# Ficha Exercício 1

## 1. Simulação de Processamento de Pedidos num E-commerce.

1.1. Implemente uma classe `Pedido` que represente um pedido de um e-commerce, com informações como `id`, `cliente` e `valor`.

1.2. Crie uma classe `ProcessadorDePedidos` que implemente `Runnable` e simule o processamento de um pedido a cada 2 segundos (usando `Thread.sleep(2000)`).

1.3. No `main`, crie uma lista de pedidos e inicie três threads do `ProcessadorDePedidos`, cada um processando um pedido diferente.

1.4. Use `interrupt()` para parar as threads após 10 segundos, simulando um evento de paragem do sistema.

## 2. Monitorização de Temperatura com Interrupção:

2.1. Implemente uma classe `SensorTemperatura` que gera uma leitura aleatória de temperatura entre 15°C e 35°C a cada segundo.

2.2. Utilize uma lista sincronizada para armazenar as leituras de temperatura.

2.3. Crie um `Runnable` que registe estas leituras durante um período de 15 segundos. Se a temperatura ultrapassar os 30°C, o thread de monitorização deve ser interrompido e imprimir uma mensagem de alerta.

2.4. No `main`, inicie a monitorização e observe as leituras serem interrompidas ao detectar altas temperaturas.

## 3. O Simulação de Sistema de Chat com Notificações.

3.1. Crie uma classe `Mensagem` que armazene informações sobre uma mensagem, como remetente, destinatário e conteúdo.

3.2. Crie uma classe `ChatNotifier` que implemente `Runnable` e simule o envio de uma notificação para um utilizador a cada nova mensagem recebida.

3.3. No `main`, inicie três threads de `ChatNotifier` com intervalos diferentes (por exemplo, 1 segundo, 2 segundos, 3 segundos).

3.4. Use o método `join()` para garantir que o thread principal aguarde o término de todos os threads de notificações antes de exibir uma mensagem final de "Todas as notificações enviadas".

## 4. Processamento de Ficheiros de Log

4.1. Crie uma classe chamada `LeitorDeLog` que implemente a interface `Runnable`. Esta classe deverá simular a leitura de um arquivo de log, linha por linha, com um intervalo de 100 milissegundos entre cada leitura.

4.2. Na classe `LeitorDeLog`, cada linha lida deverá ser adicionada a uma lista compartilhada entre os threads.

4.3. No método `main` da aplicação:

4.3.1. Crie três threads, cada um utilizando uma instância de `LeitorDeLog` para ler um arquivo de log diferente (`log1.txt`, `log2.txt` e `log3.txt`).

4.3.2. Inicie os três threads e, em seguida, use `join()` para garantir que o programa principal aguarde a conclusão dos três threads antes de continuar..

4.3.3. Após a conclusão, exiba o total de linhas processadas dos arquivos de log.

**Lendo um Arquivo em Java**: Para ler um arquivo linha a linha, utilize a classe `BufferedReader` combinada com `FileReader`.

```java
try (BufferedReader br = new BufferedReader(new FileReader("caminho_do_arquivo.txt"))) {
    String linha;
    while ((linha = br.readLine()) != null) {
        System.out.println(linha); // Processa cada linha lida
    }
} catch (IOException e) {
    e.printStackTrace();
}
```