Fondamenti della Programmazione: Metodi Evoluti

Prof. Enrico Nardelli

Esercitazione 3

Acrobat game

Hands-On

- We will play a little game now.
- Objects will have different roles.

There is an acrobat object

- When asked to **Clap**, it will be given a number and it has clap its hands that many times.
- When asked to **Twirl**, it will be given a number and it has to turn completely around that many times.
- When asked for **Count**, it has to announce how many actions it has performed. This is the sum of the numbers that have been given to date.

Pseudocode

«Pseudocode» means sentences in natural language, which are not yet written code

Example:

```
-- "Clap n times"
```

We write pseudocode as comments and between quotation marks

Style: when the actual code is written, it is a good idea to keep the pseudo-code in the program as a regular comment

There is an *ACROBAT* – first version

```
class
        ACROBAT

feature
        clap (n: INTEGER)
                do
                        -- "Clap `n' times and adjust `count'."
                end

        twirl (n: INTEGER)
                do
                        -- "Twirl `n' times and adjust `count'."
                end

        count: INTEGER
                        -- "Total # of times clapped or twirled."
end
```

There is an *ACROBAT* – adding contracts

```
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
```

We need a root object

- It got created by the runtime.
- It is executing the first feature.

Here is the *DIRECTOR*

```
class
        DIRECTOR

create
                prepare_and_play

feature
                prepare_and_play
                        do
                                -- See following slides.
                        end
```

Here is the root object (version 1)

```
prepare_and_play
        local
                mario, luigi, piero: ACROBAT


        do
                create mario
                create luigi
                create piero
                mario.clap (3)
                luigi.clap (4)
                piero.clap (5)
                mario.twirl (3)
                luigi.twirl (4)
                piero.twirl (5)
                mario.count
                luigi.count
                piero.count
        end
```

Allow objects to give feedback to what happens to them by printing it.
For example: print("%N mario.count = " + mario.count.out)

There is an *ACROBAT* – implementation

Open EiffelStudio,

copy-paste the code,

and complete it !

There are Acrobat and Copycat objects

- Both kind of objects will **Clap**, **Twirl**, and **Count**
- Each Acrobat object will have another object as its Copycat. (N.B. asymmetric relation!)
- When asked to **Clap** (or **Twirl**),
  - the Acrobat will be given a number. It will clap its hands (or turn completely around) that many times and pass the same instruction to its Copycat, who will just **Clap** (or **Twirl**)
  - the Copycat will be given a number and will clap its hands (or turn completely around) that many times
- When asked for **Count**,
  - the Acrobat will ask its Copycat and answer with the number it provides
  - the Copycat will provide the total number of actions is has executed

There is a *COPYCAT* – first version

```
class
        COPYCAT

feature
        clap (n: INTEGER)
                do
                        -- "Clap `n' times and adjust `count'."
                end

        twirl (n: INTEGER)
                do
                        -- "Twirl `n' times and adjust `count'."
                end

        count: INTEGER
                        -- "Total # of times clapped or twirled."
end
```

There is a *COPYCAT* – second version

```
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 is an *ACROBAT* – first version (1)

```
class
        ACROBAT


feature
        buddy: COPYCAT

        pair (p: COPYCAT)
                do
                        -- Remember `p' being the copycat.
                end
```

There is an *ACROBAT* – first version (2)

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

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

count: INTEGER
        do
                -- "Ask copycat and return his answer."
        end
end
```

This query is no more an attribute but a routine

There is an *ACROBAT* – second version (1)

To be explained later

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

There is an *ACROBAT* – second version (2)

```
    clap (n: INTEGER)
                                    -- Clap `n' times and forward to copycat.
            require                         n>0
            do
            -- to be completed
            ensure                          count = old count + n
            end

    twirl (n: INTEGER)
                                    -- Twirl `n' times and forward to copycat.
            require                         n>0
            do
            -- to be completed
            ensure                          count = old count + n
            end

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

end
```

Here is the root object

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

There are *ACROBAT* and *COPYCAT* – implementation

Open EiffelStudio,

copy-paste the code,

compile and correct errors !