Yael Alfredo Salazar Aguilar

14/12/2023

Documentación del proyecto

Simulador de planificador de procesos FCFS

Contents

[**Objetivo** 2](#_Toc153496476)

[**Contexto** 2](#_Toc153496477)

[**FCFS** 2](#_Toc153496478)

[**Procedimiento** 2](#_Toc153496479)

[**Interfaz Gráfica** 3](#_Toc153496480)

[**Diagrama UML** 4](#_Toc153496481)

# **Objetivo**

Implementar un simulador de planificación de procesos mediante la política FCFS, esto en el Java, además, implementar una interfaz gráfica que pueda dejar en claro cómo funciona el código y dicha política.

# **Contexto**

Para la realización de este proyecto se tuvieron en cuenta algunos factores, primero, se cuenta con solo un procesador de un núcleo, es decir, solo puede entrar un proceso a la vez. Segundo, cada proceso tiene tiempo de procesador y memoria requerida. Tercero, se tiene que hacer uso de hilos e implementar una interfaz gráfica. Después de ello hay libertad para la realización de esta simulación.

# **FCFS**

FCFS es una política de planificación de procesos, esta política consiste en que el primer proceso que llegue a la cola para ser procesado es el primero en ser atendido. Este método tiene ventajas, la principal de ellas es que es sencillo de implementar a comparación de otras políticas, pero el problema es que puede ser no muy eficiente ni práctico, ya que puede haber muchos factores negativos que afecten al rendimiento.

# **Procedimiento**

Para comenzar, se va a explicar cómo funciona el código, para posteriormente explicar la interfaz gráfica. Primero, se tienen 4 clases; SimuladorFCFS, esta actúa como la función Main, en ella se encuentra la creación de la interfaz gráfica, también se tiene la creación de dos hilos y un hilo anidado, el manejo de hilos se hizo por un método llamado lambda, el cual permite la utilización de hilos de manera concisa y más fácil de leer, esto gracias a que se declaran y se proporciona su código de funcionamiento ahí mismo. Ahora, el primer hilo se encarga de la generación de procesos aleatorios, la cual se apoya de la clase “TareaProcesoAleatorio” la cual se explicará más adelante; este hilo crea un nuevo proceso cada 3 segundos, esto gracias a un sleep. El segundo hilo llamado “ejecutorProcesos”, este hilo establece los hilos que pueden ser ejecutados, los cuales son los que no superen la memoria, establecida en 1024 MB, los procesos pueden ser creados de hasta 2000 MB, en este hilo se crea una instancia de la clase “Proceso” a la cual se le asigna el primer valor de la cola de procesos creados, y si la memoria es suficiente para el proceso, éste se agrega a una parte de la interfaz gráfica, la cual muestra los procesos que pueden ser procesados, después se crea un hilo anidado, este hace uso de un semáforo que permite que entre solo 1 hilo a la vez, este hilo hace que el proceso que se está ejecutando actualmente dure procesándose su tiempo asignado en segundos, puede tener un máximo de 10 segundos (puede ser cambiado), y cuando el proceso termina, se agrega a la parte de procesos terminados en la interfaz gráfica, finalmente el semáforo es liberado. Para finalizar con la clase, se crea el main y se utiliza **SwingUtilities.invokeLater(() ->**, éste es utilizado para que una tarea se ejecute de manera segura en el hilo de la interfaz gráfica.

La segunda clase es “ColaProcesos”, en general, esta clase se encarga de la cola de los procesos que son creados y así mismo le asigna un tiempo y memoria requerida de forma aleatoria, esto gracias a la utilización de la librería Random, por último, cada proceso creado y guardado en la cola se muestra en una sección de la interfaz gráfica

También se encuentra la clase “TareaProcesoAleatorio” esta clase solo contiene un método, el cual es ejecutar(), y éste llama a un método de ColaProcesos, esto se manda a llamar en el hilo de crear procesos del programa Main, esto se hizo así para una menor escritura de código.

Finalmente está la clase “Proceso”, ésta permite la creación de instancias para la creación de nuevos procesos, por ello, solo cuenta con los atributos del proceso a crear y el constructor de la clase.

# **Interfaz Gráfica**

Para comenzar a hablar de la interfaz gráfica, primero hay que mencionar cómo se encuentra dividida, ésta cuenta con cuatro apartados, el primero es el apartado de los procesos en creación, en éste se escriben todos los procesos que se van creando, esto con su nombre, su tiempo y su memoria requerida.

En el segundo apartado se encuentran los procesos ejecutándose, estos son los procesos que sí pueden ser ejecutados, es decir, los que caben dentro de la memoria total.

En el tercer apartado se encuentran los procesos finalizados, cada vez que un proceso termine su tiempo en la cpu, se manda a este lugar.

Por último, se encuentra el apartado de los procesos excluidos, estos son aquellos que superan la capacidad de la memoria total y, por lo tanto, no pueden ser procesados.

La interfaz gráfica se decidió implementar de esta manera para una demostración gráfica de cómo funciona el proceso pasa por paso, dando una representación de cómo funciona realmente un planificador de procesos de un sistema operativo real. Cabe aclarar que puede haber diferencias entre la realidad y esta simulación, debido a que cada sistema lo puede manejar de diferente manera y esto solo fue una pequeña muestra del cómo podría funcionar este planificador de manera gráfica.

A screenshot of a computer

Description automatically generated

# **Diagrama UML**

A diagram of a data flow

Description automatically generated