Simulador de procesos de sistema operativo

# Análisis del Sistema (30 puntos)

## 1.1 Requisitos funcionales del sistema (5 pts)

Escribir los puntos exactos que debe cumplir tu sistema.

Ejemplo: “El usuario puede crear una nueva solicitud de boletos”, etc.

- El usuario puede crear la cantidad de procesos que quiera, indicando cual es su nombre y el tipo de proceso que le corresponde.

- El usuario puede ver la lista de procesos que se encuentran en espera de ejecución

- El usuario puede indicar que se ejecuten todos los procesos en espera. Algunos de ellos se ejecutan y desaparecen, mientras que otros se mantienen cada vez que se ejecuta la línea de proceso.

- Al final, el usuario puede ver el resultado de cada proceso

- Después de la ejecución, el usuario puede elegir si continuar usando el programa o salir.

## 1.2 Clases necesarias y su propósito (5 pts)

|  |  |
| --- | --- |
| Clase | Propósito |
| Proceso | Esta clase se refiere a cualquier proceso de la computadora. Tiene nombre e identificador, y cuenta con un método de operación. |
| Proceso de CPU | Esta clase se refiere a los procesos del CPU del sistema. Hereda de procesos, y su función principal es mostrar un número aleatorio entre 0 y un numero ingresado. |
| Proceso de I/O | Esta clase se refiere a los procesos de I/O del sistema. Hereda de procesos, y su función principal es pedir al usuario un numero para sumarle a otro previo. |
| Proceso de Daemon | Esta clase se refiere a los Daemons que el sistema usa. Hereda de procesos, y su función principal es dar la diferencia entre un número ingresado y un número elegido aleatoriamente entre 0 y 20. Al finalizar la ejecución de procesos, los Daemons no se eliminan de la lista. |
| Controlador | Esta clase se encarga de llevar el flujo del programa. |
| Consola | Esta clase se encarga de pedir los datos necesarios al usuario |
| Main | Esta es la clase principal, la cual usa al controlador y la consola para lo que necesita. |

## 1.3 Atributos de cada clase (10 pts)

Clase: Proceso

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo | Tipo de dato | Visibilidad | Propósito |
| Nombre | String | Protegida | Es el nombre del proceso |
| Variable de cambio | Int | Protegida | Es la forma en que el proceso cambiará la variable recibida. |
| PID | String | Protegida | Es el identificador propio del proceso |

Clase: Proceso de CPU

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo | Tipo de dato | Visibilidad | Propósito |
| Generador de numeros | Random | Privada | Es un generador de números aleatorios |

Clase: Proceso de I/O

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo | Tipo de dato | Visibilidad | Propósito |
| Ninguno |  |  |  |

Clase: Proceso de Daemon

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo | Tipo de dato | Visibilidad | Propósito |
| Generador de números | Random | Privada | Es un generador de números aleatorios |

Clase: Controlador

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo | Tipo de dato | Visibilidad | Propósito |
| Cuenta de PID | int | privada | Es la cantidad de procesos que se han creado en el sistema |

Clase: Consola

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo | Tipo de dato | Visibilidad | Propósito |
| Teclado | Scanner | Privada | Sirve para comunicarse con el usuario por medio de la terminal, para mostrarle y pedirle datos |

Clase: Main

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo | Tipo de dato | Visibilidad | Propósito |
| Lista de ejecución | ArrayList<Proceso> | Privada | En esta lista se almacenan todos los procesos que el usuario cree, en lo que esperan a ser ejecutados. |

### 1.4 Métodos de cada clase (10 pts)

Clase: Proceso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros :  Tipo de dato | Salida: Tipo de dato | Visibilidad | Propósito |
| operar | * Num: int | int | protected | Este método no se usa, solo es para indicar que la clase de Proceso y sus hijos cuentan con el método de operar |

Clase: Proceso de CPU

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros :  Tipo de dato | Salida: Tipo de dato | Visibilidad | Propósito |
| operar | * Num: int | int | publica | Este método genera un número aleatorio entre 0 y Num, y devuelve dicho número. |

Clase: Proceso de I/O

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros :  Tipo de dato | Salida: Tipo de dato | Visibilidad | Propósito |
| operar | * Num: int | int | publica | Este método le pide al usuario un número por medio de Consola, y regresa la suma de dicho número y Num |

Clase: Proceso de Daemon

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros :  Tipo de dato | Salida: Tipo de dato | Visibilidad | Propósito |
| operar | * Num: int | int | publica | Este método genera un número aleatorio entre 0 y 20, y devuelve la diferencia entre el número ingresado y el número generado. |

Clase: Controlador

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros :  Tipo de dato | Salida: Tipo de dato | Visibilidad | Propósito |
| Crear proceso | * ninguno | Proceso | publica | Este método le pide al usuario los datos del proceso que quiere crear, y el tipo de proceso que quiere crear. Al final, regresa el objeto de tipo Proceso que se ha creado. |
| Ejecutar procesos | * Lista de procesos: ArrayList <Proceso> | ArrayList <Proceso> | Publica | Este método realiza la operación de todos los procesos en la Lista de procesos, mostrando el resultado de cada uno. Además, elimina de la lista los procesos que no se guardan. Al final, regresa la lista de procesos faltantes. |

Clase: Consola

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros :  Tipo de dato | Salida: Tipo de dato | Visibilidad | Propósito |
| Pedir nombre | * ninguno | String | publica | Este método le pide al usuario que ingrese un nombre de proceso, y devuelve lo ingresado |
| Pedir variable de cambio | * ninguno | int | Publica | Este método le pide al usuario que ingrese la cantidad de cambio que hará el proceso, y regresa el dato ingresado |
| Pedir tipo | * ninguno | int | Publica | Este método le pide al usuario que ingrese el tipo de proceso que quiere crear, y regresa lo ingresado. Los tipos de procesos se definen según el siguiente código:   * 1 = Proceso CPU * 2 = Proceso I/O * 3 = Proceso Daemon |
| Mostrar resultado | * Número original: int * Tipo proceso: int * Resultado: int | void | Publica | Este método le muestra al usuario el resultado de un proceso ya ejecutado. |

Clase: Main

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros :  Tipo de dato | Salida: Tipo de dato | Visibilidad | Propósito |
| Ninguno |  |  |  |  |

### 2. Diseño: Diagrama de Clases (30 puntos)

* Asegúrate de mostrar atributos y métodos con visibilidad (+, -).
* Indica relaciones entre clases (asociación, agregación, etc.).
* Incluye el driver program (Main).

<https://lucid.app/lucidchart/3bad29a5-8e7a-4dfb-a0e6-ad0cca8bafb2/edit?viewport_loc=-377%2C-376%2C2558%2C1360%2C0_0&invitationId=inv_2b9a18f8-d7a8-48b8-a730-4393cf2689f8>

### 3. Programa (40 puntos)

En cada archivo `.java`, asegurarse de incluir:

* Las clases necesarias.
* Uso adecuado de objetos.

Menú que debe implementar el driver program:

* 1. Nuevo comprador
  2. Nueva solicitud de boletos
  3. …
  4. …
  5. Salir

GitHub: colocar aquí la URL:

https://github.com/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Checklist antes de entregar

Está claro el análisis?

El diagrama tiene los elementos UML correctamente?

Subiste tu código a GitHub con todo lo necesario?