Algunas clases de Smalltalk

Object

Magnitude

Class Object

- Todas las clases heredan de Object
  - Object es la superclase de todas las clases,
  - Todas las clases heredan sus métodos
- Su protocolo:
  - #printString: convierte al receptor en un string
    - Es un "template method" que usa el método #printOn:

```
Transcript show: (5 * 3) printString
```

  - Comparación por igualdad y equivalencia (identidad): = y ==

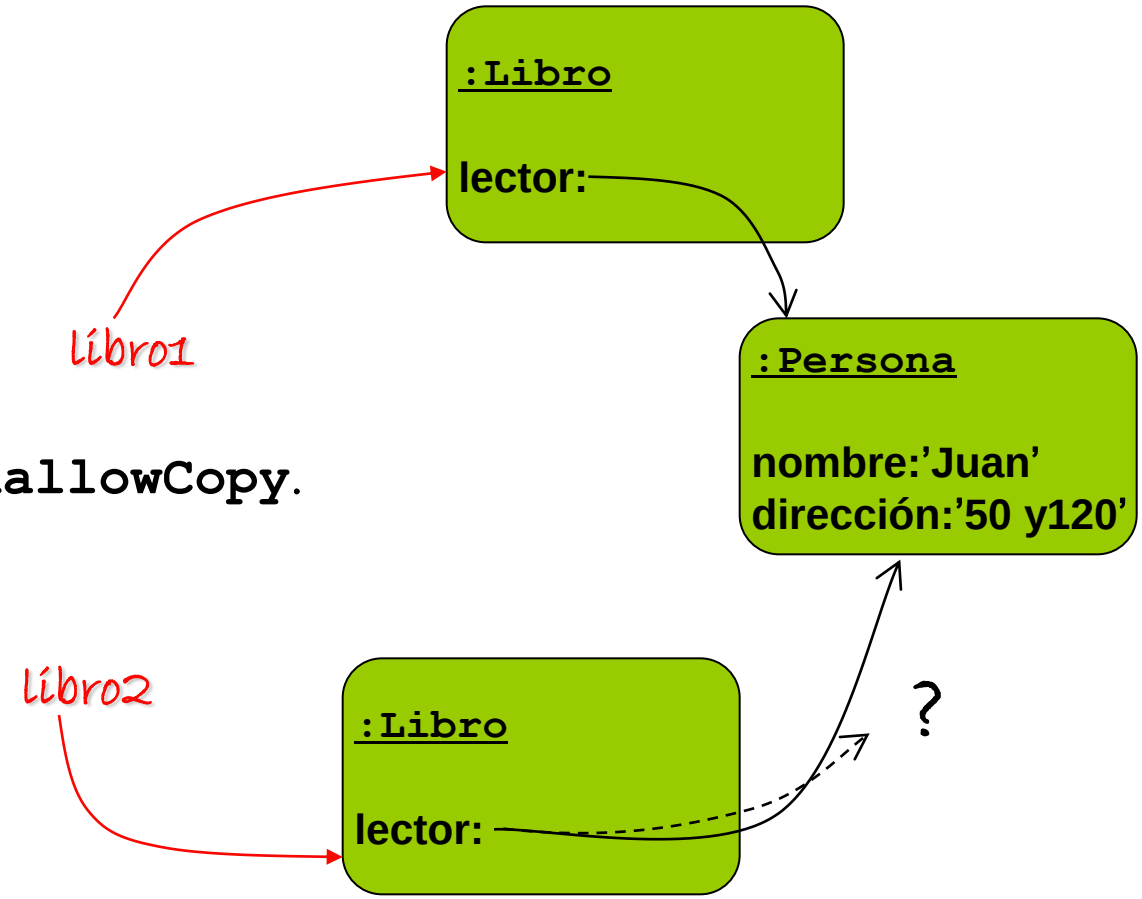
```
| p1 p2 |
p1 := (Point x: 10 y: 20).
p2 := (Point x: 10 y: 20).
Transcript clear; show: (p1 = p2) printString.
Transcript cr; show: (p1 == p2) printString.
```

Class Object (2)

- Copiando Objetos:
  - Hace una copia del receptor y crea un nuevo objeto de la misma clase del receptor: #shallowCopy

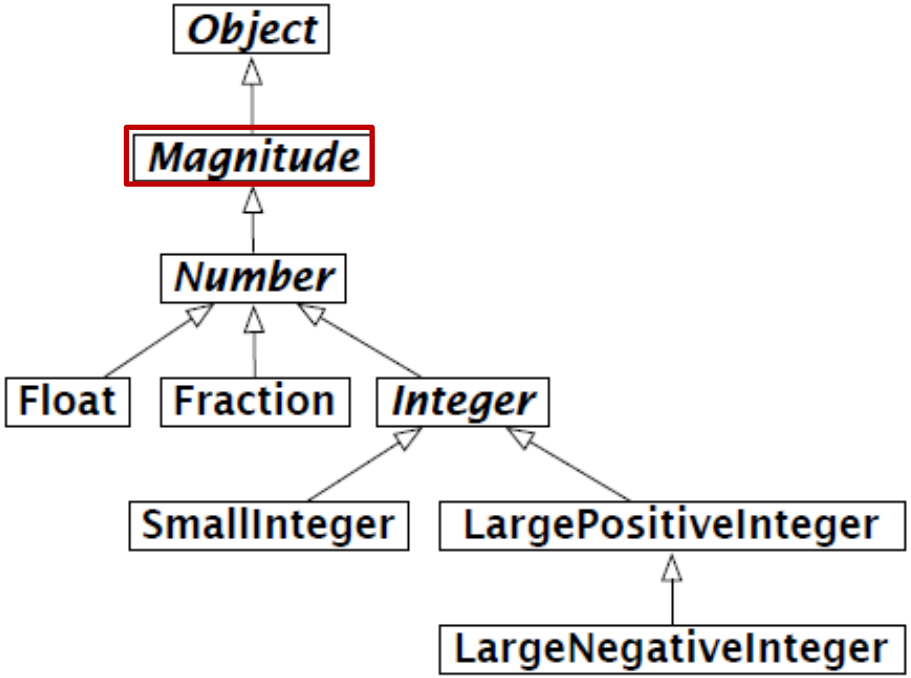
```
|libro1|
.....
```

```
|libro2|
libro2:= libro1 shallowCopy.
```



Magnitude

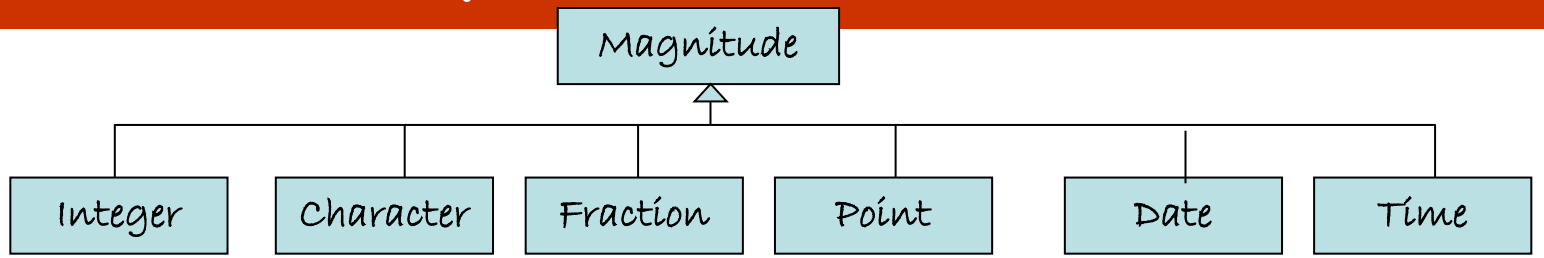
- Es la raíz abstracta de las clases que soportan operadores de comparación




```
3 + 7 / 3
(3 + 7 / 3) asFloat
(3+7/3) asFixedPoint: 2
Float pi asRational
15 log
0.3 sin
1000 factorial
37 raisedTo: 22
```

- también es superclase de **Date**, **Time**, **Character**, entre otras

#between:and: y max:




```
25 between:3 and:38
$g between:$h and:$m
Date today between:yourBirdays  and:myBirthday
```

```
Magnitude>>between: min and: max
        "Answer whether the receiver is less than or equal to the
         argument, max, and greater than or equal to the argument, min."

        ^self >= min and: [self <= max]
```

```
Magnitude>> max: aMagnitude
    "Answer the receiver or the argument, whichever has the greater magnitude"

        self > aMagnitude ifTrue:[^self] ifFalse:[^aMagnitude]
```

Clases String, Character,

- String
  - Colección indexada de caracteres
  - `'abc' < 'xyz'`
  - `'abcdefg' findString: 'de' startingAt: 1`
  - `'abcdefg' size`

- Character
  - `$3`
  - `$a`
  - `80 asCharacter  "ASCII code"`
  - `$a<$d`

Creación e inicialización de objetos

Métodos de instancia y métodos de clase

En ST hay 2 tipos

- Métodos de instancias
  - Son los que se pueden enviar a las instancias de una clase
    - `'abc' asUppercase`
    - `3/5 numerator`
    - `miCuentaBancaria saldo`

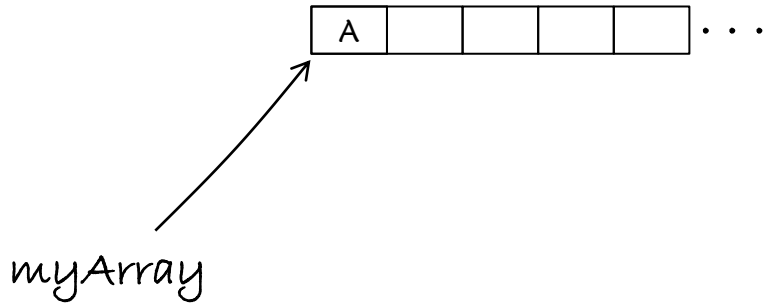
Creación

- Se hace a través de enviarle un mensaje a una clase,
- este mensaje "generalmente" es el **#new**
  - El **#new** es un mensaje que entiende cualquier clase Smallatlk
    - Ya esta definido

ClassName new

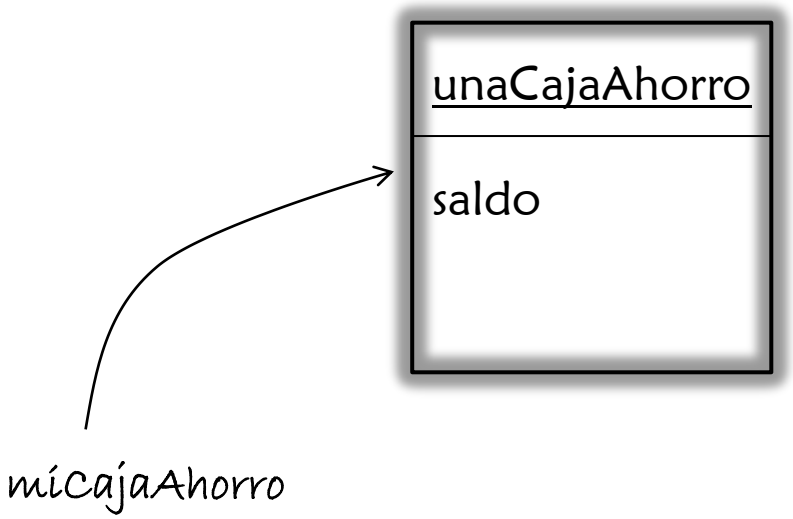
una instancia de la clase **ClassName**

```
|myArray|
myArray:= Array new.   "crea unArray"
myarray at:1 put: 'A'.
```



Creación: otro ejemplo

```
|miCajaAhorro|
miCajaAhorro := CajaAhorro new. "Crea unaCajaAhorro"
miCajaAhorro depositar: 100
```



Atención!!!
Smalltalk no maneja valores por defecto para sus variables

Inicialización: 1º opción

- Cuando creamos **miCajaAhorro** su **saldo** debe ser 0.

```
|miCajaAhorro|
miCajaAhorro := CajaAhorro new saldo:20.
```

```
CajaAhorro>>saldo:unMonto
     "Setea el saldo del receptor en unMonto"

     saldo:= unMonto.
```

Inicialización: 2º opción

- Implementando el método **#initialize**

```
|miCajaAhorro|
miCajaAhorro:= CajaAhorro new initialize
miCajaAhorro depositar: 100
```

```
CajaAhorro>>initialize
    "Setea el saldo del receptor en 0"

    saldo:= 0.
```

sirve para inicializar valores por defecto

Inicialización: 3º opción

- Cuando creamos **miCajaAhorro** esta debe ser creada con saldo 0.

```
|miCajaAhorro|
miCajaAhorro := CajaAhorro new:0
```

```
CajaAhorro class>>new:unMonto
        "Crea unaCajaAhorro y setea su saldo en unMonto."
         ^self new saldo: unMonto.
```

```
CajaAhorro>>saldo:unMonto
        "Setea el saldo del receptor en unMonto."

         saldo:= unMonto
```

Creación e inicualización

- Otro Ejemplo: Clase **`Rectangle`**
  - Tiene como vi: **`origin, corner`** ... que son puntos

1º opción:

```
|oneRectangle|
oneRectangle := Rectangle new origin: (Point x:1 y:1)
                             corner: (Point x:10 y:10).
```

2º opción:

```
|otherRectangle|
otherRectangle:=Rectangle origin: (Point x:5 y:5)
                          corner: (Point x:10 y:10).
```

Inicialización

- Implementación del método de clase **#origin:corner** y el método de instancia **#origin:corner**

```
Rectangle class>>origin:originPoint corner:cornerPoint
	"Answer an instance of the receiver whose corners
	(top left and bottom right) are determined by the
	arguments."

	^self new origin: originPoint corner: cornerPoint
```

```
Rectangle>>origin:originPoint corner:cornerPoint
	origin: originPoint corner: cornerPoint
	"Set the points at the top left corner and the
	bottom right corner of the receiver."

	origin := originPoint.
	corner := cornerPoint.
```