Smalltalk 3.

Osztályok létrehozása

Készítette: Szabó Éva

Metaosztály fogalma

Mint korában említettük, a Smalltalkban mindent objektumnak tekintünk. Még az osztályok is objektumok. De ha az osztály objektum, akkor az is - mint minden más objektum – valamilyen osztályhoz kell tartozzon. Másképp fogalmazva minden osztály (pontosan) egy másik osztály példánya. Ezen "másik" osztályt metaosztálynak hívjuk.

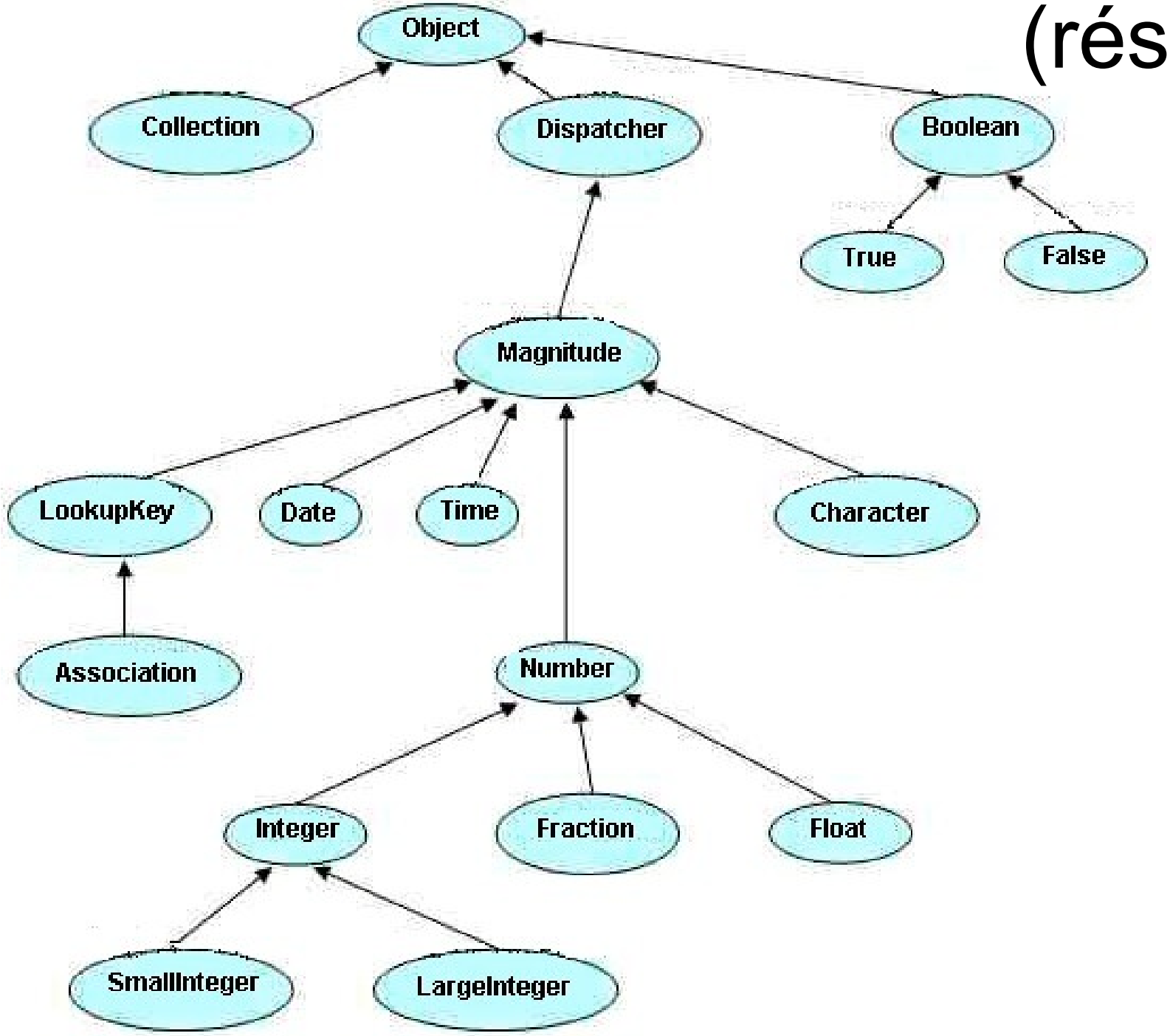
Az *Object* osztály

Az *Object* osztály minden osztály közös őse, tehát minden objektum az *Object* osztály egy példánya. Ezért minden, az *Object* osztálynak rendelkezésre álló művelettel minden más objektum is rendelkezik.

Ezek közül néhány fontosabb:

- ***class*** (unáris üzenet): Visszatérési értékként megkapjuk, hogy a címzett objetum milyen osztályhoz tartozik.
  Pl: *'Hello' class ! → String*
- ***isMemberOf*** (kulcsszavas üzenet): Visszatérési értéke egy logikai érték. Az üzenet argumentuma egy osztálynév. Ha a címzett objektum példánya ezen osztálynak, akkor "*true*" a visszatérési érték, egyébként "*false*".
  Pl: *'Hello' isMemberOf: String ! → true*

A Smalltalk osztályhierarchiája (részlet)



Új osztály definiálása

```
Ososztaly subclass: Nev [
    |peldanyadattag_1 ... peldanyadattag_n|
    peldanyuzenet_1 [
        uzenet_torzs
    ]
    peldanyuzenet_2 [
        uzenet_torzs
    ]
    ...
    peldanyuzenet_n [
        uzenet_torzs
    ]
    osztalyadattag_1 := ertek.
    osztalyadattag_2 := ertek.
    ...
    osztalyadattag_n := ertek.
]
```

```
Object subclass: #Hello
    instanceVariableNames: ' '
    classVariableNames: ' '
    poolDictionaries: ' '
    category: nil!

    ----------------------------

Object subclass: #Kor
     instanceVariableNames:
'koord_x koord_y sugar'
     classVariableNames:
'KorokSzama'
     poolDictionaries: ' '
     category: nil!
```

Mit jelent a példány- és osztályváltozó?

Tekintsünk úgy egy osztályra, mintha egy C-beli *struct* lenne. Ha C-ben létrehozunk egy „struct-típusú” változót, akkor a struktúra adattagjaival minden ebből a típusból deklarált változó rendelkezik. Ilyen adattagoknak képzeljük el az osztályon belüli példányváltozót. Objektumorientáltan fogalmazva: minden, egy adott osztályból példányosított objektum rendelkezik az osztály összes példányváltozójával. Természetesen ezek példányonként más és más értékeket vehetnek fel.

Az osztályváltozó inkább egy C-beli *static* kulcsszóval ellátott globális változóhoz hasonlítható. Egy osztályból példányosított objektumok hozzáférnek az osztály osztályváltozóihoz, de egyik példány sem rendelkezik ilyen „adattaggal” (példányváltozóval).

Üzenet definiálása osztályhoz

- Osztály definiálásakor

```
Ososztaly subclass: Nev [
    ...
"unáris"
    nev [
        utasitasok
    ]
"bináris - op lehet pl. +, -, *, stb."
    op param [
        utasitasok
    ]
"kulcsszavas"
    kulcsszo_1: param_1 kulcsszo_2: param_2 ... kulcsszo_n: param_n [
        utasitasok
    ]
    ...
]
```

- Utólag: (beépített osztályokra is működik)

```
Osztaly extend [

   "ugyanúgy, mint az 1. esetnél; lehet unáris, bináris és kulcsszavas is"
    uzenet_1 [
        utasitasok
    ]

    uzenet_2 [
        utasitasok
    ]
    ...
]
```

- Osztályüzenetek létrehozása:
  (csak utólag lehetséges)

```
Osztaly class extend [

    uzenet_1 [
        utasitasok
    ]

    uzenet_2 [
        utasitasok
    ]

    ...
]
```

```
!<osztálynév> <class> methodsFor:
'megjegyzés'!
<metódus neve>
        <utasítás_1, utasítás_2, utasítás_n>
!!
```

pl.:

```
! Kor methodsFor: 'Inicializálás' !
init
        koord_x := 0.
        koord_y := 0.
        sugar := 0.
!!
```

Konstruktor

```
Osztaly class extend [ "osztályüzenetet kell csinálni"

    "paraméter nélküli konstruktor"
    new [ "akármi lehet a neve, de a beépített osztályoknál mindig new, célszerű ezt követni"
        |a|
        a := super new.
        a init. "természetesen paraméteres initet is lehetne hívni"
        ^a.
    ]

    "paraméteres konstruktor"
    new: a new: b ... new: n [
        |a|
        a := super new.
        a init: a init: b ... init: n.
        ^a.
    ]
    ...
]
```

*! Kor **class** methodsFor: 'Létrehozás. A konstruktor szerepét tölti be.' !*

```
new
        |a| a:= super new.
        a init.
        ^a
!!
```

*”a **class** kulcsszó azért kell, mert az üzenetet nem a Kor objektumnak, hanem a Kor osztálynak küldjük közvetlenül*
*A **super** a szülőosztálynak küldi az üzenetet”*

"Létre kell hozni továbbá az inicializáló üzeneteket példányüzenetként:"

```
Osztaly extend [

    "paraméter nélküli init"
    init [
        peldanyadattag_1 := nil.
        peldanyadattag_2 := nil.
        ...
        peldanyadattag_n := nil.
        ^self. "visszaadja önmagát"
    ]

    "paraméteres init"
    init: a init: b ... init: n [
        peldanyadattag_1 := a.
        peldanyadattag_2 := b.
        ...
        peldanyadattag_n := n.
        ^self. "visszaadja önmagát"
    ]
    ...
]
```

A super és a self

A ***super*** kulcsszó a Parent::, a ***self*** pedig a this, C++-ból ismert kulcsszavak megfelelői. Tehát egy metódusban levő self mindig arra az objektumra hivatkozik, melyre azt meghívtuk, a super pedig a szülőosztályra.

Feladatok:

1. Adjunk az **Integer** osztályhoz egy **summTo:** üzenetet, amelynek eredménye A summTo: B esetén a számok összege A-tól B-ig.

2. Írjunk egy **Tört** osztályt, és valósítsuk meg a négy alapműveletet!
Ehhez definiáljuk a következő üzeneteket:

- **szamlalo**: visszaadja a tört számlálóját,
- **nevezo**: visszaadja a tört nevezőjét,
- **egyszerusit**: egyszerűsíti a törtet,
- **printNl**: kiírja a törtet számláló/nevező formájában,
- **+** összeadó bináris operátor,
- **-** kivonó bináris operátor,
- * szorzó bináris operátor,
- */* osztó bináris operátor.

9. Adjunk az Integer osztályhoz egy summTo: üzenetet, amelynek eredménye A summTo: B esetén a számok összege A-tól B-ig.

```
Integer extend [
summTo: B
[   self < B ifTrue:
  [
     |sum| sum:=0.
     self to: B do: [:i | i printNl.].
     ^sum.
  ]
ifFalse:
  [
      'A megadott ertek kisebb' printNl.
  ^self.
  ].
]
]


5 summTo: 4.
4 summTo: 13.
```

```
!Integer methodsFor: 'summTo muvelet'!
summTo: B
        self < B ifTrue:
          [ self to: B do: [:i | i printNl.] ]
        ifFalse:
          ['A megadott ertek kisebb' printNl.]!
!!

5 summTo: 4.
4 summTo: 13.
```

10. Írjunk tört osztályt, és valósítsuk meg a négy alapműveletet! Ehhez definiáljuk a következő üzeneteket:

- szamlalo: visszaadja a tört számlálójá,
- nevezo: visszaadja a tört nevezőjét,
- egyszerusit: egyszerűsíti a törtet,
- printNl: kiírja a törtet számláló/nevező formájában,
- + összeadő bináris operátor,
- - kivonó bináris operátor,
- * szorzó bináris operátor,
- / osztó bináris operátor.

```
Object subclass: Tort [
    |szamlalo nevezo|

    init: sz init: n [
        szamlalo := sz.
        nevezo := n.
        self egyszerusit.
        ^self.
    ]

    szamlalo [
        ^szamlalo.
    ]

    nevezo [
        ^nevezo.
    ]

    egyszerusit [
        |lnko| lnko := szamlalo gcd: nevezo.
        szamlalo := szamlalo // lnko.
        nevezo := nevezo // lnko.
    ]
```

```
Number subclass: #Tort
        instanceVariableNames: 'szamlalo nevezo'
        classVariableNames: ''
        poolDictionaries: ''
        category: nil!

! Tort class methodsFor: 'konstruktor' !
        new: szamlalo new: nevezo
                |a| a := super new.
                a init: szamlalo init: nevezo.
                ^a
!!

! Tort methodsFor: 'egyeb uzenetek' !
        init: a init: b
                szamlalo := a.
                nevezo := b.
!
        szamlalo
                ^szamlalo
!
        nevezo
                ^nevezo
!
        egyszerusit
                |lnko| lnko := szamlalo gcd: nevezo.
                szamlalo := szamlalo // lnko.
                nevezo := nevezo // lnko.
!
```

```
  printNl [
      szamlalo display. $/ display. nevezo displayNl.
  ]

  + masik [
      |er sz n|
      sz := (szamlalo * masik nevezo) + (nevezo * masik szamlalo).
      n := nevezo * masik nevezo.
      er := Tort new: sz new: n.
      er egyszerusit.
      ^er.
  ]

  - masik [
      |er sz n|
      sz := (szamlalo * masik nevezo) - (nevezo * masik szamlalo).
      n := nevezo * masik nevezo.
      er := Tort new: sz new: n.
      er egyszerusit.
      ^er.
  ]
```

```
      printNl
            szamlalo display.
            $/ display.
            nevezo display.
            Character nl display.
!
      + masik
            |er sz n|
            sz := (szamlalo * masik nevezo) + (nevezo * masik szamlalo).
            n := nevezo * masik nevezo.
            er := Tort new: sz new: n.
            er egyszerusit.
            ^er
!
      - masik
            |er sz n|
            sz := (szamlalo * masik nevezo) - (nevezo * masik szamlalo).
            n := nevezo * masik nevezo.
            er := Tort new: sz new: n.
            er egyszerusit.
            ^er
!
```

```
    * masik [
        |er sz n|
        sz := szamlalo * masik szamlalo.
        n := nevezo * masik nevezo.
        er := Tort new: sz new: n.
        er egyszerusit.
        ^er.
    ]

    / masik [
        |er sz n|
        sz := szamlalo * masik nevezo.
        n := nevezo * masik szamlalo.
        er := Tort new: sz new: n.
        er egyszerusit.
        ^er.
    ]
]
```

```
!
        * masik
                |er sz n|
                sz := szamlalo * masik szamlalo.
                n := nevezo * masik nevezo.
                er := Tort new: sz new: n.
                er egyszerusit.
                ^er
!
        / masik
                |er sz n|
                sz := szamlalo * masik nevezo.
                n := nevezo * masik szamlalo.
                er := Tort new: sz new: n.
                er egyszerusit.
                ^er
!!
```

```
Tort class extend [
    new: sz new: n [
        |obj| obj := super new.
        obj init: sz init: n.
        ^obj.
    ]
]

|r1 r2|

r1 := Tort new: 2 new: 3.
r2 := Tort new: 3 new: 4.

(r1 + r2) printNl.
(r1 - r2) printNl.
(r1 * r2) printNl.
(r1 / r2) printNl.
!
```

```
Smalltalk at: #r1 put: (Tort new: 2 new: 3).
Smalltalk at: #r2 put: (Tort new: 3 new: 4).
(r1+r2) printNl!
(r1-r2) printNl!
(r1*r2) printNl!
(r1/r2) printNl!
```