

## PRACTICA 2

### ***Emulación de ejecución de procesos en un sistema monoprocesador***

---

En esta práctica hay que diseñar y construir un programa que simule la ejecución de procesos de un sistema monoprocesador. Se trata de realizar la simulación del funcionamiento de un procesador en la ejecución de tareas y de cómo se pueden gestionar dicha ejecución mediante un planificador (scheduler). Para poder hacer pruebas comprobando el funcionamiento y rendimiento de los distintos algoritmos de planificación la estrategia de planificación debe poder ser reconfigurable en tiempo de ejecución.

- Procesos. Los procesos están identificados unívocamente por un identificador y tienen asignada una determinada prioridad de ejecución. Los procesos están formados por bloques de instrucciones y por bloques de peticiones de entrada salida (E/S). Para un determinado proceso, el número de bloques de cada tipo y el tamaño de cada bloque podrán venir determinados o bien se podrán generar de forma aleatoria dentro de unos pecificados en el programa.
- Planificador. El planificador se encarga de organizar la ejecución de todos los procesos que están preparados para ser ejecutados de acuerdo con un algoritmo de planificación. En una primera aproximación, una vez que un proceso pasa a ser ejecutado se continua hasta que termina, o hasta que solicita realizar entrada/salida (en cuyo caso pasa a una cola de procesos en espera de E/S).
- Algoritmo de planificación. Define el procedimiento por el que se decide cuál es el siguiente proceso de entre los preparados que pasa a ejecutarse. Deben implementarse dos algoritmos de planificación diferentes uno basado en el tiempo de llegada, primero se ejecuta el proceso que primero llega, y otro basado en prioridades, de modo que el siguiente proceso a ser ejecutado será aquel que tenga mayor prioridad (dentro de los procesos

- Procesador de entrada/salida. Gestiona la cola de procesos que están realizando operaciones de entrada salida. La ejecución de cada uno de los bloques, tanto de instrucciones como de peticiones de entrada/salida consistirá en escribir por pantalla los datos del proceso y bloque que se está ejecutando en ese momento.

La simulación se realizará mediante una **agenda** que contiene las tareas que hay que realizar en cada momento de tiempo determinado.

Las operaciones a realizar con la agenda vienen determinadas por el siguiente interfaz Java:

```
public interface Agenda{

    /** añade una tarea a la agenda que debe ejecutarse cuando transcurra el
    tiempo indicado */
    public void añadirAAgenda(int tiempo, TareaEjecutableAgenda tarea);

    /** ejecuta cada una de las tareas de la agenda en secuencia*/
    public void ejecutar();
}
```

La simulación se lanza mediante el método *ejecutar()* de la agenda, que ejecuta cada una de las tareas de la agenda según la secuencia temporal. Es decir ejecuta la primera tarea y la elimina de la agenda, luego pasa a ejecutar la siguiente tarea. En general, según se ejecuta la simulación se irán añadiendo nuevas tareas a la agenda, de modo hasta que no queden tareas en la agenda.

Con el propósito de obtener generalidad todas las tareas que se incorporen a la agenda deben cumplir el siguiente interfaz Java:

```
public interface TareaEjecutableAgenda {
    public void ejecutar(Agenda agenda);
}
```

El programa debe ser sencillo de manejar y debe permitir realizar por lo menos las siguientes operaciones:

- añadir procesos al sistema, tanto especificando su estructura como generados de forma aleatoria
- seleccionar el algoritmo de planificación
- lanzar la ejecución de la simulación
- terminar el programa.

### ***Parte opcional***

Se deja libertad al alumno para incluir nuevos algoritmos de planificación u otras en función de su conocimiento de los sistemas operativos.

### ***Notas de implementación***

En la implementación se puede utilizar cualquier clase estándar incluida en el API de Java 1.3.

### ***Memoria de la práctica***

Se debe entregar una **breve** memoria impresa (no superior a cuatro páginas) con el diseño de la práctica (clases, organización de estas, detalles significativos, ...). Se entregará el disquete con el código del proyecto *CodeWarrior*, correctamente comentado y formateado, así como la documentación generada automáticamente mediante *javadoc* siguiendo las pautas explicadas en clase.

Cada grupo entregará la memoria exclusivamente en el laboratorio y en el turno correspondiente. El plazo de realización y entrega de la memoria es de 3 semanas.