2do Proyecto- Simulador de Procesos

Un Sistema Operativo es un complejo software que coordina los múltiples programas que será ejecutados por múltiples usuarios haciendo un efectivo uso de los recursos que tiene el computador.

El desarrollo de este programa debe ser Orientado a Objetos por lo que las diferentes clases que forman parte de este enunciado han de desarrollarse. La información debe de guardarse en un archivo plano con separadores (CSV), JSON o XML

1. Software que desarrollar

Su trabajo consiste en crear una aplicación que simulará la ejecución de procesos de un computador haciendo uso de los conceptos visto relacionado a la orientación de objetos.

Para este proyecto se requiere del desarrollo de una interfaz gráfica donde podrá apreciarse cada una de las ejecuciones de procesos y el uso de la memoria principal.

1.1. Los procesos

Un proceso en este caso se entenderá como un programa que debe ser ejecutado por el sistema operativo, para ello estos programas tendrán las siguientes características:

- Identificador: Es un valor numérico
- Nombre: nombre del programa
- Tipo de programa: Los tipos de programa serán ejecutable, multimedia y documento
- Fecha de ejecución
- Hora de ejecución
- Hora de finalización
- Tamaño: Este será el tamaño del programa, será un valor numérico, entero, es decir sin decimales. Este tamaño lo limitaremos entre 10 y 20 por motivos de no tener que saturar la memoria.
- Duración: Este será la duración del programa en segundos, será un valor numérico, entero, es decir sin decimales. Esta duración lo limitaremos entre 5 y 20 por motivos de no tener una extensa ejecución del programa.
- Usuario: nombre del usuario
- Estado: Tendrá los posibles valores asignados, en espera, finalizado
- CPU: El número de la CPU que fue asignado, entre 1 a 4

A continuación, se detalla atributos especiales dependiendo el tipo de programa:

- Si el programa es de **tipo Ejecutable**, especificar si es un Exe o Bat, además si es **cooperativo**: Si / No.

- Si el programa es de **tipo multimedia**, especificar los **recursos** que necesitará y estos podrían ser video, sonido o ambos.
- Si el programa es de **tipo documento**, especificar **el tipo de formato**, es decir texto plano o cifrado

Para poder simular la ejecución de programa o programas por usuario, es lo estaremos agrupado por **archivo**, es decir cada archivo será un usuario, y el contenido de estos ficheros será la lista de programas que deberán ser ejecutados.

Supongamos que exista el archivo llamado ccambronero.prs, en este caso el nombre del archivo sin la extensión será el **nombre del usuario**, es decir "ccambronero". Tomar en cuenta que solo deberá leer archivos cuya extensión sea prs.

Dentro este archivo tendrá la siguiente estructura:

> Nombreprograma, tamaño, duración, tipoPrograma, AtributoEspecial

Por ejemplo:

Sea el archivo llamado ccambronero.prs y dentro de este está:

> Python.exe, 12, 20, ejecutable, exe, Si
> MiDocumento.doc, 10, 8, documento, cifrado
> Seguridad.bat, 12, 15, ejecutable, bat, No
> JohnWick.mov, 20, 20, multimedia, ambos

Es necesario resaltar que el programa deberá permitir cargar **más de un archivo prs**, antes de ejecutar el simulador.

1.2. Lista de procesos

Cuando se carguen los archivos, este cargará todos los programas que deberá ser ejecutados por nuestro simulador del sistema operativo, la forma de visualizar deberá ser de una tabla para poder observar cada uno de los atributos de los procesos que han sido definidos en la sección anterior (1.1).

- El valor de identificador será un valor numérico consecutivo cuyo valor podría ser entre 256 a 1024, es decir en forma aleatoria el valor inicial deberá se definida, y de ahí en adelante será asignado a cada uno de los procesos a ejecutar en forma consecutiva.
- El valor de fecha será la fecha que será obtenido a través de la fecha del computador.
- La hora de ejecución será la hora en que iniciará a ejecutarse este proceso
- La hora de finalización será la hora en que finaliza su ejecución

Estos procesos deberán ser ejecutados en las CPU que el computador tenga disponible, esto será una cantidad variables desde 1 hasta 4. Esto debe ser configurado antes de cargar los archivos de programas.

1.3. Memoria principal

Este recurso debe ser visible en el programa, consistirá en una lista, iniciando desde la posición 1 hasta un tamaño de 128. Una vez abierto los archivos con los programas a ejecutar, serán ubicados cada uno de ellos hasta que todos estén en memoria o esta esté lleno. Los programas no deberán sobreponerse o traslaparse. Conforme los programas terminen su ejecución este liberará su espacio utilizado en la memoria principal y serán ocupados por aquellos programas que no hayan podido asignarse al principio de la carga de los programas.

1.4. Asignación de CPU

Una vez definido la cantidad de CPU y los programas cargados, se procede a la asignación de estos a su respectiva CPU (Esto puede ser a través de un botón). Esta asignación deberá ser **aleatoria** para cada uno de los procesos a ser ejecutados. Cada CPU podrá ejecutar a la vez hasta 5 procesos, si existen más de 5 procesos asignados a una CPU, deberá estar **en espera** hasta que un proceso finalice y uno de los que está en espera tome su lugar y proceda su ejecución.

1.5. Ejecución de los procesos

Cuando cada uno de los procesos esté asignado a su respectiva CPU, se procede a su ejecución a través de un botón. La ejecución deberá ser visualizado en pantalla. La ejecución será de segundo a segundo y durará según la cantidad asignada en su atributo. Por cada segundo de ejecución, este alternará al siguiente proceso dentro de la misma CPU.

Si existe más de un CPU, este tendrá el mismo comportamiento.

2. Aspectos administrativos

- Trabajo en grupo de 3 personas
- Usar Netbeans
- Entrega en el Github Classroom (https://classroom.github.com/a/YRxW7671), domingo 21 de enero del 2024 hasta las 11 pm.
- Tomar en cuenta toda explicación o aclaración del proyecto por parte del profesor durante la clase, para que el estudiante aclare dudas y no asuma requerimientos equivocado.
- Deberá aplicarse Hilos, herencia, uso de enumeradores y clase abstracta.
- La aplicación debe contar con interfaz gráfica, además de la funcionalidad se evaluará la estética de este.
- Tomar en cuenta las validaciones al formato del archivo y la documentación interna (código)

3. Documentación

La documentación es un aspecto de gran importancia en el desarrollo de programas, especialmente en tareas relacionadas con el mantenimiento de estos. Para la documentación interna, deberán incluir comentarios descriptivos para cada clase y funciones **principales**, con sus entradas, salidas, restricciones y objetivo (**Seguir javadoc**).

La documentación externa deberá incluir:

1. Portada.
2. Manual de usuario: **screenshot de las pantallas desarrolladas** y su explicación.
3. Pruebas de funcionalidad: incluir el enlace de *un video* ***YouTube. Modo público o al menos que sea visible por el profesor***
4. Descripción del problema.
5. Diagrama UML de Clases: imagen en el documento y el url al sitio donde esté el diagrama
6. Diagrama UML de Paquetes: imagen en el documento y el url al sitio donde esté el diagrama.
7. Librerías usadas: creación de archivos, etc.
8. Análisis de resultados: objetivos alcanzados, objetivos no alcanzados, y razones por las cuales no se alcanzaron los objetivos (en caso de haberlos).

4. Evaluación

4.1. Rubro para el programa

Descripción	**Valor**
Documentación interna **(javadoc)**	5%
Documentación externa	15%
Diseño Orientado a Objetos	40%
Lectura de los archivos	5%
Lista de Procesos	5%
Gestión de la Memoria	10%
Asignación a la CPU	5%
Ejecución	15%
TOTAL	**100%**

4.2. Rubros del proyecto en total

Descripción	Valor
Planificación del proyecto	5%
Seguimiento al proyecto (Cumplimiento/Semana) Demostrar al menos 30% de avance. Control de acceso y todas las ventanas con sus eventos.	5%
Video explicativo YouTube *	5%
Funcionalidad del programa	85%
TOTAL	**100%**

- (*) Debe seguir las instrucciones
- **El diseño de la interfaz gráfica tiene un peso del 40%**