EDA (ETS de Ingeniería Informática). Curso 2020-2021

Práctica 5. Implementación de la Cola de Prioridad de un Servidor de Impresión mediante un Montículo (una sesión)

Departamento de Sistemas Informáticos y Computación. Universitat Politècnica de València

1. Objetivos

El principal objetivo de esta práctica es que el alumno aplique al diseño de una aplicación concreta los conceptos sobre Cola de Prioridad y Montículo Binario que ha estudiado en el Tema 5 de la asignatura. Específicamente, al concluir esta práctica el alumno deberá ser capaz de...

- Diseñar una clase Java que represente un Montículo Binario con Raíz en 0, una implementación del Modelo Cola de Prioridad que emplean un gran número de aplicaciones en la vida real.
- Implementar la Cola de Prioridad que usa una aplicación de Simulación de un Servidor de Impresión eficiente.

2. Descripción del problema

Un servidor de impresión es, como su nombre indica, un servidor que conecta (al menos) una impresora a red para que cualquiera de sus clientes pueda imprimir sus trabajos en ella. Como en un momento dado pueden haber varios trabajos a imprimir y, obviamente, una impresora solo puede imprimir uno a la vez, este tipo de servidor debe poder almacenar y gestionar los trabajos en espera de impresión.

El modelo de gestión más sencillo que puede usar un servidor de impresión es el FIFO (*First In, First Out*), i.e. el modelo de una Cola: una Cola almacena, en orden de llegada, los trabajos a imprimir, de forma que el primero de la Cola (*First In*) también es el primero que se elimina de esta para ser impreso (*First Out*), una vez la impresora quede libre. Ahora bien, esta gestión FIFO presenta un problema de eficiencia cuando, por ejemplo, los trabajos que han llegado antes al servidor tienen muchas más páginas que los que han llegado más tarde: ¡la cola puede hacerse eterna! Afortunadamente, para mejorar significativamente el tiempo medio de espera de los trabajos en el servidor basta con hacer lo siguiente:

a. Asignar a los trabajos a imprimir una prioridad que sea inversamente proporcional a su número de páginas.

b. Gestionar los trabajos en espera de impresión que almacena el servidor con una Cola de Prioridad implementada con un Montículo Binario.

Para comprobarlo, en esta práctica se concluirá la implementación de un simulador de los dos tipos de servidores de impresión descritos, que denominaremos *Servidor Cola* y *Servidor Cola de Prioridad*, y se comparará el tiempo medio que espera un trabajo a imprimir en cada uno de ellos.

2.1. Las clases de una aplicación de simulación de un servidor de impresión

Las principales clases que componen una aplicación de simulación de un servidor de impresión son:

- `Trabajo`, que representa un trabajo a imprimir (*print job*). Un trabajo `TIENE UN` título que lo identifica, un cierto número de páginas y el instante de tiempo(en segundos) en el que "entra" (es enviado) al servidor de impresión. Conviene resaltar, además, que el método `compareTo` de esta clase permite establecer cuál de dos trabajos a imprimir dados debe ser atendido con mayor prioridad: aquel cuyo número de páginas sea menor.
- `ServidorDeImpresion` que, como su nombre indica, representa un servidor de impresión (*print server*). Para ser más exactos, dado que un servidor de impresión puede gestionar de formas distintas los trabajos

a la espera de ser impresos que almacena, esta clase es una interfaz Java que establece las operaciones que realiza un servidor de impresión; a saber:

- `insertar(Trabajo t)`, que añade un nuevo trabajo `t` al servidor;
- `hayTrabajos()`, que comprueba si aún quedan trabajos en el servidor;
- `getTrabajo()`, que devuelve el trabajo del servidor que va a ser impreso;
- `imprimirTrabajo()`, que elimina del servidor el trabajo que va a ser impreso y devuelve los segundos que este tardará en imprimirse, en base a la velocidad de la impresora.

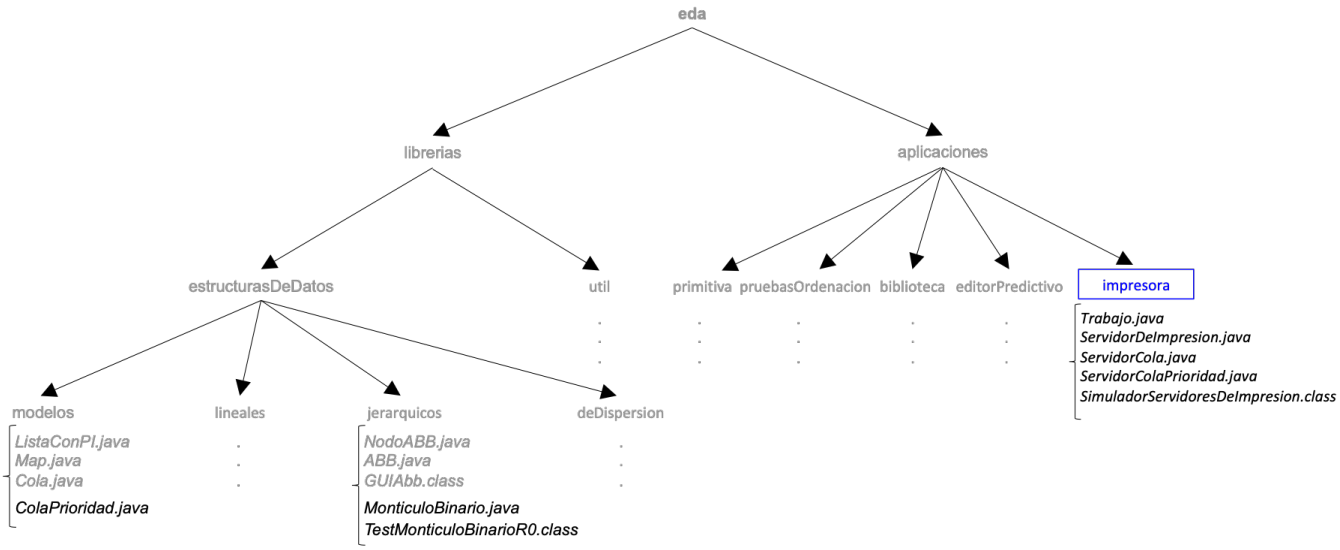
Además, en esta interfaz se define la velocidad de la impresora asociada a un servidor de impresión, i.e. el número de páginas por minuto que puede imprimir (`PAGINAS_POR_MINUTO`).

- `ServidorCola`, que representa un *Servidor Cola*. Así, implementa la interfaz `ServidorDeImpresion` y `TIENE UNA Cola<Trabajo> c` como único atributo.
- `ServidorColaPrioridad`, que representa un *Servidor Cola de Prioridad*. Por ello, esta clase implementa la interfaz `ServidorDeImpresion` y `TIENE UNA ColaPrioridad<Trabajo> cP` como único atributo.
- `SimuladorServidoresDeImpresion`, un programa Java que simula el funcionamiento de dos servidores de impresión, un *Servidor Cola* y un *Servidor Cola de Prioridad*, para calcular el tiempo medio que espera a ser impreso un trabajo en cada uno de ellos; en base a dicho valor se determinará cuál de los dos servidores es el más eficiente.

3. Actividades

Antes de llevar a cabo las actividades que se proponen en este apartado, es necesario que el alumno actualice la estructura de paquetes y ficheros de su proyecto *Bluej eda* siguiendo los pasos que se indican a continuación.

- Entrar en el proyecto *BlueJ eda*.
- Abrir el paquete *aplicaciones* y crear en él un nuevo paquete de nombre *impresora*, que contendrá las clases de la aplicación a desarrollar en esta práctica.
- Salir de *BlueJ* **seleccionando la opción *Salir* de la pestaña *Proyecto***.
- Descargar los ficheros (`.java`) disponibles en *PoliformaT* en sus correspondientes directorios, tal y como muestra la siguiente figura:



- Entrar en el proyecto *BlueJ eda* y compilar la clase `ColaPrioridad`, situada en el paquete *modelos* de *librerias.estructurasDeDatos*.
- Salir de *BlueJ* **seleccionando la opción *Salir* de la pestaña *Proyecto***.
- Entrar en el proyecto *BlueJ eda* y situarse en su paquete *librerias.estructurasDeDatos.jerarquicos*.

3.1. Implementación de un Montículo Binario con Raíz en 0

La clase `MonticuloBinario` vista en teoría representa un Montículo Binario con Raíz en 1 simplemente porque, así, se simplifica el cálculo de la posición del padre y los hijos de su i-ésimo nodo (por Niveles).

Sin embargo, la implementación de Montículo Binario que usan la mayoría de las aplicaciones en la vida real es con Raíz en 0, quizás porque están escritas en lenguajes en los que la primera posición de un array es la cero (como C, Java o Python). Por este motivo, principalmente, en esta actividad el alumno debe implementar en el paquete *jerarquicos* una nueva clase, de nombre `MonticuloBinarioR0`, que represente un Montículo Binario con Raíz en 0; para ello, puede modificar el código de la clase `MonticuloBinario` (disponible en el mismo paquete) para que su `i`-ésimo nodo (por Niveles) pase a tener...

- su hijo izquierdo en la posición $2 * i + 1$, si $2 * i + 1 < talla$;
- su hijo derecho en la posición $2 * i + 2$, si $2 * i + 2 < talla$;
- su padre en la posición $(i - 1)/2$, si $i \neq 0$.

Para validar la clase `MonticuloBinarioR0`, basta ejecutar el programa `TestMonticuloBinarioR0`, disponible en el mismo paquete.

3.2. Ejecución de la aplicación de simulación de un servidor de impresión

Antes de poder ejecutar la aplicación de simulación, ubicada en el paquete *aplicaciones/impresora*, el alumno debe completar el código de las clases `Trabajo` y `ServidorColaPrioridad`; para ello debe tener en cuenta lo siguiente:

- Un *Servidor Cola de Prioridad* almacena y gestiona elementos de tipo `Trabajo`.
- Un *Servidor Cola de Prioridad* solo difiere de un *Servidor Cola* en el modelo que usa para gestionar los trabajos a la espera de ser impresos, por lo que el código de las clases `ServidorColaPrioridad` y `ServidorCola` es muy similar.
- El número de páginas de un trabajo establece su prioridad.
- La clase `MonticuloBinarioR0` es una Implementación eficiente del modelo `ColaPrioridad`.

Hecho esto, la aplicación se puede validar ejecutando el programa `SimuladorServidoresDeImpresion`: si el código de las clases `Trabajo` y `ServidorColaPrioridad` es correcto aparece en pantalla una pequeña simulación donde se compara el tiempo medio que espera a ser impreso un trabajo en un *Servidor Cola* y en un *Servidor Cola de Prioridad*. En esta simulación se muestra una línea por cada uno de los trabajos, en el orden en el que se van imprimiendo, y con el formato que se detalla a continuación:

```
[175] El nombre del viento - P. Rothfuss (55 pag.) Entra al servidor: 59 (6 seg. de espera)
```

- `[175]` $\longrightarrow$ Instante de tiempo (en segundos) en el que finaliza la impresión del trabajo.
- `El nombre del viento - P. Rothfuss` $\longrightarrow$ Título del trabajo.
- `(55 pag.)` $\longrightarrow$ Número de páginas del trabajo.
- `Entra al servidor: 59` $\longrightarrow$ Instante de tiempo (en segundos) en el que el trabajo "entra" (es enviado) al servidor de impresión.
- `(6 seg. de espera)` $\longrightarrow$ Tiempo que el trabajo espera en el servidor de impresión.

NOTA: cada vez que se ejecuta el programa `SimuladorServidoresDeImpresion` se genera de forma aleatoria un conjunto de trabajos y, por tanto, la simulación que aparece en pantalla es siempre distinta de las anteriores.