Programozási nyelvek és paradigmák

Absztrakt osztályok

Kozsik Tamás (2020)

Absztrakt osztály

- Cél: származtassunk belőle
  - Ugyanannak az absztrakciónak többféle megvalósítása lehet
  - Kód redundanciájának csökkentése
- Implementálatlan viselkedést deklarál(hat)
  - Teljesen absztrakt? (interface, protocol)
  - Részleges implementációt tartalmaz? Művelettel való paraméterezés!
- Példányosítás?
  - Simula 67: implementálatlan művelet hívása futási hibát okoz!
  - Smalltalk: szintén (absztrakt osztály/metódus: csak idióma)
  - C++, Java, Eiffel...: fordítási hiba!

```
deferred class ANIMAL
...
```

Absztrakt művelet

- C++: pure virtual member function
- Java...: abstract method
- Eiffel: deferred feature

```
deferred class ANIMAL
feature
    talk: STRING
        deferred
        end
end
```

Megvalósítás (effecting, making effective)

```
deferred class ANIMAL
feature
    talk: STRING
        deferred
        end
end

class CAT
inherit ANIMAL
feature
    talk: STRING
        do
            Result := "Miaow"
        end
end
```

Effecting a deferred feature with an attribute

```
deferred class ANIMAL
feature
    talk: STRING
        deferred
        end
end

class CAT
inherit ANIMAL
feature
    talk: STRING
        attribute
            Result := "Miaow"
        end
end
```

Picit másként

```
class CAT
inherit
    ANIMAL
        redefine
            default_create
        end
feature
    talk: STRING
    default_create
        do
            talk := "Miaow"
        end
end
```

Once routine

```
deferred class ANIMAL
feature
    talk: STRING
        deferred
        end
end

class CAT
inherit ANIMAL
feature
    talk: STRING
        once
            Result := "Miaow"
        end
end
```

Másik példa

```
deferred class COMPLEX
...
feature
    re, im: REAL deferred end
    r, arg: REAL deferred end
...
invariant
    arg < 2*Pi
    arg >= 0
end -- class COMPLEX
```

Másik példa

```
deferred class COMPLEX
...
feature
    re, im: REAL deferred end
    r, arg: REAL deferred end
...
invariant
    arg < 2*Pi
    arg >= 0
end -- class COMPLEX

class POLAR_COMPLEX
inherit COMPLEX
feature
    r, arg: REAL
...
end -- class POLAR_COMPLEX
```

Öröklődés, mint másik modul használata

```
deferred class COMPLEX
inherit MATH
feature
    re, im: REAL deferred end
    r, arg: REAL deferred end
...
invariant
    arg < 2*Pi
    arg >= 0
end -- class COMPLEX


class MATH
feature
    Pi: REAL = 3.14159265358979323846264338327950 29
...
end -- class MATH
```

A MATH modul: állapotmentes osztály

```
class MATH
feature
    Pi: REAL = 3.14159265358979323846264338327950 29
    log_2 (v: REAL): REAL
        -- Base 2 logarithm of `v'
        do
            Result := log (v) / log (2.0)
        end
    cosine (v: REAL): REAL ...
    arc_cosine (v: REAL): REAL ...
    sqrt (v: REAL): REAL ...
    exp (x: REAL): REAL ...
    ...
end -- class MATH
```

FFI – más nyelven írt implementáció

```
class MATH
feature
    ...

    cosine (v: REAL): REAL
        -- Trigonometric cosine of radian `v'
        -- approximated in the range [-pi/4, +pi/4]
        external
            "C signature (double): double use <math.h>"
        alias
            "cos"
        end

    ...

end -- class MATH
```

Korábban: built-in

```
class ANY
feature

    ...

    conforms_to (other: ANY): BOOLEAN
        -- Does type of current object conform to type
        -- of `other' (as per Eiffel: The Language)?
        require
            other_not_void: other /= Void
        external
            "built_in"
        end
...

end
```

Részleges implementáció

```
deferred class COMPLEX
...
feature  -- creation and setting
    from_cart( re_, im_: REAL ) deferred end
    from_polar( r_, arg_: REAL ) deferred end
feature  -- conversion between representations
    polar: POLAR_COMPLEX
        do
            create Result.from_polar(r,arg)
        end
    cart: CART_COMPLEX
        do
            create Result.from_cart(re,im)
        end
...
end -- class COMPLEX
```

Megkérdőjelezhető tervezés

- Bázisosztály függ az alosztályaitól
- Moduláris programozás: nincs körkörös függőség

```
deferred class COMPLEX
...
feature  -- conversion between representations
    polar: POLAR_COMPLEX
        do
            create Result.from_polar(r,arg)
        end
    cart: CART_COMPLEX
        do
            create Result.from_cart(re,im)
        end
...
end -- class COMPLEX
```

Deferred feature szerződése

```
deferred class COMPLEX
...
feature  -- arithmetics
    times alias "*" ( other: COMPLEX ): COMPLEX
        deferred
        end

    divided_by alias "/" ( other: COMPLEX ): COMPLEX
        require
            nonzero_divisor: other.r /= 0.0
        deferred
        ensure
            inverse_of_times: Current * Result ~ other
        end

...
end -- class COMPLEX
```

Deferred feature szerződése öröklődik

```
class POLAR_COMPLEX
inherit COMPLEX
...
feature  -- arithmetics

    divided_by alias "/" ( other: COMPLEX ): COMPLEX
        -- require else False
        do
            create {POLAR_COMPLEX} Result.from_polar(
                                r / other.r, arg - other.arg)
        -- ensure then True
        end

...
end -- class POLAR_COMPLEX
```

Segédműveletek

```
deferred class COMPLEX
...
feature {NONE}  -- conversion helpers
    r_from_cart( re_, im_: REAL ) : REAL do ... end
    arg_from_cart( re_, im_: REAL ) : REAL do ... end
    re_from_polar( r_, arg_: REAL ) : REAL
        do
            Result := r_ * cosine(arg_)
        end -- re_from_polar
    im_from_polar( r_, arg_: REAL ) : REAL
        do
            Result := r_ * sine(arg_)
        end -- im_from_polar
...
end -- class COMPLEX
```

Segédműveletek

```
arg_from_cart( re_, im_: REAL ) : REAL
    do
        if re_ = 0 then
            if im_ >= 0 then
                Result := Pi / 2
            else
                Result := 3*Pi/2
            end
        else
            Result := arc_tangent(im_/re_)
            if re_ < 0 then
                Result := Result + Pi
            elseif im_ < 0 then
                Result := 2*Pi + Result
            end
        end
    end -- arg_from_cart in COMPLEX
```

Befagyasztott művelet: nem definiálható felül

```
deferred class COMPLEX
...
feature {NONE}  -- conversion helpers
    frozen r_from_cart( re_, im_: REAL ) : REAL ...
    frozen arg_from_cart( re_, im_: REAL ) : REAL ...
    frozen re_from_polar( r_, arg_: REAL ) : REAL
        do
            Result := r_ * cosine(arg_)
        end -- re_from_polar
    frozen im_from_polar( r_, arg_: REAL ) : REAL
        do
            Result := r_ * sine(arg_)
        end -- im_from_polar
...
end -- class COMPLEX
```

Befagyasztott osztály: nem származtathatunk belőle

```
frozen class POLAR_COMPLEX
inherit
    COMPLEX
...
end -- class POLAR_COMPLEX
```

Megvalósítás – reprezentáció

```
class POLAR_COMPLEX
inherit COMPLEX
create from_polar, from_cart, default_create
feature
    r: REAL assign set_r
    arg: REAL assign set_arg
    re: REAL
        do
            Result := re_from_polar(r,arg)
        end -- re
    im: REAL
        do
            Result := im_from_polar(r,arg)
        end -- arg
...
end -- class POLAR_COMPLEX
```

Megvalósítás – beállítás

```
from_polar( r_, arg_: REAL )
    do
        set_r(r_)
        set_arg(arg_)
    end -- from_polar
from_cart( re_, im_: REAL )
    do
        r := r_from_cart(re_,im_)
        arg := arg_from_cart(re_,im_)
    end -- from_cart
set_r( r_: REAL )
    do
        r := r_
    end -- set_r
set_arg( arg_: REAL ) do ... end
```

Megvalósítás – beállítás és az invariáns

```
set_arg( arg_: REAL )
    do
        from
            arg := arg_.abs
        invariant
            arg >= 0
        until
            arg < 2*Pi
        loop
            arg := arg - 2*Pi
        variant
            arg.truncated_to_integer
        end
        if arg_ < 0 then
            arg := 2*Pi - arg
        end
    end -- set_arg
```

Megvalósítás – aritmetika

```
class POLAR_COMPLEX
inherit COMPLEX
...
feature  -- arithmetics
    times alias "*" ( other: COMPLEX ): COMPLEX
        do
            create {POLAR_COMPLEX} Result.from_polar(
                                r * other.r, arg + other.arg)
        end
...
end -- class POLAR_COMPLEX
```

Megvalósítás – aritmetika

```
class POLAR_COMPLEX
inherit COMPLEX
...
feature  -- arithmetics
    times alias "*" ( other: COMPLEX ): COMPLEX
        do
            create {POLAR_COMPLEX} Result.from_polar(
                        r * other.r, arg + other.arg)
        end
...
end -- class POLAR_COMPLEX
```

```
cc: CART_COMPLEX          c  := cc * cp  -- ok
pc: POLAR_COMPLEX         c  := c  * c   -- ok
c : COMPLEX               cc := cc * cp  -- fordítási hiba
```

Különböző implementációk: kevesebb számolás kell

```
class COMPLEX
feature times alias "*"( other: COMPLEX ): COMPLEX ...
...

class POLAR_COMPLEX inherit COMPLEX
feature times alias "*" ( other: COMPLEX ): COMPLEX
    do create {POLAR_COMPLEX} Result.from_polar(
                    r * other.r, arg + other.arg )
    end
...

class CART_COMPLEX inherit COMPLEX
feature times alias "*"( other: COMPLEX ): COMPLEX
    do create {CART_COMPLEX} Result.from_cart(
                     re * other.re - im * other.im,
                     re * other.im + im * other.re )
    end
...
```

Újradeklarálás: kovariáns visszatéréssel (covariant return)

```
class COMPLEX
feature times alias "*"( other: COMPLEX ): COMPLEX ...
...

class POLAR_COMPLEX inherit COMPLEX
feature times alias "*" ( other: COMPLEX ): POLAR_COMPLEX
    do create Result.from_polar(
                    r * other.r, arg + other.arg)
    end
...

class CART_COMPLEX inherit COMPLEX
feature times alias "*"( other: COMPLEX ): CART_COMPLEX
    do create Result.from_cart(
                     re * other.re - im * other.im,
                     re * other.im + im * other.re )
    end
...
```

A pontosabb típus kényelmesebbé teszi a használatot!

```
class COMPLEX
feature times alias "*"( other: COMPLEX ): COMPLEX ...
...

class POLAR_COMPLEX inherit COMPLEX
feature times alias "*" ( other: COMPLEX ): POLAR_COMPLEX
...

class CART_COMPLEX inherit COMPLEX
feature times alias "*"( other: COMPLEX ): CART_COMPLEX
...
```

```
cc: CART_COMPLEX          c  := cc * cp  -- ok
pc: POLAR_COMPLEX         c  := c  * c   -- ok
c : COMPLEX               cc := cc * cp  -- ok
```