

## **Class Object**

**isSmallInteger**  
"Answer true if receiver is an instance of class SmallInteger, else answer false."

**isString**  
"Answer true if receiver is an instance of class String, else answer false."

**= anObject**  
"This is the default equality test. Answer true if the receiver and anObject are the same object, else answer false."

**== anObject**  
"Answer true if the receiver and anObject are the same object, else answer false."

**asString**  
"Added by OSI - Answer the receiver's string representation"

**at: anInteger**  
"Answer the object in the receiver at index position anInteger. If the receiver does not have indexed instance variables, or if anInteger is greater than the number of indexed instance variables, report an error."

**at: anInteger put: anObject**  
"Answer anObject. Replace the object in the receiver at index position anInteger with anObject. If the receiver does not have indexed instance variables, or if anInteger is greater than the number of indexed instance variables, report an error."

**class**  
"Answer the class of the receiver."

**deepCopy**  
"Answer a copy of the receiver with shallow copies of each instance variable."

**error: aString**  
"Create a walkback window describing an error condition with the error message aString in the window label."

**isArray**  
"Answer true if receiver is an instance of class Array or one of its subclasses, else answer false."

**isAssociation**  
"Answer true if receiver is an instance of class Association or one of its subclasses, else answer false."

**isCharacter**  
"Answer true if receiver is an instance of class Character, else answer false."

**isClass**  
"Answer true if receiver is an instance of class Class or one of its subclasses, else answer false."

**isCollection**  
"Answer true if receiver is an instance of class Collection or one of its subclasses, else answer false."

**isDictionary**  
"Added by OSI"

**isFloat**  
"Answer true if receiver is an instance of class Float, else answer false."

**isKindOf: aClass**  
"Answer true if receiver is an instance of aClass or one of its subclasses, else answer false."

**isMemberOf: aClass**  
"Answer true if the receiver is an instance of aClass, else answer false."

**isNil**  
"Answer true if the receiver is the object nil, else answer false."

**isNumber**  
"Answer true if receiver is an instance of class Number or one of its subclasses, else answer false."

**isString**  
"Answer true if receiver is an instance of class String, else answer false."

**notNil**  
"Answer true assuming the receiver is not the object nil."

**respondsTo: aSymbol**  
"Answer true if the receiver class or one of its superclasses implements a method with selector equal to aSymbol."

**shallowCopy**  
"Answer a copy of the receiver which shares the receiver instance variables."

**size**  
"Answer the number of indexed instance variables in the receiver."

**yourself**  
"Answer the receiver."

**~= anObject**  
"Answer true if the receiver and anObject do not compare equal (using =), else answer false."

**~~ anObject**  
"Answer true if the receiver and anObject are not the same object, else answer false."

## **Class Magnitude**

**< aMagnitude**  
"Answer true if the receiver is less than aMagnitude, else answer false."

**<= aMagnitude**  
"Answer true if the receiver is less than or equal to aMagnitude, else answer false."

**= aMagnitude**  
"Answer true if the receiver is equal to aMagnitude, else answer false."

**hash**  
"Answer the positive integer hash value for the receiver."

## **Class Ordered Collection**

**, aCollection**  
"Answer an OrderedCollection containing all the elements of the receiver followed by all the elements of aCollection."

**add: anObject afterIndex: anInteger**  
"Answer anObject. Insert anObject at index position anInteger + 1 in the receiver collection. If anInteger is out of the collection bounds, report an error."

**add: anObject beforeIndex: anInteger**  
"Answer anObject. Insert anObject at index position anInteger - 1 in the receiver collection. If anInteger is out of the collection bounds, report an error."

**after: anObject ifNone: aBlock**  
"Answer the element that immediately follows anObject in the receiver collection. If anObject is not an element of the receiver, aBlock is evaluated (with no arguments)."

at: anInteger  
 "Answer the element of the receiver at index position anInteger. If anInteger is an invalid index for the receiver collection, report an error."

at: anInteger put: anObject  
 "Answer anObject. Replace the element of the receiver at index position anInteger with the anObject. If anInteger is an invalid index for the receiver collection, report an error."

before: anObject ifNone: aBlock  
 "Answer the element that immediately precedes anObject in the receiver collection. If anObject is not an element of the receiver, aBlock is evaluated (with no arguments)."

copyFrom: beginning to: end  
 "Answer an OrderedCollection containing the elements of the receiver from index position beginning through index position end."

do: aBlock  
 "Answer the receiver. For each element in the receiver, evaluate aBlock with that element as the argument."

includes: anObject  
 "Answer true if the receiver contains an element equal to anObject, else answer false."

remove: anObject ifAbsent: aBlock  
 "Answer anObject. Remove the element anObject from the receiver collection. If anObject is not an element of the receiver, aBlock is evaluated (with no arguments)."

removeAbsoluteIndex: anInteger  
 "Private - Answer the receiver. Remove the element of the receiver at absolute index position anInteger. If anInteger is an invalid index for the receiver, report an error."

replaceFrom: start to: stop with: aCollection  
 "Answer a new OrderedCollection containing the receiver whose elements at index position start through stop have been replaced by the elements of aCollection."

size  
 "Answer the number of elements contained by the receiver collection."

startPosition: start endPosition: end  
 "Private - Answer the receiver. Set the position of the first and last elements of the receiver, to the arguments start and stop respectively."

### **Class Array**

isArray  
 "Answer true if receiver is an instance of class Array or one of its subclasses, else answer false."

printOn: aStream  
 "Append the ASCII representation of the receiver to aStream."

storeOn: aStream  
 "Append the ASCII representation of the receiver to aStream from which the receiver can be reinstated."

### **Class String**

< aString  
 "Answer true if the receiver is before aString, else answer false. The comparison is not case sensitive."

<= aString  
 "Answer true if the receiver is before or equal to aString, else answer false. The comparison is not case sensitive."

= aString  
 "Answer true if the receiver is equal to aString, else answer false. The comparison is case sensitive."

asDate  
 "Answer a Date representing the date described by the receiver. The receiver must contain first the day number then the month name and then the year separated by blanks."

asInteger  
 "Answer the integer conversion of the receiver; the receiver is expected to be a sequence of digits with optional leading minus sign."

asLowerCase  
 "Answer a String containing the receiver with alphabetic characters in lower case."

asUpperCase  
 "Answer a String containing the receiver with alphabetic characters in upper case."

at: anInteger  
 "Answer the character at position anInteger in the receiver string."

at: anInteger put: aCharacter  
 "Answer aCharacter. At index position anInteger in the receiver put the character aCharacter."

basicAt: anInteger  
 "Answer the character at position anInteger in the receiver string."

basicAt: anInteger put: aCharacter  
 "Answer aCharacter. At index position anInteger in the receiver put the character aCharacter."

isString  
 "Answer true if receiver is an instance of class String, else answer false."

replace: count with: aCollection  
 "Answer the receiver. Replace count number of characters in the beginning of the receiver from the same number of characters in the beginning of aCollection. aCollection can be a WinAddress and count can be a large integer."

replaceFrom: start to: stop with: aString startingAt: repStart  
 "Replace the characters of the receiver at index positions start through stop with consecutive characters of aString beginning at index position repStart. Answer the receiver."

replaceFrom: start to: stop withObject: aCharacter  
 "Replace the characters of the receiver at index positions start through stop with aCharacter. Answer aCharacter."

size  
 "Answer the size of the receiver string."

storeOn: aStream  
 "Append the ASCII representation of the receiver to aStream from which the receiver can be reinstated."

## Class Dictionary

**associationsDo:** aBlock  
"Answer the receiver. For each key/value pair in the receiver, evaluate aBlock with that pair as the argument."

**associationsSelect:** aBlock  
"For each key/value pair in the receiver, evaluate aBlock with the association as the argument. Answer a new object containing those key/value pairs for which aBlock evaluates to true."

**at:** aKey  
"Answer the value of the key/value pair whose key equals aKey from the receiver. If not found, report an error."

**deepCopy**  
"Answer a copy of the receiver with shallow copies of each element."

**do:** aBlock  
"Answer the receiver. For each value in the receiver, evaluate aBlock with that value as the argument."

**findKeyIndex:** aKey  
"Private - Answer the index position of the key/value pair in the receiver whose key equals aKey or the index of the first empty position where such a pair would be stored."

**includes:** anObject  
"Answer true if the receiver contains the key/value pair whose value equals anObject, else answer false."

**isDictionary**  
"Added by OSI"

**keysDo:** aBlock  
"Answer the receiver. For each key in the receiver, evaluate aBlock with the key as the argument."

**lookupKey:** aKey  
"Private - Answer the association in the receiver whose key equals aKey or nil if it doesn't exist."

**occurrencesOf:** anObject  
"Answer the number of key/value pairs in the receiver, whose values are equal to anObject."

**removeKey:** aKey ifAbsent: aBlock  
"Answer aKey. Remove the key/value pair whose key equals aKey from the receiver. If such a pair is not found, evaluate aBlock (with no arguments)."

**select:** aBlock  
"For each key/value pair in the receiver, evaluate aBlock with the value part of the pair as the argument. Answer a new object containing those key/value pairs for which aBlock evaluates to true."

**shallowCopy**  
"Answer a copy of the receiver which shares the receiver elements."

**storeOn:** aStream  
"Append the ASCII representation of the receiver to aStream from which the receiver can be reinstantiated."

## Class SortedCollection

**addAll:** aCollection  
"Answer aCollection. Add all the elements in aCollection to the receiver in sorted order."

**addAllFirst:** aCollection  
"Add all the elements of aCollection to the receiver before its first element. This method reports an error since the sortBlock determines element order."

**addAllLast:** aCollection  
"Add all the elements of aCollection to the receiver after its last element. This method reports an error since the sortBlock determines element order."

**at:** anInteger put: anObject  
"Replace the element at index position anInteger in the receiver collection with anObject. This method reports an error since the sortBlock determines element order."

**copyFrom:** beginning to: end  
"Answer a SortedCollection containing the elements of the receiver from index position beginning through index position end."

**sort:** lower to: upper  
"Private - Sort the elements in the receiver that are between lower and upper index positions."

**sortBlock:** aBlock  
"Answer the receiver. Set the sort block for the receiver to aBlock and resort the receiver."

**sortBlock**  
"Answer the block that determines sort ordering for the receiver."

## **Implementar Clase Singleton**: (ejemplos del TP del año 2004)

```
OrderedCollection subclass: #Flota
  instanceVariableNames: ""
  classVariableNames: 'UnicaInstancia'
```

### **Flota class methods**

#### **new**

```
( UnicaInstancia isNil )
  ifTrue: [ UnicaInstancia:= super new ] .
^UnicaInstancia.!
```

#### **actualizarEstado**

```
UnicaInstancia do: [ :unaAeronave |
  ( unaAeronave getVuelos ) do: [ :unVuelo |
    unVuelo actualizarEstado
  ].
].!
```

#### **agregarAeronave:unaAeronave**

```
(( UnicaInstancia detect: [:x | x getNombre = unaAeronave getNombre] ifNone:[nil] ) isNil )
  ifTrue: [^UnicaInstancia add: unaAeronave]
  ifFalse: [MessageBox message: 'El nombre de la nave ya existe'.]!
```

buscarAeronave:unNombre

```
^ UnicaInstancia detect: [:unaAeronave | unaAeronave getNombre = unNombre] ifNone:[nil].!
```

[.....]

### **Flota methods**

((no hay ninguno))

## **Copia en SmallTalk**

Comentario: Prueba es una clase cuyas instancias sólo tienen dos atributos: "nombre" y "dato". Entre sus métodos de instancia tenemos *getDato*: y *set:con*:

```
| a b c t1 t2 t3 |
```

```
a:=Prueba new.
a set: 1 con: #a.
b:=Prueba new.
b set: 1 con: #b.
c:=Prueba new.
c set: 1 con: #c.
```

```
t1:=OrderedCollection new add:a;add:b;add:c.
t2:=OrderedCollection new.
```

```
1 to: (t1 size) do: [:x| t2 add: ((t1 at:x) getDato)].
t2 yourself. "OrderedCollection(1 2 3)"
a set:4 con:#d.
```

```
t3:=t1 deepCopy.
t3=t1. "false"
t3==t1 "false"
```

```
t3:=t1 shallowCopy (igual que copy).
t3=t1. "true"
t3==t1 "false"
```

```
t3:=t1.
t3=t1. "true"
t3==t1 "true"
```

```
1 to: (t3 size) do: [:x| t2 add:((t3 at:x) getDato) ].
t2 yourself. "OrderedCollection(1 2 3 4 2 3)"
```

```
| x y |
x:=1
```

```
y:=x deepCopy.
x=y. "true"
x==y. "true"
```

```
y:=x shallowCopy.
x=y. "true"
x==y. "true"
```

```
y:=x.
x=y. "true"
x==y. "true"
```