Paradigmas de Programación
Paradigma de Objetos

Programación en Smalltalk

Expresiones de mensajes

Para cada uno de los siguientes ejercicios de evaluaciones de expresiones de mensajes, indique el resultado de dichas expresiones, justificando claramente su respuesta.

Ejercicio 1

```
|lista|
lista := OrderedCollection new.
lista add:(OrderedCollection new).
(lista at:1) add:#(1 2).
lista add:((lista at:1) at:1).
(lista at:2) at:1 put:33.
^lista.
```

Ejercicio 2

```
|lista|
lista := OrderedCollection new.
lista add:(OrderedCollection new).
(lista at:1) add:#(1 2).
lista add:((lista at:1) at:1).
(lista at:2) at:1 put:33.
lista add:'SmallTalk'.
lista removeAt:2.
lista add:(((lista at:1) at:1) at:2).
^lista.
```

Ejercicio 3

```
|b c t|

b := [:tope :colleccion|
       1 to: tope do:[:i| colleccion add:(i*2)].
      ].
c := OrderedCollection new.
b value:10 value:c.
t := c.
b value:10 value:t.
^(t =c).
```

Ejercicio 4

```
|a b c|
a:= [ 2 + 3.[super yourself. b]. a yourself].
b:= a value.
```

c:= b value yourself.
(a == b) not | (b == c) not.

Ejercicio 5

Se evalúa la siguiente expresión en una ventana Workspace del intérprete Smalltalk

```
A :=  OrderedCollection new.
A add: (1 @ 2) ;  add: (2 @ 2) x + 1.
```

Se le solicita que indique y justifique:
a) El orden en que se evalúan los distintos mensajes.
b) A qué objeto estará asociada la variable global A luego de la mencionada evaluación.

Nota
x es un método instancia de la clase punto (Point) que retorna la coordenada x del punto.

Ejercicio 6

Para la siguiente expresión de asignación, indicar cuales afirmaciones son correctas y cuales incorrectas y justificar la respuesta.

```
.....
A := B + 1 * 2; class.
```

a. Se crea el objeto B.
b. Se copia el objeto referenciado por A.
c. A referencia a la clase del contenido de B.
d. A y B referencian distintas instancias de la misma clase.

Ejercicio 7

Indicar a que objetos estarán asociadas las variables globales B y C luego de evaluar la siguiente secuencia de expresiones.

```
A :=  OrderedCollection new.
A add: (OrderedCollection new); add: ((A at: 1) add: 1; yourself).
B := (A at: 1) = (A at: 2).
C := (A at: 1) = = (A at:2).
```

Ejercicio 8

Dados los siguientes fragmentos de código:

a)	b)
| a block | a := 1. block := [a]. a := 3. ^block value	| a block | a := 1. block := [:x | [x]] value: a. a := 3. ^block value

Indicar el resultado de la evaluación de cada uno y justifique.

Ejercicio 9

Se ha definido una clase Persona, subclase de Object. La clase Persona posee el método de instancia: *inicializa*

```
edad := 0.
peso := 0.
dni := 0.
```

A continuación se proponen tres alternativas para el método de clase de creación de instancias **new**

a. New ^super new inicializa.	b. New ^self new inicializa.	c. New ^Object new inicializa.

Indique y justifique cuál es la alternativa correcta.

Ejercicio 10

Dadas las siguientes clases en Smalltalk

CLASE	Object subclass: #Figura InstanceVariableNames: 'nombre centro ' classVariableNames: '' poolDictionaries: ''	Figura subclass: #Cuadrado InstanceVariableNames: 'lado color ' classVariableNames: '' poolDictionaries: ''
METODOS DE CLASE	new | instancia | instancia := super new. instancia inicializar. ^instancia.	new | instancia | instancia := super new. instancia inicializar. ^instancia.
METODOS DE INSTANCIA	inicializar nombre := UIManager default request: 'Ingrese Nombre' initialAnswer: ''. centro := UIManager default request: 'Ingrese Centro' initialAnswer: ''.	inicializar lado := UIManager default request: 'Ingrese la dimension del lado' initialAnswer: ''. color:= UIManager default request: 'Ingrese el color' initialAnswer: ''.
	centro ^centro.	

a) Indicar qué ocurre y qué se le consulta al usuario al evaluar la siguiente expresión en el intérprete Smalltalk:

```
A := Cuadrado new.
```

b) Simule una interacción con el programa, e indique que valor se presentaría ante la evaluación de las siguientes expresiones de mensajes:

```
A := Cuadrado new.
   A centro.
```

c) Modifique los métodos dados, manteniendo el polimorfismo de manera de obtener un resultado correcto.

Ejercicio 11

Suponiendo la siguiente jerarquía de clases y definición de métodos:

`Clase A:`
`  Método y: { ... }`
	`Clase B, subclase de A:`
	`  método x: { ...`
	`  self y.`
	`  super y.`
	`  ... }`

`Clase C, subclase de B:`
`  Método y: { ... }`
	`Clase D, subclase de C:`
	`  método x: { ...`
	`  super y.`
	`  super x.`
	`  ... }`

Si se invoca el método x de una instancia de la clase D, indique claramente cuáles son los métodos activados y el orden y número de invocaciones de los mismos.

Desarrollos de programación

Ejercicio 1

Se le solicita que genere una clase arreglo que tenga las siguientes particularidades:

1. en lugar del método **at: put:**, estará disponible el método **at: put: usuario:**
2. en lugar del método **at:** estará disponible el método **at: usuario:**
3. en lugar del método de clase **new:** estará disponible el método **new: usuario:**

El argumento de **usuario:** será la identificación de uno de los usuarios autorizados para crear y acceder a las instancias, por lo cual la misma deberá poseer la información de qué usuarios pueden ver o modificar sus valores. Si bien esta clase deberá ser en todos sus aspectos igual a **Array**, no deberá permitir que sus variables sean vistas o modificadas por los mensaje **at: put:** o **at:**.

Ejercicio 2

Se desea desarrollar un sistema para una concesionaria de vehículos. Los vehículos se clasifican en: autos, camionetas y motocicletas. Todos los vehículos tienen una marca, un modelo (año), una patente y un kilometraje. Los autos pueden clasificarse en autos deportivos y no deportivos. Los autos deportivos tienen la propiedad de tener registrado el país de origen. Además la concesionaria lleva un registro de las ventas realizadas, cada registro consiste de la siguiente información: monto de la venta, vehículo vendido, apellido, nombre y dni del comprador. Se le solicita que defina la clase Administrador para administrar los autos en stock y las ventas realizadas.

Implementar el modelo teniendo en cuenta que la clase Administrador debe responder a los siguientes mensajes:

- **new**

Crea una instancia de Administrador sin objetos cargados.

- **agregarVehiculo:** unVehiculo

Agrega el vehículo argumento al sistema. unVehículo es la identidad de un objeto vehículo.

- **totalVentasAutos**

Retorna una OrderedCollection con todos los autos vendidos por la concesionaria.

- **venderAuto:**unAuto **datosCliente:**datosCliente

Crea una instancia de venta de auto con el auto argumento y los datos del cliente. El auto argumento debe ser dado de baja del stock (se supone que existe).

- **totalVentasMotocicletas**

Retorna una OrderedCollection con todos los **dni** de clientes que compraron una motocicleta.

Además, deberá definir todos los métodos correspondientes a las todas las clases de la jerarquía planteada.

Ejercicio 3

Se le solicita que cree en Smalltalk la clase *arbolBinario*. Un *arbolBinario* puede ser un árbol vacío (sin subárbol izquierdo ni derecho) o poseer uno o ambos. Cada nodo del árbol tendrá asociado un valor.
Se le pide que especialmente defina los siguientes métodos:

vacío
Método de clase que crea un árbol binario vacío (sin subárbol izquierdo ni derecho)

valor: unValor **arbolIzquierdo:** unArbolBinario **arbolDerecho:** unArbolBinario
Método de clase que crea un árbol no vacío

valor
Método instancia que retorna el valor de la raíz del árbol

esVacio
Método instancia que retorna TRUE si el objeto receptor es un árbol vacío (se considera que un árbol es vacío si el subárbol izquierdo es Nil)

profundidad
Método de instancia que retorna la profundidad de un árbol binario (puede utilizar el método binario **max:** de los enteros para comparar longitudes)**.**

Ejercicio 4

Se le solicita que defina una clase Matriz a partir de la especialización de la clase Dictionary, la cual deberá representar el concepto de matriz cuadrada, cuyo contenido serán enteros. Se requiere que las claves correspondan a enteros que representen los números de la fila de la matriz.

Se le solicita que defina los siguientes métodos:

De clase:

new: n (para generar una matriz de n x n)

De instancia:

at: columnaFila
El argumento es un par con la fila y la columna, el cual retorna el valor en esa posición.

atColumnaFila: par **put:** valor

matrizReducida: filaColumna

El cual retorna como resultado otra matriz, que es exactamente igual que la receptora, pero a la cual se le han eliminado la fila *filaColumna* y la columna *filaColumna*.

Ejercicio 5

Idem 5, pero que no sea especialización de Dictionary.

Ejercicio 6

Existen muchas aplicaciones en las cuales se utilizan matrices de dos dimensiones de gran tamaño. Si la mayoría de los elementos de la matriz son ceros, la matriz se denomina Matriz Rala. Obviamente una matriz rala puede ser eficientemente representada si se almacenan en memoria sólo las celdas con valores distintos de ceros. Por lo tanto, se le solicita que:
Defina en Smalltalk una clase MatrizRala para representar este tipo de matrices (puede definir otras clases si lo necesita para completar el concepto).

Defina como mínimo los siguientes métodos:

NewFila: nFilas **columna:** nColumnas	Crea una instancia de MatrizRala de *nFilas* x *nColumnas*
= matriz	Compara dos matrices ralas
atFila: fila **columna:** columna **put:** valor	Asigna valor a la celda(fila, columna)
suma: matriz	Retorna la matriz rala resultante de la suma
Producto: matriz	Retorna la matriz rala resultante del producto.

M1 M2

2	3	0
0	0	4
0	0	0
0	0	0
0	5	0
0	0	0
0	0	0
0	0	2

X

0	0	3	0	0	0	0	0
0	0	0	0	9	0	3	0
4	0	0	0	0	0	0	2

= M3

0	0	6	0	27	0	9	0
16	0	0	0	0	0	0	8
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	45	0	15	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	4

M1, M2 y M3 serían tres ejemplos de matrices ralas. En M1 hay sólo cinco elementos distintos de cero, y serían los únicos que habría que mantener en memoria. M3 es la matriz rala producto de M1 y M2, y en ella sólo hay 9 valores distintos de cero, y serían los únicos que deberían mantenerse en memoria.

Ejercicio 7

El método básico **do:** permite iterar sobre los elementos de una colección. Este método toma un bloque de un parámetro como argumento y ejecuta dicho bloque para cada elemento de la colección, pasando el elemento como argumento para el bloque.
El método no está implementado en la clase Collection y debe hacerse para cada subclase de la clase Collection.
La clase Interval, es una colección ordenada de valores que representan una secuencia aritmética. Se caracteriza por los valores de inicio y final y el tamaño de paso entre los elementos del intervalo. El tamaño de paso puede ser positivo o negativo.

1. Definir el método **do:** para la clase Interval.
2. Idem para la clase Array.
3. Escribir en la clase Collection el método que permita determinar el número de veces que un elemento está presente en una colección.

Ejercicio 8

Suponer que se quiere definir una clase ventana para implementar un entorno gráfico. Se le solicita que especifique una clase Ventana en Smalltalk, la cual puede contener a su vez ventanas. Una ventana tendrá definida un nombre, su posición por el vértice superior izquierdo con valores de las coordenadas X e Y, y posee un alto y ancho.

Se le solicita que defina especialmente los métodos:

De clase

new

Al crearse la instancia se deberán incorporar el nombre, la posición y las dimensiones.
Tenga en cuenta que una ventana poseerá ventanas contenidas.

De instancia:

moverAx: valorX **Y:** valorY
borrarVentana: unaVentana
cerrar (coloca ancho y alto en cero)
dimensionarX: Ancho **Y:** Alto
adicionarSubVentana: unaVentana

Posiblemente para implementar estos métodos requiera definir otros métodos auxiliares. Tenga fundamentalmente en cuenta que cuando mueve una ventana también mueve las ventanas contenidas, y lo mismo ocurre al cerrarla. Otra consideración importante es que se exigen que no puede haber dos ventanas con el mismo nombre.