EL ELEVADOR - TAREA 1

@Montiel Juarez Oscar Ivan

@Torres Delgadillo Mixcotal Samuel

Explicación al problema

Se implementa la simulación de un elevador en una facultad que opera en 5 pisos.

- **Contexto:**

  El elevador debe atender múltiples usuarios (alumnos) que llaman al elevador desde cualquier piso y desean ir a otro.

- **Problema:**

  Se debe sincronizar el acceso al elevador evitando que se sobrecargue o monopolice un piso en particular.

- **Restricciones:**
  - El elevador tiene capacidad para 5 pasajeros.
  - Los usuarios pueden llamar al elevador desde cualquier piso y tienen que subir y bajar en el mismo.

Lenguaje y Entorno

- **Lenguaje:** Python
- **Entorno de desarrollo:**

  Se ha desarrollado y probado en un entorno de Python 3

- **Dependencias:**
  - Módulos estándar: `threading`, `time`, `random`, `signal`, `sys`, `os`

Instrucciones para Ejecutar el Programa

1. **Requisitos previos:**

- Tener instalado Python 3.
- Clonar o copiar el archivo (`tarea1_montielOscar_torresSamuel.py`) en un directorio adecuado.

2. **Ejecución:**

   Abrir una terminal o consola y ejecutar:

   ```
   python3 tarea1_montielOscar_torresSamuel.py
   ```

3. **Notas adicionales:**
   - El programa genera pasajeros de forma continua.
   - Se usa `Ctrl+C` para detener el programa

Estrategia de Sincronización

Para lograr la correcta sincronización se emplean las siguientes técnicas y mecanismos:

- **Hilos (Threads):**
  - Se lanza un hilo para el elevador y cada pasajero es un hilo independiente.
- **Semáforos:**
  - Se utiliza un semáforo para controlar el número máximo de pasajeros concurrentes que pueden generar nuevos hilos (definido como `MAX_PASAJEROS` , basado en `os.cpu_count()`).
  - Otro semáforo controla la capacidad del elevador (limitada a 5 pasajeros).
- **Bloqueos (Locks):**
  - Se utiliza un `threading.Lock()` para proteger el acceso a estructuras compartidas (como el diccionario `paradas` y la lista `pasajeros`), evitando condiciones de carrera.
- **Eventos (Event):**
  - Cada pasajero tiene un `threading.Event()` (atributo `evento_subir`) que se activa cuando el elevador lo recoge, sincronizando la espera y el inicio de su viaje.

- **Algoritmo LOOK:**
    - El elevador implementa una versión modificada del algoritmo LOOK para priorizar pisos con solicitudes en la dirección actual y cambiar de dirección cuando no hay solicitudes, evitando así la inanición de usuarios en otros pisos.