_value
10    ensure
```

```
11     value_set: value = a_value
12     end
13
15 feature -- Access
16
17     value: REAL
18         -- Temperature value in degrees celcius
19
20 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](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
```

```
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
```





```
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
  
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  
  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:  $P = [P_1, P_2, \dots, P_n]$  where  $P_i \in \{0, 1, \dots, 25\}$  and  $i \in \{1, 2, \dots, n\}$ . 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:  $K = [K_1, K_2, \dots, K_n]$

where  $K_i \in \{0, 1, \dots, 25\}$  and  $i \in \{1, 2, \dots, n\}$ . The message is encrypted by adding the key item by item to the plain text modulo 26:  $C = [C_1, C_2, \dots, C_n]$  where  $C_i = (P_i + K_i) \bmod 26$  and  $i \in \{1, 2, \dots, n\}$ . 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 XYX 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.

L	B	A	H	V	G	B
Y	,		W	F	E	D
K		X	Y	X		I
Y	W	Z	Z	V	S	I
E	M		E	K	P	C
	T	V	J		J	R
L	B	!				

Plain text: LBAHVGBY, WFEDK XYX IYWZZVSIEM EKPC TVJ JRLB!  
 Code: BDIICR !BL EYKYLBAHVGE SPJ JVTMW , WFXVKE ZXYZ

## 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](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./PMAWUHWCHGYZGYFHEIWG 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
c2 receives the character that has a code that is by 'i' larger than the code of c1	c2 := c1 + i
c2 receives the character that has a code that is by 'i' smaller than the code of c1	c2 := c1 - i

- 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!**