Typology of programming languages

~ Eiffel ~

Reading...

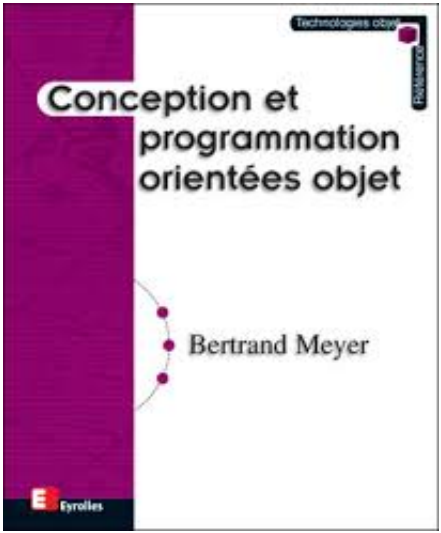

Technologies objet
Conception et programmation orientées objet
Bertrand Meyer
Eyrolles

Table of Contents

Bertrand Meyer (1950), MIT

Table of Contents

Introducing Eiffel

- High-level language designed for Software Engineering, portable, with an original and clear syntax
- Modern conception of multiple class inheritance
- High level tools and programmatic concepts (Virtual classes, Generics, Exceptions, etc.)
- Lot of standard libraries

Libraries

EiffelCOM (COM,OLE,ActiveX),
EiffelCORBA,
EiffelMath,
EiffelNet (client-serveur),
EiffelLex & *EiffelParse*,
EiffelStore (BD),
EiffelWEB,
Eiffel DLE (dynamic link),
EiffelVision (GUI),
Graphical Eiffel for Windows, *Eiffel WEL* (Windows),
EiffelThreads,
etc.

An Eiffel Application

An Eiffel Application is called a *system*.

- Classes :
  - One per file (`.e`)
  - Groupped in *clusters*
  - One one them is the main class
- Eiffel Librairies (only one in practice)
- External Librairies
- A file describing the application
  - LACE file, *Langage pour l'assemblage des classes en Eiffel*

Clusters

LOGICAL point-of-view
: Set of classes building an antonomous part of the application

PHYSICAL point-of-view
: All these classes lay in the same repository

LACE point-of-view
: A cluster is a name associated to a repository

LACE File Example

```
system
   geo

root
   TEST(TEST): "main"

default
   precompiled("$EIFFEL3/precomp/spec/$PLATFORM/base")

cluster
   TEST:   "$EIFFELDIR/TEST" ;
   FIGS:   "$EIFFELDIR/FIGURES" ;

external
   object: "$(EIFFEL3)/library/lex/spec/$(PLATFORM)/lib/lex.a"

end
```

Original Concepts

Adaptation clauses for inheritance
: resolve multiple inheritance problems

Contract Programming
: Promote reusability and modularity

Graphical User Interface
: A full dedicated GUI: drag-and-drop, etc.

A smart compiler

Compiler with three modes *really usefull in developpement phases*

FINALIZING Optimisation and production of an executable file where all optimizations have been applied. May be very slow!

FREEZING compile and produce an executable file

MELTING compilation by patches. Very fast, a modification only recompile what is necessary (not good performance, useful for

A full System

EiffelBench the visual workbench for object-oriented development

EiffelBuild the editor to build GUI

EiffelCase the tools dedicated to build and design application

Ebench

Edit a Class

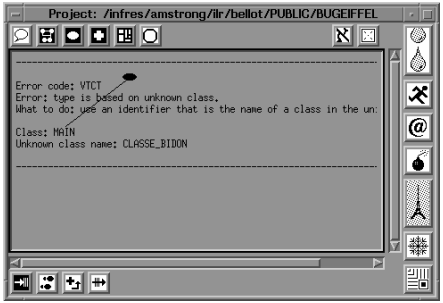

Project: /infres/amstrong/ilr/bellot/PUBLIC/BUGEIFFEL
Error code: VTCT
Error: type is based on unknown class.
What to do: use an identifier that is the name of a class in the un
Class: MAIN
Unknown class name: CLASSE_BIDON

Table of Contents

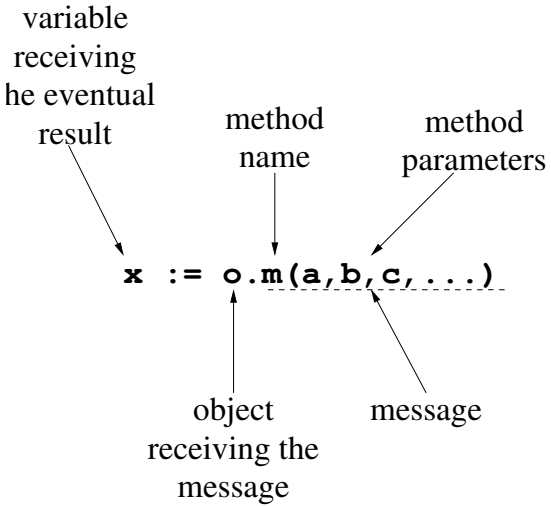
Example of a Class

```
class POINT
  -- a 2-d point

feature
   -- coordinates
   xc,yc : INTEGER ;

   -- Change coordinates
   set_x_y(x,y : INTEGER) is
     do
        xc := x ;
        yc := y ;
     end ;
end -- class POINT
```

Methods Calls

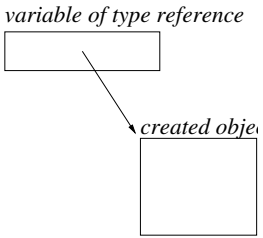


- The object execute the method `m` which is executed in its own context.
- For **distributed** objects, a message is sent, **otherwise** it is a simple procedure call.

Creating an Object

```
p : POINT ;

do
    ...
    !! p ; -- instruction
    ...
end ;
```



Except for explicit declaration, all the object's variables are *references*: they handle pointers.

Creation with Initialization 1/2

```
class POINT
create
   make  -- init method
feature
   -- init method
   make(x,y : INTEGER) is
     do
        set_x_y(x,y) ;
     end ;

   -- same as previously
end -- class POINT
```

- Attributes are initialized with a default value (e.g., 0 for an Integer, `Void` for a variable with type reference).
- **If we want to initialize an object during creation, we must build a initialization method**

Creation with Initialization 2/2

The object can then be created using its initialization method

```
p : POINT ;
create p.make(23,64) ;; -- create and initialize a Point
```

- Multiple initialization methods can be defined for a same class. **The correct method is chosen during the creation.**
- When (at least) one initialization method is declared for an class, this class cannot be created without calling one of these routines.
  $\Rightarrow$ Security

Access to Class Member Variables

READING By default, all members are readable: everyone can know the value of it (but restriction can be applied).

WRITTING Members are NEVER writtable **except for the current object**. The object mus provide a setter!
`set_x_y(x,y : INTEGER)`
de la classe POINT.
$\Rightarrow$ Security

Access to Class Member Variables

READING By default, all members are readable: everyone can know the value of it (but restriction can be applied).

WRITTING Members are NEVER writtable **except for the current object**. The object mus provide a setter!
`set_x_y(x,y : INTEGER)`
de la classe `POINT`.
$\Rightarrow$ Security

Method without arguments doesn't have an empty pair of parenthesis: this helps to keep API stable

Expanded Class

By default, all types are manipulated by reference
(i.e., with indirection)

Performances issues ...

```
expanded class ERECTANGLE
inherit
RECTANGLE ;
end -- class ERECTANGLE
```

Like current

Let us consider the following snippet from class SHAPE:

```
copy_and_move(x,y : INTEGER) : SHAPE is do
   Result := Current.copy() ;
   Result.set_x_y(x,y) ;
end ;
```

And the following code :

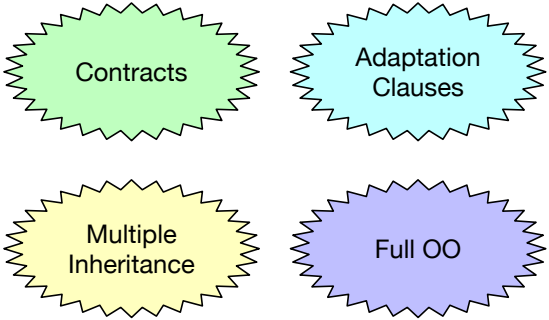
```
C1 := C2.copy_and_move(22,23)
```

- No problem when C1 and C2 are SHAPE
- Problem: type is lost when C1 and C2 are of type SQUARE (that inherits from SHAPE)
  $\Longrightarrow$ **Effeil proposes like x or like current to solve this!**

Eiffel Overview

- An object-oriented program structure in which a class serves as the basic unit of decomposition
- Static Typing
- Protection against calls on null references, through the attached-types mechanism
- Objects that wrap computations (closely connected with closures and lambda calculus)
- Garbage Collection
- Simple Concurrent Object-Oriented Programming
- Constrained and unconstrained generic programming *in a latter*

Summary


Contracts
Adaptation Clauses
Multiple Inheritance
Full OO