Ficha Exercício 4: Sistema de Processamento de Pedidos

Objetivo

Desenvolver um sistema um Sistema de Processamento de Pedidos, que implementa a comunicação entre threads utilizando os métodos `wait()`, `notify()`, e `notifyAll()`. Evitar condições de corrida (Race Condition) e garantir acesso sincronizado à fila compartilhada. Simular o ambiente com múltiplos cozinheiros e garçons.

Cenário

Imagine um sistema de processamento de pedidos de um restaurante, onde:

- Os **cozinheiros (Produtores)** preparam pratos e os colocam em uma fila.
- Os **garçons (Consumidores)** retiram os pratos da fila para servir aos clientes.

O sistema deve garantir que:

1. Os cozinheiros só coloquem pratos na fila se houver espaço disponível.
2. Os garçons só retirem pratos da fila se houver pratos prontos.

Simular dois cenários:

1. **Usando `notifyAll()`**: Todas as threads que estão esperando são notificadas.
2. **Usando `notify()`**: Apenas uma thread específica é notificada.

Comparar os resultados:

- Identificar como o comportamento do sistema muda nos dois casos.
- Analisar possíveis diferenças de desempenho e interações entre threads.

Requisitos

1. **Classe `FilaDePedidos`** (Objeto Compartilhado)
   1.1. Gerencia uma fila limitada de pedidos.
   1.2. Métodos sincronizados:
      1.2.1. `void adicionarPedido(String pedido)`: Adiciona um pedido à fila e notifica os consumidores.

1.2.2. `String retirarPedido()`: Remove um pedido da fila e notifica os produtores

2. **Classe `Cozinheiro`** (Produtor)

2.1. Implementa `Runnable` e:

2.1.1. Prepara pratos (simulados com nomes como "Prato 1", "Prato 2").

2.1.2. O tempo simulado para preparar o prato deve ser 2 segundos.

2.1.3. Usa `adicionarPedido` para colocar os pratos na fila.

3. **Classe `Garcom`** (Consumidor)

3.1. Implementa `Runnable` e:

3.1.1. Retira pratos da fila usando `retirarPedido`.

3.1.2. Exibe no console os pratos servidos.

3.1.3. O tempo para servir o prato deve ser 3 segundos.

4. Classe Principal (`Main`)

4.1. Inicia múltiplos threads de **Cozinheiros** e **Garçons**.

4.2. Simula um ambiente onde os cozinheiros e garçons trabalham simultaneamente, utilizando a fila compartilhada.

4.3. A capacidade máxima da fila compartilhada deve ser de 6 pratos.

4.4. Permite o sistema rodar por 15 segundos antes de encerrar.

5. **Executar Simulações**:

5.1. Rodar o programa em ambos os cenários (uma vez com `notifyAll()` e outra com `notify()`).

5.2. Observar o comportamento das threads e o impacto nas mensagens exibidas no console.

6. **Relatar Diferenças**:

6.1. Analisar como o uso de `notifyAll()` ou `notify()` afeta a execução do programa.

6.2. Identificar possíveis vantagens e desvantagens de cada abordagem.