Programozási nyelvek és paradigmák

Többszörös öröklődés

Kozsik Tamás (2020)

Többszörös öröklődés

- egyszeres öröklődés (Simula, Smalltalk)
- több szülőinterfész, egy bázisosztály (Java, C#, Ada)
- több szülőosztály (CLOS, C++)

Problémákat vethet fel (névütközés, diamond)

- az Eiffel támogatja a több szülőosztályt
- gazdagabb lehetőségeket ad, mint a C++
- tud publikus és privát öröklődést is

Többszörös kódöröklés

- Max egy konform, többi privát
- Több is konform, pl. független funkciók
- Ismételt öröklés (repeated inheritance), pl. `ANY`

Független szülőosztályokból kódöröklés

```
class CELL [G]        -- standard library
create put
feature
    item: G
    put, replace (v: like item)
        do item := v ensure item_inserted: item = v end
end

class LINKED
feature
    next: detachable like Current assign set
    set( v: detachable like Current )
        do next := v ensure next = v end
end

class LIST [T] inherit {NONE} CELL[T] LINKED end
```

Független funkciók összeöröklése

```
deferred class NUMERIC
feature
    opposite alias "-": like Current deferred end
    plus alias "+"( other: like Current ): like Current
        deferred end
    minus alias "-"( other: like Current ): like Current
        do Result := Current + (-other) end
...

deferred class COMPARABLE
feature
    is_less alias "<"( other: like Current ): BOOLEAN
        deferred end
    is_less_equal alias "<="( other: like Current ): BOOLEAN
        do Result := not other < Current end
...

class FRACTION inherit NUMERIC COMPARABLE HASHABLE ...
```

Ütközések feloldása név alapján

- minden feature neve különböző kell legyen egy adott osztályban
- örökölt feature-ök átnevezhetők

Fordítási hiba: két feature ugyanazzal a névvel

```
class STATEMENT
feature
    is_right: BOOLEAN
end

class PARTY
feature
    is_right: BOOLEAN
end

class CAMPAIGN_PROMISE
inherit STATEMENT
inherit PARTY
end
```

Öröklődés során átnevezés

```
class STATEMENT
feature
    is_right: BOOLEAN
end

class PARTY
feature
    is_right: BOOLEAN
end

class CAMPAIGN_PROMISE
inherit STATEMENT rename is_right as holds end
inherit PARTY
end
```

Dinamikus kötés

```
class CAMPAIGN_PROMISE
inherit
    STATEMENT
        rename is_right as holds
        redefine holds
    end
    PARTY    -- is_right inherited as is
...
end


local
    s: STATEMENT
do
    create {CAMPAIGN_PROMISE} s
    if s.is_right then ...
```

Ugyanazt a feature-t ugyanazzal az implementációval több irányból

```
class STATEMENT                 -- inherits is_equal from ANY
feature
    is_right: BOOLEAN
end

class PARTY                     -- inherits is_equal from ANY
feature
    is_right: BOOLEAN
end

class CAMPAIGN_PROMISE          -- inherits is_equal from ANY
inherit STATEMENT rename is_right as holds end
inherit PARTY
end
```

Ugyanazt a feature-t több irányból különböző implementációval: fordítási hiba

```
deferred class NUMERIC
    -- is_equal inherited from ANY
...

deferred class COMPARABLE
feature
    is_less alias "<"( other: like Current ): BOOLEAN
        deferred end
    is_equal ( other: like Current ): BOOLEAN
        do
            Result := not (Current < other) and
                      not (other < Current)
        end
...

class FRACTION inherit NUMERIC COMPARABLE end
```

Legfeljebb egy implementációt örökölhetünk egy feature-höz

```
deferred class NUMERIC
    -- is_equal inherited from ANY
...

deferred class COMPARABLE
feature
    is_less alias "<"( other: like Current ): BOOLEAN
        deferred end
    is_equal ( other: like Current ): BOOLEAN
        do
            Result := not (Current < other) and
                      not (other < Current)
        end
...

class FRACTION inherit NUMERIC undefine is_equal end
               inherit COMPARABLE
end
```

Megörökölhetjük mindkét implementációt különböző feature-ként

```
deferred class NUMERIC
    -- is_equal inherited from ANY
...

deferred class COMPARABLE
feature
    is_less alias "<"( other: like Current ): BOOLEAN
        deferred end
    is_equal ( other: like Current ): BOOLEAN
        do
            Result := not (Current < other) and
                      not (other < Current)
        end
...

class FRACTION inherit NUMERIC rename is_equal as eq end
               inherit COMPARABLE
end     -- még nem tökéletes...
```

Dinamikus kötés

```
class FRACTION inherit NUMERIC rename is_equal as eq end
               inherit COMPARABLE select is_equal end
end



local
    a: ANY
    n: NUMERIC
    c: COMPARABLE
    f: FRACTION
do
    create f; a := f; n := f; c := f
    f.is_equal( f )
    c.is_equal( c )
    n.is_equal( n )
    a.is_equal( a )     -- ehhez kell a select
```

Diamond esetben

- Ha ugyanazon a néven örököljük több irányból ugyanazt, csak egyet kapunk belőle (“join”)
- Ha különböző néven, akkor többet
- Ha több implementációját örököljük, akkor nem lehet join-olni
- De öröklődésnél törölhetjük az implementációt: `undefine`
- Különböző feature-öket nem lehet join-olni

Diamond példa

```
class A
inherit ANY redefine default_create end
feature default_create do end
end

class B
inherit ANY redefine default_create end
feature default_create do end
end

class C
inherit          A undefine default_create end
inherit {NONE} B rename default_rescue as b_rescue end
end
```

Dinamikus kötés?

Ugyanazt a feature-t több irányból megörökölhetjük különböző néven akár (de nem feltétlenül) különböző implementációkkal

```
class A
inherit ANY redefine default_create end
feature default_create do end
end

class B
inherit ANY redefine default_create end
feature default_create do end
end

class C
inherit A undefine default_create select default_rescue end
        B rename default_rescue as b_rescue end
end
```

Ismételt öröklés

```
class BINTREE[T]
inherit CELL[T]  -- provides item and put
inherit {NONE}
    LINKED
        rename
            next as left,
            set as set_left
        end
    LINKED
        rename
            next as right,
            set as set_right
        end

create put
end
```