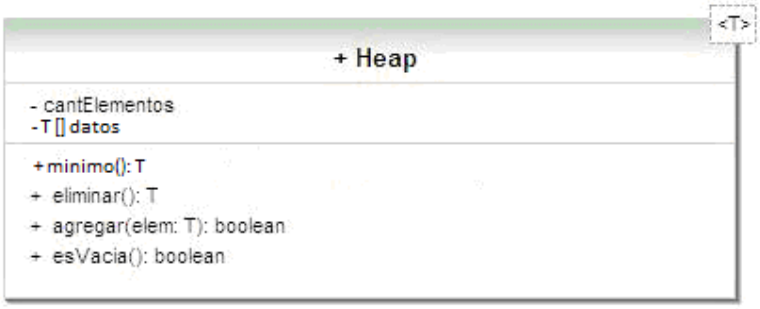
UNLP. Facultad de Informática.
Algoritmos y Estructuras de Datos **Trabajo Práctico 7**
Cursada 2011

Heap

Objetivos
- Representar e implementar las operaciones de la abstracción HEAP.
- Describir soluciones utilizando HEAP.

Ejercicio 1
Muestre las transformaciones que sufre una **Max HEAP** (inicialmente vacía) al realizar las siguientes operaciones:
a) Insertar *50, 52, 41, 54, 46*
b) Eliminar tres elementos
c) Insertar *45, 48, 55, 43*
d) Eliminar tres elementos

Ejercicio 2
Considere la siguiente especificación de la clase Heap, ordenando los elementos con criterio de MinHeap:



El constructor **Heap()** // inicializa una Heap vacía.
El constructor **Heap(ListaGenerica<Comparable<T>> lista)** // crea una Heap a partir de los elementos de una lista. (Se debe utilizar la implementacion de **ListaGenerica<T>** utilizada en la practica 2)
El constructor **Heap(T [] datos)** // crea e inicializa una Heap a partir de reordenar los elementos del arreglo de objetos comparable que recibe como parámetro.
El método **esVacia():boolean** // devuelve true si no hay elementos para extraer.
El método **agregar(T elem) :boolean** // agrega un elemento en la heap y devuelve true si la inserción fue exitosa y false en caso contrario.
El método **eliminar() : T** // elimina el tope de la heap y lo retorna.
El método **minimo() : T** // retorna el tope de la heap.

a) A partir de la especificación anterior implemente la Clase Heap.
Recordar que los arreglos en Java son Zero-Based, con lo cual la primera posición siempre es 0, y es conveniente poner al primer elemento a partir de la posición 1 del arreglo, dejando esa posición para hacer intercambios o simplemente no usarla.
También tener en cuenta que la estructura a utilizar internamente para almacenar los elementos, tiene que tener un mecanismo de comparación entre objetos para poder evaluar las claves, por lo que se recomienda utilizar una estructura que almacene Comparable<T>

b) Que cambios debería hacer sobre la definicion y/o implementación anterior, si los elementos estuvieran ordenados con el criterio de MaxHeap?

c) Calcular el tiempo de ejecución de los constructores: **Heap(ListaGenerica<Comparable<T>> lista)** usado para inicializar una Heap a partir de una lista y **Heap(T extends Comparable<T> [] datos)** usado para inicializar una Heap a partir de un arreglo.
Calcule el tiempo teniendo en cuenta que los datos están ordenados:
(i) en forma creciente
(ii) en forma decreciente.

Ejercicio 3
Se cuenta con una cola de prioridades en donde se almacenan los procesos que se ejecutan en un sistema operativo multitarea. En un determinado momento uno de los procesos comienza a consumir muchos recursos, por lo que el sistema operativo decide bajarle la prioridad. Implemente un algoritmo eficiente para bajar la prioridad del proceso, asumiendo que conoce la ubicación del proceso dentro de la cola de prioridades

Ejercicio 4
Se desea implementar una clase Impresora que ofrezca la funcionalidad dada por la siguiente interfaz:

```
public interface DispositivoImpresion {
        // Almacena un nuevo documento en el dispositivo
        void nuevoDocumento(Documento d);

        // Imprime (saca por pantalla) el documento almacenado más corto //y lo elimina de memoria. Si no hay
        documentos en espera //devuelve false
        boolean imprimir();
}
```

La idea de imprimir el documento más corto en primer lugar es para evitar que un trabajo largo bloquee durante mucho tiempo al resto. Se necesita:

a) Modelar la clase Documento, que consiste simplemente en una cadena de texto (String)
b) Modelar la clase Impresora que implemente la interfaz DispositivoImpresion

Ejercicio 5
Extienda la clase Heap:

a) Agregue un método denominado **public Heap unionCon(Heap OtraHeap)** que realice la unión de la heap que recibe el mensaje con la heap que viene por parámetro y almacene el resultado en la heap que recibe el mensaje.

b) Agregue un método denominado **public Heap intersectarCon(Heap OtraHeap)** que realice la intersección de la heap que recibe el mensaje con la heap que viene por parámetro y almacene el resultado en la heap que recibe el mensaje.