Fondamenti della Programmazione: Metodi Evoluti

Prof. Enrico Nardelli

Esercitazione 1

Let's play to be the compiler

Try to understand whether the code is correct!

Compilation error? (1)

Hands-On

```
class PERSON
feature
        name: STRING

        set_name (a_name: STRING)
                do
                        name := a_name
                end
        exchange_names (other: PERSON)
                local
                        s: STRING
                do
                        s := other.name
                        other.set_name (name)
                        set_name (s)
                end
        print_with_semicolon
                do
                        s := "Hello"
                        s.append("!")
                        print (s)
                end
end
```

Error: this variable was not declared

Compilation error? (2)

Hands-On

```
class PERSON
feature
        ...         -- name and set_name as before

        exchange_names (other: PERSON)
                local
                        s: STRING
                do
                        s := other.name
                        other.set_name (name)
                        set_name (s)
                end

        print_with_semicolon
                local
                        s: STRING
                do
                        s := "Hello"
                        s.append('!')
                        print (s)
                end
end
```

OK: two different local variables in two routines

Compilation error? (3)

Hands-On

```
class PERSON
feature
        ...         -- name and set_name as before

        s: STRING

        exchange_names (other: PERSON)
                do
                        s := other.name
                        other.set_name (name)
                        set_name (s)
                end

        s: STRING

        print_with_semicolon
                do
                        s := "Hello"
                        s.append('!')
                        print (s)
                end
end
```

Error: an attribute with the same name was already defined

Compilation error? (4)

Hands-On

```
class PERSON
feature
        ...         -- name and set_name as before

        exchange_names (other: PERSON)
                do
                        s := other.name
                        other.set_name (name)
                        set_name (s)
                end

        print_with_semicolon
                do
                        s := "Hello"
                        s.append("!")
                        print (s)
                end

        s: STRING
end
```

OK: a single attribute used in both routine

Compilation error? (5)

Hands-On

```
class PERSON
feature
        ...         -- name and set_name as before

        exchange_names (other: PERSON)
                do
                        s := other.name
                        other.set_name (name)
                        set_name (s)
                end

        print_with_semicolon
                local
                        s: STRING
                do
                        s := "Hello"
                        s.append("!")
                        print (s)
                end

        s: STRING
end
```

Error: an attribute with the same name was already defined

Local variables vs. attributes

- Which one of the two correct versions (2-local and 4-attribute) do you like more? Why?

- Describe the conditions under which it is better to use a local variable instead of an attribute and vice versa

Find 5 errors

```
class VECTOR
feature
        x, y: REAL

        copy_from (other: VECTOR)
                do
                        Current := other
                end

        copy_to (other: VECTOR)
                do
                        create other
                        other.x := x
                        other.y := y
                end

        reset
                do
                        create Current
                end
end
```

Current is not a variable and can not be assigned to

other is a formal argument (not a variable) and thus can not be used in creation

other.x is a qualified attribute call (not a variable) and thus can not be assigned to

the same reason for *other.y*

Current is not a variable and thus can not be used in creation

Installation

A couple of exercises to practice with

- Compilation
- Code browsing

A first useful practical thing

To create a new project

Menu File -> New Project ->

Basic applications (no graphics library)

EiffelStudio Components

- editor
- context tool
- clusters pane
- features pane
- compiler
- project settings
- …

Editor

- Syntax highlighting
- Syntax completion
- Auto-completion (CTRL+Space)
- Class name completion (CTRL+SHIFT+Space)
- Smart indenting
- Block indenting or unindenting (TAB and SHIFT+TAB)
- Block commenting or uncommenting (CTRL+K and SHIFT+CTRL+K)
- Unbounded number of Undo/Redo (reset after a save)
- Quick search features (first CTRL+F to enter words then F3 and SHIFT+F3)
- Pretty printing (CTRL+SHIFT+P)

Self presentation

```
class
    APPLICATION
feature
    make
            -- self presentation.
        do
            Io.put_string("Mi chiamo: ")
            Io.put_string("Mario Rossi%N") -- is a new_line character
            Io.put_string("La mia eta' e' :")
            Io.put_integer(24)
            Io.new_line   -- same as %N at the end of the string
            Io.put_string("Sono tifoso della Lazio? ")
            Io.put_boolean(False)
            Io.new_line
        end
end
```

Eiffel Naming: Classes

- Full words, no abbreviations (with some exceptions)
- Classes have global namespace
  - Name clashes arise
- Usually, classes are prefixed with a library prefix
  - Traffic: TRAFFIC_
  - EiffelVision2: EV_
  - Base is not prefixed

Eiffel Naming: Features

- Full words, no abbreviations (with some exceptions)
- Features have namespace per class hierarchy
  - Introducing features in parent classes, can clash with features from descendants

Eiffel Naming: Locals / Arguments

- Locals and arguments share namespace with features
  - Name clashes arise when a feature is introduced, which has the same name as a local (even in parent)
- To prevent name clashes:
  - Locals are prefixed with **l_**
  - Arguments are prefixed with **a_**
- But exceptions may exist...

Self presentation: interactive version

```
class
    APPLICATION
feature
    make
            -- self presentation, interactive version.
        local
            nome: STRING
            eta: INTEGER
        do
            Io.put_string("Scrivi il tuo nome e premi Return: ")
            Io.read_line
            nome := Io.last_string

            Io.put_string("Scrivi la tua eta' e premi Return: ")
            Io.read_integer
            eta := Io.last_integer

            print("Mi chiamo: ")
            print(nome+"%N") -- is a new_line character
            print("La mia eta' e': ")
            print(eta)
            print("%N")
            print("Sono tifoso della Lazio? ")
            print(False)
            print("%N")
        end
end
```

Compilation Stages I

Degree 6: Examining System

Degree 5: Parsing Classes

Degree 4: Analyzing Inheritance

Degree 3: Checking Types

Degree 2: Generating Byte Code

Degree 1: Generating Metadata

Workbench vs. Finalizing

Workbench mode

Assertions (contracts)

No optimizations

Debugging possible

Slow / safe

Finalized mode

No assertions

Code optimizations

No Debugging

Fast / unsafe

Compilation Stages II

Workbench mode

Degree -1: Generating Code

Finalized mode

Degree -2: Constructing Polymorphic Table

Dead Code Removal

Degree -3: Generating Optimized Code

Compiler highlights

- **Melting**: uses quick incremental recompilation to generate bytecode for the changed parts of the system. Used during development (corresponds to the button "Compile").

- **Freezing**: uses incremental recompilation to generate more efficient C code for the changed parts of the system. Initially the system is frozen (corresponds to "Freeze...").

- **Finalizing**: recompiles the entire system generating highly optimized code. Finalization performs extensive time and space optimizations (corresponds to "Finalize..."), this may take a long time to do.