CC41A Lenguajes de Programación - Control 2

Profesor: Luis Mateu

Pregunta 1 (60%)

Smalltalk incluye la clase predefinida OrderedCollection. Esta clase es similar a la clase ArrayList de Java y permite administrar colecciones indexables de objetos. Entre las operaciones más destacables se encuentran:

L:= OrderedCollection new	Crea una colección ordenada
L add: 'hola' beforeIndex: 1 L add: 'tal' beforeIndex: 2	Agrega el objeto 'hola' en la posición 1, luego 'tal' en la posición 2
L add: 'que' beforeIndex: 2	Agrega 'que' en la posición 2, desplazando 'tal' a la posición 3
L size	Entrega el tamaño (i.e. 3)
L at: 2	Obtiene el objeto que se encuentra en la posición 2 (i.e. 'que')
L removeAt: 2	Elimina el objeto que se encuentra en la posición 2, desplazando en una posición los que se encuentran más atrás (i.e. 'tal' pasa a la posición 2)

Se desea construir las clases Queue, AscQuickSorter y DesQuickSorter que posea al menos los siguientes métodos:

Q:= Queue new	Crea una cola fifo
Q put: 'perro' ; put: 'gato'	Agrega 'perro' y 'gato' al final de la cola, en ese orden
Q get	Extrae y entrega el elemento más antiguo en la cola (i.e. 'perro')
Q isEmpty	Entrega verdadero o falso según la cola esté vacía o no
S1:= AscQuickSorter new S1 sort: Q	Ordena ascendente la cola Q usando quick-sort. Si la cola fuese 6, 2, 7 y 3, Q quedaría como 2, 3, 6 y 7
S2:= DesQuickSorter new S2 sort: Q	Ordena descendentemente la cola Q usando quick-sort. En este caso la cola quedaría como 7, 6, 3, 2

a) Implemente la clase Queue a partir de OrderedCollection usando la estrategia de uso.

b) Implemente la clase Queue extendiendo a partir de OrderedCollection.

c) Diseñe e implemente una jerarquía de clases que incluya AscQuickSorter y DesQuickSorter y que permita ordenar usando quick-sort definiendo una sola vez el método sort:. Para particionar cree dos colas auxiliares, una para los menores que el pivote y otra para los mayores o iguales.

(Hint: x < y sirve para compar objetos. [x == 0] whileFalse: [Transcript show: x] para ciclos.)

Pregunta 2 (40%)

Resuelva en Scheme puramente funcional los siguientes problemas:

a) El procedimiento rotar. Ejemplos:

```
(rotar '(1 2 3))          => (3 1 2)
(rotar '(a b c d e f))    => (f a b c d e)
(rotar '(z))              => (z)
(rotar '( ))              => ( )
```

b) El procedimiento primeros, *utilizando el procedimiento map*. Ejemplos:

```
(primeros '( ( a  b  c) (1  2  3) ) )                    => (a 1)
(primeros '( ("hola") ("que" "tal") (4.2 6.7 45.0) ) )   => ("hola" "que"  4.2)
```