Complejidad Temporal, Estructuras de Datos y Algoritmos

Enunciado Trabajo Final

Consideraciones Generales

El objetivo de este trabajo es integrar los contenidos vistos en la materia. El trabajo deberá realizarse en forma individual.

La fecha límite para su defensa se encuentra publicada en el aula virtual en el enlace: “Plan de trabajo”.

El trabajo se presentará junto con un informe que debe incluir:

- Datos del autor del trabajo final.
- Un diagrama de clases UML describiendo la estructura del sistema, mostrando las clases del sistema, sus atributos, métodos y las relaciones entre los objetos.
- Detalles de implementación, problemas encontrados y como fueron solucionados, condiciones de ejecución, etc.
- Las imágenes de todas las pantallas que componen el sistema codificado junto con una descripción completa de las mismas.
- Ideas o sugerencias para mejorarlo o realizar una versión avanzada del mismo.
- Una breve conclusión o reflexión de la experiencia adquirida a partir de la realización del trabajo final.

Este informe se evaluará tanto en su contenido como en la forma en el que es presentado y tendrá una nota que afectará la nota final del trabajo.

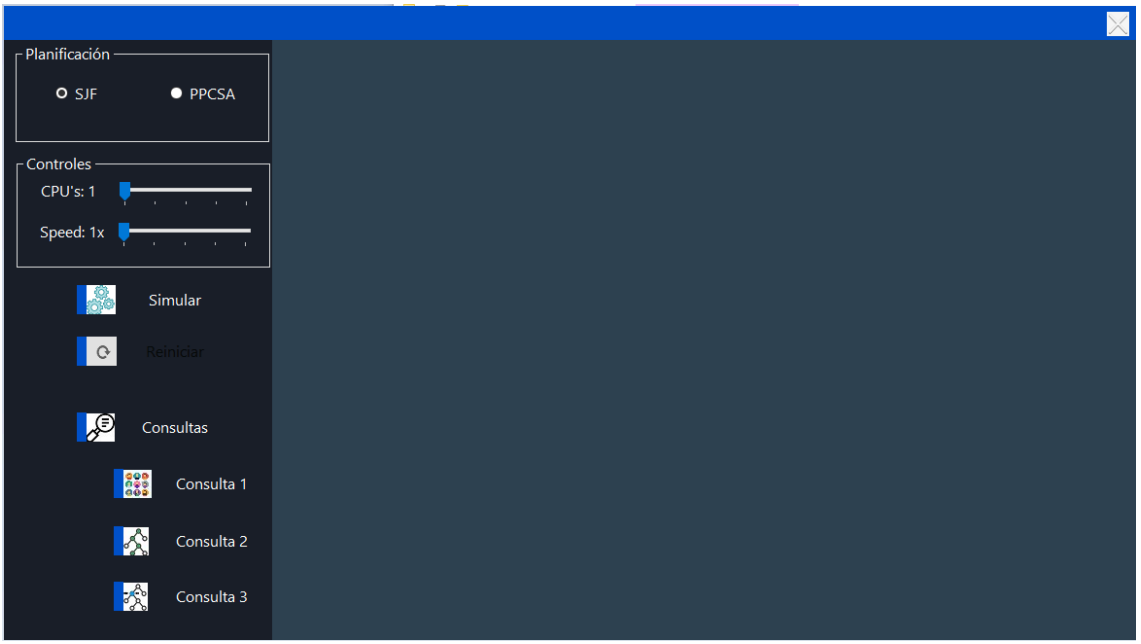
El trabajo deberá defenderse en un coloquio presencial.

Enunciado

Una empresa informática está llevando a cabo un simulador de planificación (*scheduling*) de CPU para un nuevo sistema operativo. El simulador tiene por objetivo tomar conjuntos de datos almacenados en archivos *csv* (*datasets*) y a partir de ellos diseñar el *scheduling.* Los datos almacenados en los *csv* son: el nombre del proceso, tiempo de uso de la CPU y la prioridad del proceso. Con los datos anteriores se simulará la competencia por el acceso a la o las CPU's y se tomará la decisión de cederle la CPU a uno u otro proceso de acuerdo a las siguientes planificaciones:

- El trabajo más corto primero (*Shortest Job First - SJF*) es una planificación que selecciona el proceso en espera que ocupará la CPU el menor tiempo.
- *Preemptive Priority CPU Scheduling Algorithm (PPCSA)* es una planificación basada en la prioridad del proceso, en este caso el proceso con mayor prioridad tomará la CPU primero.

Al iniciar el aplicativo, éste solicita al usuario que seleccione el archivo *csv* donde se encuentran almacenados los datos de los procesos con los cuales se desea trabajar. Luego el aplicativo construirá una lista de objetos *`Proceso`* (*`List<Proceso>`*) con todos los datos tomados del archivo *csv* para luego desplegar la siguiente vista a fin de realizar distintas simulaciones:



Para realizar una simulación es necesario:

1. Seleccionar la planificación que se desea utilizar: *SJF* o *PPCSA*.

2. Indicar la cantidad de procesos ejecutando concurrentemente utilizando la barra deslizante "*CPU's*" y que velocidad tendrán las CPU's a través de la barra deslizante "*Speed*".

3. Activar el botón "*Simular"* para dar comienzo a la simulación.

La empresa informática lo ha contratado a Ud. para completar el aplicativos antes descrito, su función será la de implementar los siguientes métodos pertenecientes a la clase **Estrategia**:

1. **ShortesJobFirst**(List<Proceso> datos, List<Proceso> collected): Retorna en la variable *collected* los procesos ordenados del de menor tiempo de uso de la CPU al de mayor de la lista *datos* utilizando una **MinHeap** como estructura de datos soporte.

2. **PreemptivePriority**(List<Proceso> datos, List<Proceso> collected): Retorna en la variable *collected* los procesos ordenados del de mayor prioridad al de menor prioridad de la lista *datos* utilizando una **MaxHeap** como estructura de datos soporte.

3. **Consulta1** (List<Proceso> datos): Retorna un texto con las hojas de las Heaps utilizadas en los métodos anteriores construidas a partir de los datos de entrada.

4. **Consulta2** (List<Proceso> datos): Retorna un texto con las alturas de las Heaps utilizadas en los métodos anteriores construidas a partir de los datos de entrada.

5. **Consulta3** (List<Proceso> datos): Retorna un texto que contiene los datos de las Heaps utilizadas en los métodos anteriores, explicitando en el texto resultado los niveles en los que se encuentran ubicados cada uno de los datos.

La cátedra proveerá métodos y clases utilitarias a fin de simplificar la construcción y prueba del aplicativo. A continuación, se describen los métodos más importantes y a que clase pertenecen:

Clase	**Métodos**
Proceso	**- Proceso(string nombre, int tiempo, int prioridad)**: Construye un objeto proceso que almacena el nombre, la cantidad de tiempo que necesita para ejecutar y la prioridad del mismo. **- ToString():** Devuelve un texto que contiene los datos almacenados en formato *string* .