**Laboratorio #1:**

Todo sistema operativo gestiona los programas mediante el concepto de proceso. En un instante dado, en el ordenador pueden existir varios procesos listos para ejecutarse. Sin embargo, solamente uno de ellos puede ser ejecutado (en cada microprocesador). De ahí la necesidad de que el sistema operativo gestione, mediante el planificador de procesos y de una manera equitativa, qué proceso debe ejecutarse en cada momento, para hacer un uso eficiente del o de los procesadores.

Se desea simular el funcionamiento del planificador de procesos de cierto sistema operativo en un ordenador con un solo procesador y para ello se ha diseñado una estructura de datos que almacena, de acuerdo a su prioridad (de forma descendente), cada uno de los procesos disponibles para ser ejecutados, así, si un nuevo proceso solicita su ejecución, el valor correspondiente a su prioridad será añadido en la estructura de datos mencionada, de forma tal que se mantenga el orden de ejecución. El planificador ejecutará un proceso “A” de menor prioridad que el primero de la lista si y solo si“A”tiene un tiempo de espera mayor que la prioridad del primero de la lista.

Haciendo uso del “cascarón” proporcionado por el profesor:

1. Implemente la funcionalidad **Adicionar(prioridad: entero)**, de la clase **ListaSE**, del paquete **TDA**, la cual ubica, en la lista de ejecución, de acuerdo a su prioridad, cada proceso que solicita ser ejecutado. Luego:
2. implemente el método **runProcess()** de la clase **OperativeSystem**, perteneciente al paquete **Ox**, el cual ejecuta el proceso de mayor prioridad o tiempo de espera según lo descrito en el ultimo párrafo de la introducción.

**Figura 1:** Paquete **Ox**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | OperativeSystem | | # procesos: ListaSE<Process>  # siguiente: Proceso<T> | | + OperativeSystem()  + newProcess(pp: Process)  + runProcess()  + ProcessRequest(): entero  + getProcess(i: entero): Process  + getALLProcess(): ListaSE<Process> | | |  | | --- | | Process | | # id: entero  # prioridad: entero  # waitTime: entero  **... ... ...** | | + Process(Priority: entero, id: entero)  + getPriority(): entero  + getWaitTime(): entero  + execute()  **… … …** | |

* newProcess(pp: Process): ubica un nuevo proceso en la lista de procesos disponibles para ejecutar.
* runProcess(): ejecuta un proceso de acuerdo a la descripción dada en la introducción y lo elimina de la lista.
* ProcessRequest(): entero : devuelve la cantidad de procesos en la lista.
* getProcess(i: entero): Process : devuelve un proceso dado su posición en la lista.
* getPriority(): entero: devuelve la prioridad del proceso.
* getWaitTime(): entero: devuelve el tiempo que lleva esperando el proceso para ser ejecutado.
* execute(): ejecuta el proceso.

**Figura 2:** Paquete TDA

T

T

T

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | <<Interface>>  Lista | |  | | + Longitud(): entero  + Obtener(): T  + Adicionar(x: T)  + Insertar(x: T, pos: entero)  + Eliminar(pos: entero) | | |  | | --- | | NodoSE | | # pdato: T  # siguiente: Proceso<T> | | + Proceso(prioridad: T)  + Proceso(prioridad: T, sgte: Proceso<T>)  + getPrioridad(): T  + setPrioridad(prioridad: T)  + getSgte (): Proceso<T>  + setSgte(sgte: Proceso<T>) | |
| |  | | --- | | ListaSE | | # longitud: entero  # cabeza: NodoSE<T> | | + Adicionar(x: T)  … … … | |  |

* Adicionar(x T): Ubica un elemento en la lista de forma que esta mantenga su orden descendente.