Fondamenti della Programmazione: Metodi Evoluti

Prof. Enrico Nardelli

Esercitazione 6

Remember the Acrobat and Copycat game

- There were Acrobat and Copycat objects
- The Acrobat was asked to **Clap** and given a number. It had to clap its hands that many times and pass the same instruction to its Copycat, who had to just to **Clap**
- The Acrobat was asked to **Twirl** and given a number. It had to turn completely around that many times and pass the same instruction to its Copycat, who had just to **Twirl**
- When asked for **Count**, the Copycat provided the total number of actions it had executed until them. The Acrobat instead asked its Copycat and answered with the number it provided

There was a *COPYCAT*

```
class
        COPYCAT
feature
        clap (n: INTEGER)
                        -- Clap `n' times and adjust `count'.
                require n>0
                do
                -- to be completed
                ensure  count = old count + n
                end
        twirl (n: INTEGER)
                        -- Twirl `n' times and adjust `count'.
                require n>0
                do
                -- to be completed
                ensure  count = old count + n
                end
        count: INTEGER
                        -- Total # of times clapped or twirled.
end
```

There was an *ACROBAT* (1)

```
class
        ACROBAT
create
        pair
feature
        buddy: COPYCAT
        pair (p: COPYCAT)
                -- Remember `p' being the copycat.
                do
                -- to be completed
                ensure      buddy = p
                end
```

There was an *ACROBAT* (2)

```
        clap (n: INTEGER)
                    -- Clap `n' times and forward to copycat.
            do
            -- to be completed
            ensure                      count = old count + n
            end

        twirl (n: INTEGER)
                    -- Twirl `n' times and forward to copycat.
            do
            -- to be completed
            ensure                      count = old count + n
            end

        count: INTEGER
                    -- Ask copycat and return his answer.
            do
            -- to be completed
            end

end
```

Hands-On

ACROBAT and COPYCAT share some behaviour. How can we use inheritance to avoid duplicating code?

What do we model at super-class level?

What at sub-class level?

Different choices are posible

There is an *ACROBAT* – generic version

```
class
        ACROBAT
feature
        clap (n: INTEGER)
                        -- Clap `n' times and adjust `count'.
                require n>0
                do
                -- to be completed
                ensure  count = old count + n
                end
        twirl (n: INTEGER)
                        -- Twirl `n' times and adjust `count'.
                require n>0
                do
                -- to be completed
                ensure  count = old count + n
                end
        count: INTEGER
                        -- Total # of times clapped or twirled.
end
```

There is an *ACROBAT_WITH_COPYCAT* (1)

```
class
        ACROBAT_WITH_COPYCAT
inherit
        ACROBAT
                redefine
                        twirl, clap, count
                end
create
        pair
feature
        buddy: ACROBAT
        pair (p: ACROBAT)
                        -- Remember `p' being the copycat.
                do
                -- to be completed
                ensure          buddy = p
                end
```

There is an *ACROBAT_WITH_COPYCAT* (2)

```
        clap (n: INTEGER)
                    -- Clap `n' times and forward to copycat.
              do
              -- to be completed
              end

        twirl (n: INTEGER)
                    -- Twirl `n' times and forward to copycat.
              do
              -- to be completed
              end

        count: INTEGER
                    -- Ask copycat and return his answer.
              do
              -- to be completed
              end

end
```

You are an *ACROBAT_WITH_COPYCAT* – implementation

Open EiffelStudio,

copy-paste the code,

compile and correct errors !

Cannot redefine an attribute in descendants!

- Not allowed by the language definition.
- It might break code in ancestors where a value might be assigned to the attribute
- Performance degradation: replacing a simple memory access with a function call
- It might make things slower for incremental compiler and slow down dynamic access to entities.
- Choice might change in the future (even if unlikely)

- SOLUTION
- Hide the attribute in the ancestor
- Expose a function which is inherited and redefined

There is an *ACROBAT* – new solution

```
class
        ACROBAT
feature {NONE}
        internal_count: INTEGER   -- Store value of a private counter.
feature
        count: INTEGER
                        -- Total # of times clapped or twirled.
                do
                         Result := internal_count
                end
        clap (n: INTEGER)
                        -- Clap `n' times and adjust `count'.
                require n>0
                do
                        internal_count := internal_count + n
                ensure  count = old count + n
                end
        twirl (n: INTEGER)
                        -- Twirl `n' times and adjust `count'.
                require n>0
                do
                        internal_count := internal_count + n
                ensure
                        count = old count + n
                end
```

There is an *ACROBAT_WITH_COPYCAT* – new sol. (1)

```
class
        ACROBAT_WITH_COPYCAT
inherit
        ACROBAT
                redefine
                        twirl, clap, count
                end
create
        pair
feature
        count: INTEGER
                        -- Ask copycat and return his answer.
                do
                         Result := buddy.count
                end
        buddy : ACROBAT
        pair (p: ACROBAT)
                        -- Remember `p' being the copycat.
                do
                        buddy := p
                ensure
                        buddy = p
                end
```

There is an *ACROBAT_WITH_COPYCAT* – new sol. (2)

```
        clap (n: INTEGER)
                    -- Clap `n' times and forward to copycat.
              do
                    buddy.clap(n)
              end
        twirl (n: INTEGER)
                    -- Twirl `n' times and forward to copycat.
              do
                    buddy.twirl(n)
              end
end
```

Here is the root object

```
prepare_and_play
        local
                mariuccio, luigino: ACROBAT
                mario, luigi: ACROBAT_WITH_COPYCAT
        do
                create mariuccio
                create mario.pair(mariuccio)
                create luigino
                create luigi.pair(luigino)
                print ("%N mario.clap(3)") ;  mario.clap(3)
                print ("%N luigi.clap(7)") ;  luigi.clap (7)
                print ("%N mario.twirl(2)") ;  mario.twirl (2)
                print ("%N luigi.twirl(8)") ;  luigi.twirl (8)
                print ("%N mario.count = " + mario.count.out)
                print ("%N luigi.count = " + luigi.count.out)
        end
```

There is an *ACROBAT_WITH_COPYCAT* – implementation

Open EiffelStudio,

copy-paste the code,

compile and correct errors !

There is an Author

- When the Author is asked to **Clap**, it will be given a number. It has to Clap its hands that many times, then to say “Thank You”, and then to take a bow
- When the Author is asked to **Twirl**, it will be given a number. It has to Turn completely around that many times, the to say “Thank You”, and then to take a bow
- When the Author is asked for **Count**, it has to announce how many actions it has performed. This is the sum of the numbers it has been given to date

There is an *AUTHOR*

```
class
        AUTHOR

inherit
        ACROBAT
                redefine
                                clap, twirl
                end

feature
        clap (n: INTEGER)
                                -- Clap `n' times say thanks and bow.
                do
                                Precursor(n)
                                print("%N Thanks! As an Author I bow to you.")
                end

        twirl (n: INTEGER)
                                -- Twirl `n' times say thanks and bow.
                do
                                Precursor(n)
                                print("%N Thanks! As an Author I bow to you.")
                end
end
```

Here is the root object

```
prepare_and_play
        local
                mariuccio, luigino: ACROBAT
                mario, luigi: ACROBAT_WITH_COPYCAT
                dante: AUTHOR
        do
                create mariuccio
                create mario.pair(mariuccio)
                create luigino
                create luigi.pair(luigino)
                create dante
                print ("%N mario.clap(3)") ;  mario.clap(3)
                print ("%N luigi.clap(7)") ;  luigi.clap (7)
                print ("%N mario.twirl(2)") ;  mario.twirl (2)
                print ("%N luigi.twirl(8)") ;  luigi.twirl (8)
                print ("%N dante.clap(11)") ;  dante.clap (11)
                print ("%N mario.count = " + mario.count.out)
                print ("%N luigi.count = " + luigi.count.out)
                print ("%N dante.count = " + dante.count.out)
        end
```

There is an *AUTHOR* – implementation

Open EiffelStudio,

copy-paste the code,

compile and correct errors!