EDA (ETS de Ingeniería Informática). Curso 2015-2016

Práctica 1. Elementos para el diseño y uso de una EDA en Java

Sesión 2: Aplicación del criterio de eficiencia en el diseño y uso de una Cola de Prioridad

Departamento de Sistemas Informáticos y Computación. Universitat Politècnica de València

1. Objetivos de la práctica

Como ya se comentó, el principal objetivo de la práctica 1 es que el alumno aplique al diseño de una aplicación concreta los conceptos Java para Estructuras de Datos (EDAs) estudiados en el Tema 1 de la asignatura. Así, en la segunda sesión de esta práctica el alumno deberá ser capaz de aplicar el criterio de eficiencia tanto al proceso de diseño de las Implementaciones Java de una *Cola de Prioridad* como al de elección de la mejor de éstas cara a su reutilización en la aplicación de simulación del funcionamiento de un hospital.

2. Descripción del problema

Se dispone de las siguientes implementaciones del modelo Java `ColaPrioridad`: `LPIColaPrioridad` y `PriorityQColaPrioridad`, que usan como soporte de datos en memoria, respectivamente, una *Lista con PI ordenada* y una *java.util.PriorityQueue*. Evidentemente, la implementación que debe usar la clase `Urgencias` de la aplicación de gestión de un hospital debe ser la más eficiente de ellas ... Pero, ¿cómo saber cuál es?

Aunque el análisis teórico, o *a-priori*, del coste Temporal de los métodos de estas clases puede dar una medida del tiempo que emplearán para ejecutarse independiente del entorno de programación donde lo hagan, es obvio que siempre se pueden realizar implementaciones ineficientes de ellos; así, por ejemplo, si los métodos de `ListaConPI` implementados en la clase `LEGListaConPI` no se ejecutan en un tiempo acotado superior e inferiormente por una constante ($\Theta(1)$), el coste temporal efectivo de los métodos `insertar`, `recuperarMin` y `eliminarMin` implementados en `LPIColaPrioridad` no se ajustará al previsto en promedio por su análisis *a-priori*: lineal con la talla de la *Lista* en el caso de `insertar` e independiente de ella tanto para `recuperarMin` como para `eliminarMin`.

Además, la respuesta a la cuestión planteada se complica porque `PriorityQColaPrioridad` se ha implementado reutilizando la clase `PriorityQueue` del estándar de Java, cuyos detalles de implementación y coste asociado se desconocen por el momento.

Por tanto, decidir cuál de las dos implementaciones disponibles de `ColaPrioridad` se debe reutilizar en la clase `Urgencias`, bien `LPIColaPrioridad` o bien `PriorityQColaPrioridad`, pasa por:

- diseñar la clase `LEGListaConPI` como una implementación eficiente de `ListaConPI`, esto es, que garantice un coste Temporal promedio $\Theta(1)$ para cada uno de sus métodos;
- realizar un análisis experimental, o *a-posteriori*, del coste temporal promedio de sus métodos; ello permitirá, a su vez, validar el coste teórico de los métodos de `LPIColaPrioridad` y conocer la eficiencia de los métodos de `PriorityQColaPrioridad` y, consecuentemente, la de `PriorityQueue`.

3. Actividades en el laboratorio

Para el desarrollo de esta segunda sesión de la práctica 1 se proponen las siguientes actividades:

3.1. Diseño de una Implementación eficiente de `ListaConPI`: la clase `LEGListaConPI`

En el documento *Implementación de Lista con PI* disponible en *PoliformaT* se proporcionan las claves para el diseño eficiente de una Implementación Enlazada de `ListaConPI`. Su lectura comprensiva permitirá completar la clase `LEGListaConPI`, también disponible en *PoliformaT* y que se deberá añadir al paquete *librerias.estructurasDeDatos.lineales*.

Tras conseguir compilar la clase `LEGListaConPI`, el alumno tiene que comprobar si el código que ha escrito en esta actividad funciona correctamente ejecutando el programa `PruebaLEGLPI` en el paquete *lineales* de su proyecto *BlueJ eda*.

Para llevar a cabo esta actividad sin mayores problemas, es aconsejable...

- Recordar la forma de añadir ficheros `.class` a un paquete *BlueJ* cuando se vaya a descargar de *Poliformat* el fichero `PruebaLEGLPI.class`.
- Comprobar que el programa `TestUrgencias` (del paquete *aplicaciones.hospital*) sigue funcionando correctamente cuando usa la clase `LEGListaConPI` desarrollada en esta actividad (aunque haya pasado sin problemas el test de corrección).

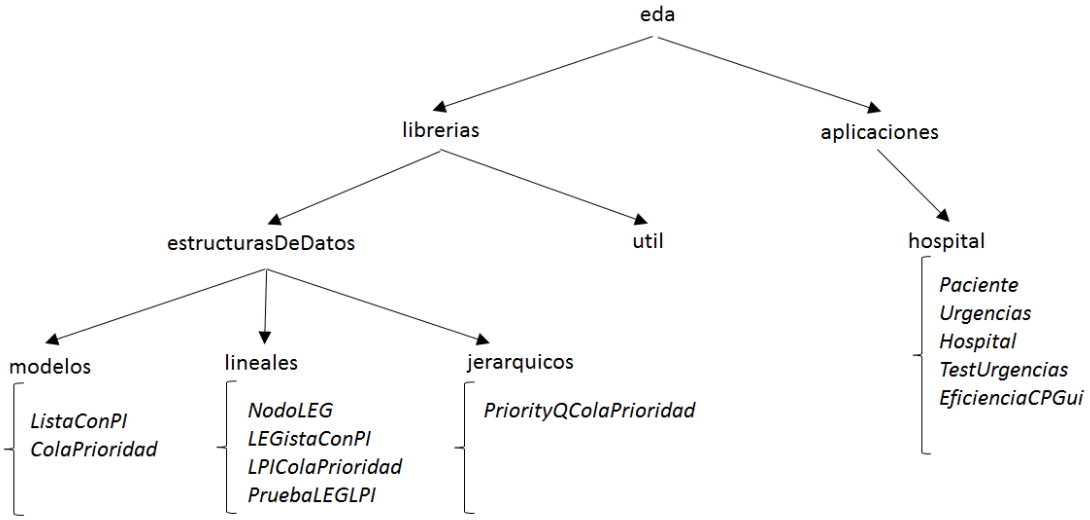
3.2. Análisis y comparación de la eficiencia de los métodos de `LPIColaPrioridad` y `PriorityQColaPrioridad`

En *PoliformaT* se encuentra disponible una aplicación (`EficienciaCPGUI`) que, en modo gráfico, permite obtener y comparar el coste Temporal promedio de los métodos `insertar` y `eliminarMin` implementados en las clases `LPIColaPrioridad` y `PriorityQColaPrioridad`.

Tras descargarla y compilarla en el paquete *BlueJ aplicaciones.hospital*, el alumno debe ejecutar esta aplicación y, en función de lo que observe, debe decidir cuál de las dos implementaciones de `ColaPrioridad` realizadas es la mejor en términos de eficiencia y, por tanto, la que debe reutilizar en la clase `Urgencias` de la aplicación.

Para llevar a cabo esta actividad sin mayores problemas, es aconsejable...

- Comprobar que, antes de ejecutar el programa `EficienciaCPGUI`, la estructura de paquetes y ficheros del proyecto *BlueJ eda* es la que muestra la siguiente figura:



- Recordar que los resultados del estudio experimental del coste Temporal de los métodos de `LPIColaPrioridad` refrendarán los proporcionados por su análisis teórico **si y solo si** el código desarrollado en la actividad `3.1` es eficiente.