**METODOS Y CLASES**

**“CLASE 09/10”**

“Realizar un método que devuelva si un número es primo o no. El método se debe llamar: esPrimo y devolver *verdadero* o *falso*.”

“Clase: SmallInteger”

esPrimo| primo divisor resto |"self es el numero que viene como parametro"(self <= 1) ifTrue: [ ^false. ].divisor:= 2.primo:= true." ~= es el signo distinto "[primo & (((self sqrt)) >= divisor)] whileTrue: [ ((self \\ divisor) = 0) ifTrue: [ primo:= false. ] ifFalse: [ divisor:= divisor + 1. ]. ].^primo.

“Realizar un método que devuelva un arreglo de dos (2) elementos donde el elemento uno es la parte entera y el elemento 2 es la parte decimal

Ejemplo: 23.44 **dividirNumero** ----> #(23 44) “

“Clase: SmallFloat64“

DividirNumero| nro |nro:= self asString.nro:= nro substrings: '.'.nro at: 1 put: (nro at: 1) asInteger.nro at: 2 put: (nro at: 2) asInteger.^nro.

“Dado un arreglo de números encontrar, realizar un método que encuentre el mayor valor y otro método que encuentre el menor valor. “

“Clase: Array”

buscoMayor| mayor |mayor:= self at: 1. "tomo el primer elemento como mayor""salteo la primer iteracion"2 to: (self size) do: [ :i| ((self at: i) isNumber) ifTrue: [ ((self at: i) > mayor) ifTrue: [ mayor:= self at: i. ]. ]. "si no es un numero, lo saltea" ].^mayor.

**"CLASE 10/10"**

**"Cargar un arreglo con 30 elementos aleatorios, entre –1000 y 1000.**

**Crear una lista con los elementos que estén en un rango ingresado por el usuario.**

**Imprimir la lista.**

**Informar la cantidad de elementos de la lista."**

| lista arreglo menor mayor |

lista:= OrderedCollection new.

arreglo:= Array new: 30. "creo un arreglo de 30 elementos"

"cargo el arreglo"

1 to: (arreglo size) do: [ :i|

"() atRandom , te da un elemento aleatorio en ese rango"

arreglo at: i put: ((-1000 to: 1000) atRandom).

].

menor:= (UIManager default request: 'Ingrese el menor valor del rango de la lista') asInteger.

"controlo que no ingrese una palabra"

[menor = nil] whileTrue: [

menor:= (UIManager default request: 'Numero no valido, intente nuevamente') asInteger.

].

mayor:= (UIManager default request: 'Ingrese el mayor valor del rango de la lista') asInteger.

"controlo que no ingrese una palabra y que este bien ingresado el rango"

[mayor = nil and: (menor > mayor)] whileTrue: [

mayor:= (UIManager default request: 'Numero no valido, intente nuevamente') asInteger.

].

"cargo la lista"

arreglo do: [:c|

"pregunto si el elemento esta en el rango solicitado"

((c >= menor) and: (c <= mayor)) ifTrue: [

lista add: c.

].

].

Transcript clear.

Transcript show: (arreglo) ; cr.

Transcript show: 'El rango inferior es ', menor asString, ' y el rango superior es ', mayor asString ; cr.

Transcript show: (lista) ; cr.

Transcript show: ('La cantidad de elementos de la lista es ', (lista size) asString).