UNLP. Facultad de Informática.

Algoritmos y Estructuras de Datos **Trabajo Práctico 7**

Cursada 2015

Cola de Prioridades - HEAPs

Ejercicio 1

Muestre las transformaciones que sufre una **Max HEAP** (inicialmente vacía) al realizar las siguientes operaciones:

a) Insertar 50, 52, 41, 54, 46

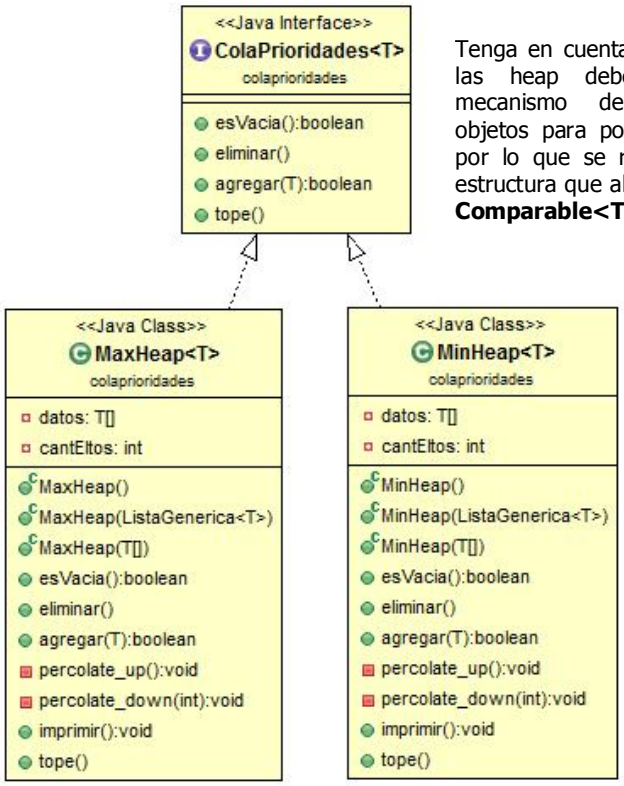
b) Eliminar tres elementos

c) Insertar 45, 48, 55, 43

d) Eliminar tres elementos

Ejercicio 2

Considere la siguiente especificación de la interface **ColaPrioridades<T>** y las clases **MaxHeap()** y **MinHeap()**:



Tenga en cuenta que los elementos de las heap deben responder a un mecanismo de comparación entre objetos para poder evaluar las claves, por lo que se recomienda utilizar una estructura que almacene objetos de tipo **Comparable<T>**

a) Implemente la clase **MinHeap<T>** de la siguiente manera:

El constructor **MinHeap()** inicializa una Heap vacía.

El constructor **MinHeap(ListaGenerica<T> lista)** crea una Heap a partir de los elementos de una lista. (Se debe utilizar la implementación de ListaGenerica<T> utilizada en la práctica 3).

El constructor **MinHeap(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 y lo retorna.

El método **tope() : T** retorna el tope de la heap.

Recordar que en JAVA los arreglos comienzan en 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.

b) ¿Qué cambios debería hacer sobre la definición y/o implementación anterior, si los elementos estuvieran ordenados con el criterio de MaxHeap<T>?

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 para bajar la prioridad del proceso, asumiendo que conoce la ubicación del proceso dentro de la cola de prioridades y esta última está implementada en una **MaxHeap**.

Ejercicio 4

Se desea implementar una clase **Impresora** que ofrezca la siguiente funcionalidad:

```
public class Impresora {
  // Almacena un nuevo documento en el dispositivo
  public void nuevoDocumento(Documento s) {
  }
  // Si hay documentos para imprimir, imprimir el más corto y retorna true, de lo
  // contrario devolver false
  public 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 debe contener una cadena de texto (Strings).

b) Complete la clase **Impresora** y pruebe su funcionamiento.

Ejercicio 5

Extienda la clase **MinHeap**:

a) Agregue un método denominado **public void 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 void 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.

Anexo Ejercicios de Parciales

Ejercicio 1

En los últimos años, los huracanes y tsunamis han mostrado el poder destructivo del agua. Ese poder devastador no se limitó únicamente a los mares y océanos, sino que las fuertes lluvias han causado inundaciones, destruyendo muchísimas casas y campos con la consecuencia inevitable de muertes y enfermedades además de pérdidas de fauna y flora silvestre.
Un grupo internacional de prestigiosos científicos está utilizando sofisticados y complejos modelos tratando de predecir en qué lugares la caída de agua se acumulará de manera muy significativa como resultado de importantes lluvias.
Este grupo de acuerdo a estadísticas realizadas detectó cuáles son las ciudades consideradas de alto riesgo, sumando un total de 150000. Una de las entradas a este modelo es la cantidad de milímetros de lluvia caída de este grupo de ciudades. Esta información está almacenada en una lista.
Los científicos necesitan encontrar la ciudad con el k-ésimo elemento mayor, sólo se pueden almacenar en memoria k elementos simultáneamente. Resolver el ejercicio de forma tal que la cantidad de comparaciones entre elementos sea mínima.

Ejercicio 2

Implemente el método **int extract()** en la clase **MaxHeap** para que no devuelva valores repetidos. Por ejemplo, si la Heap contiene el siguiente conjunto: 10, 10, 9, 8, 8, 7, 7, 7, 5, 4, 2, 2, la primera vez, el método extract() devuelve 10, la segunda vez devuelve 9, y las posteriores invocaciones devuelven 8, 7, 5, 4, 2.

Ejercicio 3

Un sistema operativo administra sus procesos por prioridades, es decir, de un conjunto de procesos selecciona el de mayor prioridad para ejecutar. Defina una clase ConjuntoDeProcesos e implemente los métodos : **agregarProceso (String id, int prioridad)**, **String retornarMayorPrioridad ()**, **incrementarPrioridad (String id)**. Tenga en cuenta que las operaciones deben ser lo mas eficientes posibles.