Ejercicio1: Lavandería CleanFast

La empresa **CleanFast** ofrece un servicio de **lavandería automática** con autoservicio. Es un pequeño local de barrio que dispone de **4 lavadoras**, donde los clientes pueden venir a lavar su ropa sin asistencia. Cada lavado dura un tiempo aleatorio entre **5 y 10 segundos**, simulando el tiempo real que tomaría un ciclo corto.

El lavado cuesta **3 euros**, pero **el cliente no paga ni se tiene en cuenta como atendido hasta que accede efectivamente a una lavadora**. Si no hay ninguna lavadora disponible, el cliente debe esperar en la cola hasta que alguna quede libre.

Queremos desarrollar una **simulación concurrente** de esta lavandería, en la que se gestionen correctamente los recursos disponibles (las lavadoras) y el flujo de clientes de forma realista.

El programa debe mostrar un menú con tres opciones:

1. **Simular llegada de clientes:**
   Esta opción lanza varios hilos, cada uno representando un cliente que llega a la lavandería.
   - Si hay lavadoras disponibles, el cliente accede, paga, y comienza su lavado (tiempo aleatorio entre 5 y 10 segundos).
   - Si todas las lavadoras están ocupadas, el cliente espera en una cola hasta que una quede libre.
2. **Mostrar estado actual de la lavandería:**
   Esta opción muestra:
   - El número total de clientes **atendidos** (solo se cuenta cuando empieza el lavado).
   - La ganancia total (clientes atendidos × 3 €).
   - El número de lavadoras **disponibles**.
   - El número de lavadoras **en uso**.
3. **Cerrar la lavandería:**
   Al seleccionar esta opción:
   - No se aceptan más clientes nuevos.
   - El programa debe **esperar a que todos los clientes que estaban esperando o lavando terminen**.
   - Cuando todas las lavadoras estén libres, se debe finalizar correctamente la ejecución.

Ejercicio2: Fistro Servidooorrrrr

(Diseñada por Felipe G.R.)

🕸️ Práctica: Servidor HTTPS con respuesta aleatorias de Chiquito de la Calzada

Descripción

El objetivo de esta práctica es desarrollar una aplicación **cliente-servidor HTTPS en Java** que sirva una página HTML interactiva. El servidor debe generar dinámicamente una página HTML con un botón titulado **"AL ATAQUERRR!!"**. Al pulsar dicho botón, se enviará una **petición POST** al servidor. Este devolverá la misma página HTML con una **frase aleatoria de Chiquito de la Calzada** justo debajo del botón.

🎯 Requisitos

- El servidor debe funcionar bajo el protocolo **HTTPS** utilizando un certificado SSL (autofirmado para pruebas locales).
- El contenido HTML debe generarse manualmente como **texto** dentro del código Java.
- El botón debe estar contenido en un formulario con método `POST`.
- Al hacer clic en el botón, se debe mostrar en la misma página una **frase aleatoria** obtenida de una colección de frases predefinidas de Chiquito de la Calzada.
- El servidor debe ser capaz de manejar múltiples peticiones concurrentes utilizando **hilos** (threads).
- El servidor debe ser capaz de manejar excepciones y errores de forma adecuada, devolviendo un mensaje de error en caso de fallo.
- El servidor debe ser capaz de cerrar conexiones de forma segura y liberar recursos al finalizar.
- Debe de gestionarse la concurrencia de peticiones utilizando metodos sincronizados o bloqueos (locks) para evitar problemas de acceso concurrente a los recursos compartidos.

🔑 Certificado SSL

- Para generar un certificado SSL autofirmado, puedes utilizar el siguiente comando de `keytool`:

```
keytool -genkeypair -alias mydomain -keyalg RSA -keystore keystore.jks -
storepass password -validity 365
```

Donde `mydomain` es el alias del certificado, `keystore.jks` es el nombre del archivo donde se guardará el certificado, `password` es la contraseña del keystore y `365` es el número de días de validez del certificado.

- Asegúrate de que el keystore se encuentre en la misma carpeta que tu código Java o proporciona la ruta completa al archivo.

- Asegúrate de que el puerto que elijas para el servidor HTTPS esté disponible y no esté siendo utilizado por otro servicio. Puedes elegir un puerto como `8443`.

🔌 Posibles mejoras optativas a realizar

- Implementar un sistema de logging para registrar las peticiones y respuestas del servidor. Por ejemplo:
- `[2017-05-18 14:04:07] [INFO] peticion recibida y procesada por el servidor`
- `[2017-05-18 14:04:07] [ERROR] error al procesar la peticion recibida`
- Obtener las frases de un archivo `frases.txt` en lugar de tenerlas codificadas en el programa.
- Añadir un contador de peticiones para mostrar cuántas veces se ha pulsado el botón.

💡 Comportamiento esperado

1. El cliente accede a `https://localhost:puerto/`.
2. Se muestra una página HTML con un título, un botón `"AL ATAQUERRR!!"` y (en principio) ningún mensaje.
3. Al pulsar el botón, se realiza una petición `POST` al servidor.
4. El servidor responde con la misma página HTML, pero incluyendo una **frase aleatoria** bajo el botón.

🛠️ Sugerencias

- Se recomienda generar el certificado SSL con la herramienta `keytool`.
- La colección de frases puede estar implementada como una lista estática en Java.
- La lógica para elegir una frase debe asegurar aleatoriedad.