PROLOG

Predicados

- - member/2
  Descripcion: Verifica si un elemento pertenece a una lista
  Ejemplo: member(X, [1,2,3]). → X = 1 ; X = 2 ; ...
- - append/3
  Descripcion: Concatena listas o las divide
  Ejemplo: append([1,2], [3,4], L). → L = [1,2,3,4]
- - length/2
  Descripcion: Calcula la longitud de una lista
  Ejemplo: length([a,b,c], N). → N = 3
- - sort/2
  Descripcion: Ordena lista sin duplicados
  Ejemplo: sort([3,1,2,1], L). → L = [1,2,3]
- - msort/2
  Descripcion: Ordena lista conservando duplicados
  Ejemplo: msort([3,1,2,1], L). → L = [1,1,2,3]
- - nth1/3
  Descripcion: Elemento en índice 1-based
  Ejemplo: nth1(2, [a,b,c], X). → X = b

- - nth0/3
  Descripcion: Elemento en índice 0-based
  Ejemplo: nth0(0, [a,b,c], X). → X = a
- - last/2
  Descripcion: Último elemento de una lista
  Ejemplo: last([1,2,3], X). → X = 3
- - between/3
  Descripcion: Generador de enteros en rango
  Ejemplo: between(1, 3, X). → X = 1 ; X = 2 ; X = 3
- - is_list/1
  Descripcion: Verifica si un término es lista
  Ejemplo: is_list([a,b]). → true
- - list_to_set/2
  Descripcion: Convierte lista a conjunto
  Ejemplo: list_to_set([a,b,a], L). → L = [a,b]
- - is_set/1
  Descripcion: Verifica si no hay duplicados
  Ejemplo: is_set([a,b]). → true
- - union/3
  Descripcion: Unión de conjuntos
  Ejemplo: union([1,2], [2,3], X). → X = [1,2,3]

- - intersection/3
  Descripcion: Intersección de conjuntos
  Ejemplo: intersection([1,2], [2,3], X). → X = [2]
- - subset/2
  Descripcion: Subconjunto
  Ejemplo: subset([1], [1,2]). → true
- - subtract/3
  Descripcion: Diferencia de listas
  Ejemplo: subtract([1,2], [2], X). → X = [1]
- - select/3
  Descripcion: Elimina 1ra ocurrencia de un elemento
  Ejemplo: select(2, [1,2,3], R). → R = [1,3]
- - delete/3
  Descripcion: Elimina todas las ocurrencias
  Ejemplo: delete(2, [1,2,2,3], R). → R = [1,3]
- - reverse/2
  Descripcion: Invierte una lista
  Ejemplo: reverse([1,2,3], R). → R = [3,2,1]
- - atom/1
  Descripcion: ¿Es un átomo?
  Ejemplo: atom(foo). → true

- - number/1
  Descripcion: ¿Es un número?

Ejemplo: number(5). → true

- - numlist/3

Descripcion: Genera lista de enteros

Ejemplo: numlist(1, 3, L). → L = [1,2,3]

- - sum_list/2

Descripcion: Suma todos los elementos

Ejemplo: sum_list([1,2,3], X). → X = 6

- - flatten/2

Descripcion: Aplana listas anidadas

Ejemplo: flatten([1,[2,3]], X). → X = [1,2,3]

Operaciones Extra-logicas

- - is

Descripcion: Evalúa una expresión aritmética

Ejemplo: X is 2+3. → X = 5

- - \=

Descripcion: No unifican

Ejemplo: 1 \= 2. → true

- - ==

Descripcion: Iguales sin instanciar

Ejemplo: X = 1, Y = 1, X == Y. → true

- - =\=

Descripcion: Diferencia numérica

Ejemplo: 3 =\= 4. → true

- - >

Descripcion: Mayor que

Ejemplo: 4 > 2. → true

- - <

Descripcion: Menor que

Ejemplo: 2 < 3. → true

- - >=

Descripcion: Mayor o igual que

Ejemplo: 3 >= 3. → true

- - =<

Descripcion: Menor o igual que

Ejemplo: 2 =< 4. → true

- - abs

Descripcion: Valor absoluto

Ejemplo: abs(-5, A). → A = 5

- - max

Descripcion: Máximo de dos valores

Ejemplo: max(3, 5, X). → X = 5

- - min

Descripcion: Mínimo de dos valores

Ejemplo: min(3, 1, X). → X = 1

- - mod

Descripcion: Resto de división

Ejemplo: 5 mod 2 = 1

Metapredicado permitido

- - not

Descripcion: Negación por falla

Ejemplo: not(member(4, [1,2,3])). → true

SMALLTALK

Metodos de Coleccion

- - select:
  Descripcion: Filtra los elementos que cumplen con un bloque booleano
  Ejemplo: [1 2 3 4] select: [:x | x even]. → #(2 4)
- - reject:
  Descripcion: Filtra los elementos que NO cumplen con un bloque
  Ejemplo: [1 2 3] reject: [:x | x > 2]. → #(1 2)
- - collect:
  Descripcion: Transforma cada elemento aplicando un bloque
  Ejemplo: [1 2 3] collect: [:x | x * 2]. → #(2 4 6)
- - detect:
  Descripcion: Devuelve el primer elemento que cumple con el bloque
  Ejemplo: [1 2 3] detect: [:x | x odd]. → 1
- - inject:into:
  Descripcion: Reduce la colección con un acumulador
  Ejemplo: (1 to: 5) inject: 0 into: [:sum :x | sum + x]. → 15
- - fold:
  Descripcion: Similar a inject:into:, dependiendo del dialecto
  Ejemplo: ((1 to: 3) fold: [:x :y | x + y]). → 6
- - add:
  Descripcion: Agrega un elemento a una colección (mutable)
  Ejemplo: Set new add: 5. → a Set(5)
- - at:
  Descripcion: Accede a un índice o clave
  Ejemplo: #(a b c) at: 2. → b
- - at:put:
  Descripcion: Asigna valor en índice o clave (mutable)
  Ejemplo: Dictionary new at: 'key' put: 42.
- - do:
  Descripcion: Itera sobre la colección
  Ejemplo: [1 2 3] do: [:x | Transcript show: x printString].
- - keysAndValuesDo:
  Descripcion: Itera sobre diccionarios
  Ejemplo: dict keysAndValuesDo: [:k :v | ...]
- - keysDo:
  Descripcion: Itera solo por las claves
  Ejemplo: dict keysDo: [:k | ...]
- - valuesAndCountsDo:
  Descripcion: Itera con valores y su frecuencia
  Ejemplo: #(1 1 2) valuesAndCountsDo: [:v :c | ...]
- - withAll:
  Descripcion: Concatena colecciones
  Ejemplo: #(1 2) withAll: #(3 4). → #(1 2 3 4)
- - includes:
  Descripcion: Verifica pertenencia
  Ejemplo: #(a b) includes: 'a'. → true
- - includesKey:
  Descripcion: Verifica clave en diccionario
  Ejemplo: dict includesKey: 'key'. → true

Metodos de Metaprogramacion

- - doesNotUnderstand:
  Descripcion: Intercepta mensajes no entendidos
  Ejemplo: obj doesNotUnderstand: #foo. → define manejo personalizado
- - perform:
  Descripcion: Envía un mensaje dinámicamente
  Ejemplo: 'hello' perform: #size. → 5
- - respondsTo:
  Descripcion: Verifica si puede responder a un selector
  Ejemplo: 'abc' respondsTo: #size. → true
- - sendTo:
  Descripcion: Envía mensaje, similar a perform:

Ejemplo: obj sendTo: #message.

- - value

Descripcion: Evalúa bloques

Ejemplo: [3 + 4] value. → 7

- - class

Descripcion: Devuelve la clase del receptor

Ejemplo: 'abc' class. → String

- - species

Descripcion: Clase que representa instancias equivalentes

Ejemplo: #(1 2) species. → Array

- - superclass

Descripcion: Clase padre

Ejemplo: String superclass. → Object

- - isKindOf:

Descripcion: Verifica si instancia es de una clase o subclase

Ejemplo: 'a' isKindOf: Object. → true

- - subclass:instanceVariableNames:classVariableNames:category:

Descripcion: Define una nueva subclase dinámicamente

Ejemplo: Object subclass: #Person instanceVariableNames: 'name' classVariableNames: '' category: 'People'

- - allSubclasses

Descripcion: Lista todas las subclases directas e indirectas

Ejemplo: Object allSubclasses. → list

- - allSubclassesDo:

Descripcion: Itera sobre subclases

Ejemplo: Object allSubclassesDo: [:cls | ...]

- - withAllSubclassesDo:

Descripcion: Itera incluyendo self y subclases

Ejemplo: Object withAllSubclassesDo: [:cls | ...]

- - methodDictionary

Descripcion: Devuelve los métodos definidos

Ejemplo: String methodDictionary. → Dictionary

Clases utiles para Metaprogramacion

- - Object: Clase raíz de la jerarquía, base de todas las clases
- - Message: Representa un mensaje enviado
- - MessageSend: Forma de enviar mensajes diferidos
- - Behavior: Superclase de Class y Metaclass
- - Class: Instancias son clases, define comportamiento
- - Metaclass: Clase de una clase
- - Method: Representación interna de un método compilado
- - Context: Representa una activación de método (frame)

notación para definir clausures (bloques) en Smalltalk

- Los bloques se definen entre corchetes []. Pueden tomar argumentos y se evalúan con value, value:, etc.
- Ejemplo: [1 + 2] value => 3
- Ejemplo: [:x | x * 2] value: 5 => 10
- Ejemplo: [:x :y | x + y] value: 3 value: 4 => 7
- Ejemplo: [:x | x > 5 ifTrue: ['Mayor'] ifFalse: ['Menor']] value: 6 => 'Mayor'