1. Dormindo e retomando uma Thread

- **Objetivo:** Demonstrar o uso do método `sleep()` para pausar uma *thread* e como interrompê-la.

2. Introdução

A programação concorrente permite que múltiplas *threads* sejam executadas simultaneamente. No entanto, pode ser necessário pausá-las temporariamente. O método `TimeUnit.SECONDS.sleep(n)` suspende a execução de uma *thread* por um determinado tempo, enquanto `interrupt()` pode ser usado para interrompê-la.

Este exemplo explora o comportamento das *threads* ao serem pausadas e retomadas, o que é útil para cenários como agendamentos e temporizações.

3. Descrição do Código Original

O código contém duas classes principais:

Relogio.java

- Implementa a interface `Runnable`, criando uma *thread* que imprime a hora atual a cada segundo, por 10 iterações.
- Usa `TimeUnit.SECONDS.sleep(1)` para pausar a execução por 1 segundo entre cada impressão.
- Se a *thread* for interrompida, exibe a mensagem `"O relógio foi interrompido"`.

Principal.java

- Instancia e inicia a *thread* `Relogio`.
- Aguarda 5 segundos antes de chamar `thread.interrupt()` para interromper a execução.

4. Modificações Realizadas

- **Tradução:** Todos os nomes de classes, métodos e variáveis foram traduzidos para o português.
- **Alteração na Mensagem de Interrupção:** Foi adicionado `"O relógio foi interrompido"` para identificar claramente quando a *thread* foi interrompida.
- **Teste com Diferentes Valores de `sleep()`:** Alteramos o tempo de espera da *thread* principal para observar mudanças no comportamento.

5. Testes Realizados e Resultados Obtidos

Cenário 1: Código Original

Saída esperada: A *thread* imprime a data a cada segundo por 10 vezes, mas é interrompida após 5 segundos.

Saída real:

```
Wed Feb 07 15:00:00 BRT 2025
Wed Feb 07 15:00:01 BRT 2025
Wed Feb 07 15:00:02 BRT 2025
Wed Feb 07 15:00:03 BRT 2025
Wed Feb 07 15:00:04 BRT 2025
O relógio foi interrompido
```

O comportamento está conforme esperado.

Cenário 2: Mudando `sleep(5)` para `sleep(8)`

- A *thread* `Relogio` continua imprimindo a hora normalmente até o tempo de interrupção.
- Como o `sleep(8)` da *thread* principal é maior que o tempo total do `Relogio`, a interrupção acontece tarde demais, permitindo que o relógio complete sua execução.

Conclusão dos Testes

- O método `interrupt()` interrompe a *thread*, mas apenas se ela estiver em um estado que permita ser interrompida.
- Alterar os valores de `sleep()` afeta diretamente o tempo de execução da *thread*.

6. Conclusão

Este exemplo demonstrou: - Como usar `sleep()` para controlar a execução de *threads*. - Como `interrupt()` pode interromper uma *thread* em execução. - A importância de testar diferentes tempos para compreender melhor o comportamento de concorrência em Java.