EDA (ETS de Ingeniería Informática). Curso 2016-2017

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

Departamento de Sistemas Informáticos y Computación. UPV

# 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, la implementación del modelo Cola de Prioridad que se emplea en la práctica.
- Implementar la Cola de Prioridad que usa una aplicación de simulación de un Servidor de Impresión eficiente.

# 2. Introducción y planteamiento 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*): 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 (en segundos) en el que se envía 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 en espera 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 en espera `t` al servidor;
  - `hayTrabajos()`, que comprueba si aún quedan trabajos en espera en el servidor;
  - `getTrabajo()`, que devuelve el trabajo en espera del servidor que va a ser impreso;
  - `imprimirTrabajo()`, que elimina el trabajo en espera del servidor 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, o número de páginas por minuto que puede imprimir.
- `ServidorCola`, que representa un servidor de impresión que usa una Cola para gestionar los trabajos que almacena. Por ello, esta clase implementa la interfaz `ServidorDeImpresion` y `TIENE UNA Cola<Trabajo>c` como único atributo.
- `ServidorColaPrioridad`, que representa un servidor de impresión que usa una Cola de Prioridad para gestionar los trabajos que almacena. Así, 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 y, con ello, calcular el tiempo medio de espera que requiere la impresion de un trabajo en cada uno de ellos; en base a dicho valor se determinara cuál de los dos servidores es el más eficiente.

# 3. Actividades

Para realizar las distintas actividades que se describen en esta sección existen varios ficheros disponibles en *PoliformaT*:

- Los ficheros `MonticuloBinario.java` y `TestMonticuloBinarioR0.class`. Estos dos ficheros se deberán ubicar en el paquete *librerias.estructurasDeDatos.jerarquicos* del proyecto *Bluej eda*.
- Los ficheros `SimuladorServidoresDeImpresion.class` y los `.java Trabajo`, `ServidorDeImpresion`, `ServidorCola` y `ServidorColaPrioridad`; todos ellos deberán ser ubicados en un nuevo subpaquete de *aplicaciones* denominado *impresora*.

## 3.1. Actividad A: 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, a la hora de desarrollar una aplicación siempre resulta preferible usar una implementación de un Montículo Binario con Raíz en 0 porque evita dejar vacía la posición 0 de `elArray`.

En consecuencia, en esta actividad el alumno debe implementar en el paquete *librerias/estructurasDeDatos/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 (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. Actividad B: Ejecutar 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 en espera de impresión, 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 de espera que requiere la impresión de un conjunto de trabajos usando un Servidor Cola y 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:

```
[156] El hobbit - J.R.R. Tolkien (61 pag.) Envio: 34 (122,00 seg. de espera)
```

- `[156]` $\longrightarrow$ Instante de tiempo (en segundos) en el que finaliza la impresión del documento.
- `El hobbit - J.R.R. Tolkien` $\longrightarrow$ Título del documento.
- `(61 pag.)` $\longrightarrow$ Número de páginas del documento.
- `Envio: 34` $\longrightarrow$ Instante (en segundos) en el que se envió el documento a la impresora.
- `(122,00 seg. de espera)` $\longrightarrow$ Tiempo de espera para la impresión del documento. Este valor es la diferencia entre el tiempo en el que acaba la impresión y el instante de envío.

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