Universidad Nacional de Quilmes

Técnicas Avanzadas de Programación - 2010

El estudio sin pensar es vano; el pensar sin estudiar es peligroso – Confucio

Práctica n° 5

Ejercicio 1

Analice y compare, evaluando en el workspace que devuelven y que hacen los siguientes mensajes para cada una de las siguientes clases: `OrderedCollection`, `SortedCollection`, `Array`, `Dictionary`, `Bag` y `Set`.

`add:`	`at:`	`at: put:`
`size`	`do:`	`detect:`
`select:`	`collect:`	`reject:`
`Inject: into:`	`includes:`	`isEmpty`

Ejercicio 2

Responda a las siguientes preguntas:

a) ¿Es posible que algunos mensajes del ejercicio 1 no sean semánticamente correctos para algunas clases de colecciones?

Por ejemplo:

¿Se le puede enviar el mensaje `#add:` a un `Array`? ¿y a un `Set`?

¿Se le puede enviar el mensaje `#at:` y `#at:put:` a un Set?

b) En respuesta al mensaje `#select:`,

¿Qué retorna un `Array`? ¿Y un `Dictionary`? ¿Y una `SortedCollection`?

c) En respuesta al mensaje `size`:

¿Qué retorna un `Array` creado con `Array new: 10`?

¿Qué retorna una `OrderedCollection` creada con `OrderedCollection new:10`?

d) ¿Es posible remover un elemento de un `Array`? ¿Cómo?

e) ¿Cómo se averigua la posición de un elemento en un `Array`?

¿Y la del primer elemento que cumple una condición?

¿Es posible hacerlo en un `Set`?

f) Que retorna el `#detect:` si no encuentra ningún objeto que cumpla la condición?

g) Encuentre en Smalltalk un uso interesante de `#detect:ifNone:`

¿Para qué sirve el bloque que se manda como parámetro en el ifNone?

h) ¿Cómo crea una `SortedCollection` para contener Strings ordenados alfabéticamente?

¿Y para Strings ordenados por tamaño?

i) ¿Cómo consigue los elementos en un `Array` eliminado las repeticiones?

j) ¿Para qué sirve el mensaje `yourself` en las colecciones?

k) ¿Cuál es el problema con la siguiente expresión si col es un `Set` con elementos?

```
col do: [ :each | col remove: each]
```

¿Y si es una `OrderedCollection`?

l) Cual es el efecto de enviar el mensaje `#add:` con `nil` como parámetro a una `OrderedCollection`? ¿Por qué?

m) ¿Qué diferencia hay entre los mensajes `#includes:` y `#contains:`?

Ejercicio 3

Implemente para las colecciones el `#select:`, `#reject:`, `#collect:` y `#detect: ifNone:`, suponiendo la existencia del `#do:`.

Ejercicio 4

Dada una `OrderedCollection` llamada alumnos conteniendo los alumnos inscriptos en un curso, escribir expresiones Smalltalk para resolver los siguientes problemas:

a) obtener el tercer elemento de la colección.

b) reemplazar el segundo elemento de la colección por el objeto otroAlumno.

c) agregar el alumno unAlumno al final de la colección.

d) saber la cantidad de alumnos inscriptos en el curso.

e) determinar si algún alumno obtuvo la calificación 10.

f) obtener una colección con todos los alumnos que aprobaron el curso (calificación superior a 4).

g) determinar si la cantidad de alumnos que aprobó el curso es mayor a la cantidad que desaprobó.

h) calcular el promedio de las calificaciones obtenidas por los alumnos.

i) obtener una colección conteniendo las edades de todos los alumnos del curso.

j) obtener una colección con todos los alumnos que no viven en Quilmes.

k) determinar cuál fue la calificación más frecuente.

l) obtener una colección conteniendo los nombres de todas las ciudades (excepto Quilmes) donde viven alumnos.

m) determinar si el curso fue desaprobado por todos los alumnos.

n) modificar la calificación obtenida por un alumno determinado.

ñ) iniciar en cero las calificaciones de todos los alumnos.

Nota 1: Hacer en papel

***Nota* 2:** cada alumno puede entender los siguientes mensajes: nombre (retorna el nombre del alumno), edad (retorna la edad del alumno), dirección (retorna el nombre de la ciudad donde vive el alumno), calificación (retorna la calificación que el alumno obtuvo en el curso), calificación: (modifica la calificación del alumno).

Ejercicio 5

Implemente la clase `BoundedBag`. Una instancia de un `BoundedBag` es un `Bag` que tiene un límite máximo de elementos.

Las alternativas para implementar esta clase son: una instancia de `Bag` es colaborador de un `BoundedBag`, y la clase `BoundedBag` es una subclase de la clase `Bag`. Justifique su elección.

Ejercicio 6 (Opcional)

Implementar en Smalltalk la clase `Matriz` con al menos el siguiente protocolo:

`at: put:`, que recibe un Point y un objeto, y pone el objeto en la posición dada.

do:, que recorre los elementos de la matriz por filas y columnas, ejecutando el bloque que recibe como parámetro.

`collect:`, que recorre los elementos de la matriz por filas, ejecutando el bloque que recibe como parámetro, y retorna una nueva matriz con los resultados de la ejecución en cada elemento.

`select:`, que recibe un bloque y retorna una colección con los objetos que cumplen la condición representada en el bloque.

`detect: ifNone:`, que recibe un bloque que evalúa en cada objeto recorriéndolos por fila y columna, y retorna el primer elemento que cumple la condición del bloque. Si ninguno la cumple, evalúa el segundo parámetro, que es también un bloque.

`size`, que retorna el tamaño de la matriz (#filas x #colunmas)

Ejercicio 7

Un nuevo y moderno supermercado ha decidido equipar a sus carritos con un sistema de última tecnología que les permite a los compradores mantener un mejor control sobre sus compras. El carrito permite realizar a los usuarios distintas consultas acerca de los productos que contiene, entre ellas:

- obtener el monto total de dinero gastado hasta el momento.
- obtener todos los artículos comprados, sin repeticiones.
- saber cuántos artículos cuyo costo es mayor a 50 pesos hay en el carrito.
- dejar en el carrito solo aquellos productos que no son de marca "Arcor".
- obtener el primer producto de la marca "Bagley". Si no hay producto de esa marca, devolver nil.
- obtener el producto más caro.
- preguntar si se compro lechuga.
- preguntar si ya se compraron todos los productos de una "lista de compras".
- saber la cantidad de Coca Colas que se compraron.

De los productos nos interesa saber entonces, el precio, la marca, y el tipo. Modele ahora la clase `CarroDeSuper` que nos permita realizar todas las consultas mencionadas. No olvide que a un carro de compras se le deber poder agregar y quitar productos.

Ejercicio 8

Implemente en Smalltalk la clase `IterableList`. Una `IterableList` consiste en una lista que responde solamente al siguiente protocolo.

`size`, que retorna el tamaño de la lista

`next`, que retorna el siguiente elemento

`prev`, que retorna el elemento anterior

`current`, que retorna el elemento actual

`add: anElement`, que añade un elemento en la posición actual, haciendo que el elemento actual sea ahora el elemento anterior.

`hasPrev`, que retorna un Boolean determinando si hay o no un elemento previo.

`hasNext`, que retorna un Boolean determinando o no si hay un siguiente elemento

Ejercicio 9 (Opcional)

Una `CircularList` es similar a una `IterableList`, pero a diferencia de esta, siempre tiene siguiente o previo elemento ya que el siguiente del último es el primero, y el previo del primero, el último.
Implemente la `CircularList` en Smalltalk.

Ejercicio 10

Nos piden hacer parte del modelo de un ciber. En un ciber hay muchas computadoras, de cada una nos van a interesar: el espacio libre en disco (expresado en Gb), la marca del mother, y los nombres de los programas cargados.
El ciber puede recibir pedidos, en cada pedido se indica qué programas se van a usar y cuánto espacio en disco requieren.
Implementar el modelo de cibers de forma tal que se le pueda

a) agregarle una compu a un ciber.
b) preguntar cuántas computadoras hay en un ciber con más de un determinado espacio libre en disco (p.ej. ¿cuántas compus hay en el ciber1 con más de 10 Gb libres en disco?)
c) pedir una compu que tenga instalado un determinado programa, nil si no hay ninguna (p.ej. dame una compu del ciber1 que tenga instalado el Firefox).
d) obtener una colección con las marcas de los mother de un ciber.
e) obtener el espacio libre en disco total de las compus de un ciber.
f) obtener una colección con las compus copadas de un ciber, una compu es copada si tiene al menos 5 programas instalados y además el mother es Asus.
g) obtener las compus de un ciber que pueden satisfacer un pedido, o sea, que estén instalados todos los programas que el pedido necesita, y que tenga espacio en disco suficiente para lo que requiere el pedido.
h) manejar la lista de pedidos de un ciber. Hay que registrar los pedidos que llegan que quedan como pedidos pendientes. Los pedidos se atienden por orden de llegada. El ciber tiene que entender un mensaje atenderPrimerPedido que toma el primer pedido pendiente, obtiene una compu que lo puede atender, le asigna la compu al pedido (el pedido se tiene que acordar qué compu le fue asignada), y pasa el pedido de la lista de pendientes a la lista de despachados. Si no hay ninguna compu que satisfaga al pedido, este pasa de la lista de pendientes a la lista de rechazados.
i) después de lo anterior, conocer el conjunto de programas que no están instalados en ninguna compu y que sí están en al menos un pedido rechazado.

Ejercicio 11

Implemente y pruebe en Smalltalk la clase `FilteredSet`. Un `filteredSet` es un `Set` que solo agrega aquellos elementos que cumplen una condición. La condición se indica al momento de crear una instancia, y es particular a cada una.
¿Cuál es la superclase de `FilteredSet`?
¿Cómo modela la condición?

Ejercicio 12

Implementar en Smalltalk las clases `Pila` y `Cola`.

Una `Pila` es una estructura *LIFO* (Last In Last Out). Es decir, el último elemento agregado, es el último en ser recuperado. Pueden imaginarse una pila de CDs, o de platos. Para tomar el plato de más abajo, primero necesitan tomar todos los de más arriba.

Una `Cola` es una estructura *FIFO*, (First In First Out). El primer elemento en ser agregado es el primero en ser recuperado. Imagínense la cola de un banco, el primero que llega, es el primero en ser atendido.

Ambas estructuras deben tener el siguiente protocolo:

`new`, que crea una nueva pila o cola.

`push: anObject` , que añade anObject a la pila o cola

`pop`, que recupera el elemento correspondiente de la pila o cola. En el caso de la pila, el último en ser agregado, en caso de la cola, el primero que se agrego.

`size`, que determina el tamaño de la pila o cola.

Ejercicio 13 (Opcional)

Una `ColaDoble` es una secuencia de elementos a la que se puede agregar y sacar por ambos extremos. Es decir que una `ColaDoble` puede tratarse como `Cola` del derecho y del revés. Implemente la clase `ColaDoble`. El protocolo a implementar es el siguiente:

`pop`, que devuelve el elemento del frente.

`popBack`, que devuelve el elemento del final.

`push:`, que agrega un elemento al frente.

`PushBack:`, que agrega un elemento al final

¿Cuál es la relación entre una `Cola` y una `ColaDoble`?

¿Y entre una `ColaDoble` y una `Pila`?

Ejercicio 14

Una agencia de quiniela quiere armar un sistema estadístico con los resultados de los sorteos. De cada sorteo sabemos la fecha, el nombre de qué quiniela es (p.ej. 'Nacional' o 'Santiago del Estero') y la lista de los 10 primeros números sorteados. Tomamos las últimas dos cifras, números del 0 al 99. P.ej. en un sorteo salió primero el 42, segundo el 14, tercero el 69, etc. … así hasta el 10. Los números se pueden repetir, p.ej. en un mismo sorteo puede salir tercero el 33 y en el puesto 8 otra vez el 33.

Modelar sorteos y agencia (de la agencia nos interesa el nombre, el nombre del dueño, y el histórico de sorteos que tiene registrados) de forma tal que se pueda

a) saber si en un sorteo un determinado número se repitió, o sea, salió más de una vez.
b) saber si entre todos los sorteos de una fecha que tiene registrados una agencia, en alguno salió un determinado número (p.ej. El 15 de abril de 2010 ¿salió el 29?).
c) saber si entre dos fechas hubo (entre los sorteos que tiene registrados una agencia) algún sorteo maldito, o sea que haya salido el 17 en primer lugar, o los números que salieron en 1ro, 2do y 3er lugar son consecutivos (p.ej. 21, 22 y 23).
d) obtener los sorteos de una determinada quiniela (entre los que tiene registrados una agencia) en los que salió un determinado número a la cabeza (p.ej. dame los sorteos de Córdoba en los que haya salido el 25 a la cabeza) ordenados por fecha. "A la cabeza" quiere decir como primer número sorteado.
e) Obtener los nombres de las quinielas para los que una determinada agencia haya registrado un sorteo para una fecha (p.ej. ¿de qué quinielas tenés sorteos del 25 de abril?).

f) Trular un sorteo, eso quiere decir sumarle 3 a cada número sorteado, si se pasa de 99 dejarlo en 00.
g) Trular todos los sorteos de una fecha registrados en una determinada agencia.

Ejercicio 15 (Opcional)

Una cola con prioridades es una cola en la que los elementos se agregan indicando un valor de prioridad.
Los elementos en una cola con prioridades se recuperan según la prioridad con la que fueron agregados: al hacer un pop en una cola con prioridad se recupera siempre el elemento de mayor prioridad. En una cola podría haber varios elementos con la misma prioridad, en cuyo caso se recuperarán uno a uno sin establecer ningún orden entre ellos.
La iteración en colas con prioridades se lleva a cabo comenzando por los elementos agregados con prioridad más alta. Uno (1) es la máxima prioridad.
Implemente y documente las clases necesarias para que puedan utilizarse colas con prioridades, agregando elementos, recuperando, iterando, etc.