Relatório: Organização e Sincronização de Vendas de Ingressos em um Cinema

1. Objetivo

Este exercício demonstra a organização de operações concorrentes de venda e devolução de ingressos em um cinema, utilizando múltiplas threads. O foco é garantir que o acesso às vagas disponíveis nas duas salas do cinema seja feito de forma sincronizada, evitando condições de corrida.

2. Estrutura do Código

2.1. Classe `Cinema`

- **Descrição:**
  - Gerencia as vagas disponíveis para duas salas de cinema.
  - Utiliza dois objetos de controle (`controleCinema1` e `controleCinema2`) para sincronizar as operações de venda e devolução de ingressos.
- **Métodos Principais:**
  - `venderIngressos1(int quantidade)`: Vende ingressos da Sala 1 de forma sincronizada.
  - `venderIngressos2(int quantidade)`: Vende ingressos da Sala 2 de forma sincronizada.
  - `devolverIngressos1(int quantidade)`: Devolve ingressos na Sala 1 de forma sincronizada.
  - `devolverIngressos2(int quantidade)`: Devolve ingressos na Sala 2 de forma sincronizada.
  - Métodos `getVagasSala1()` e `getVagasSala2()` para retornar o número de vagas restantes.

2.2. Classes `Bilheteria1` e `Bilheteria2`

- **Descrição:**
  - Cada classe implementa a interface `Runnable` e representa uma bilheteira operando em uma thread separada.
  - As bilheteiras interagem com a classe `Cinema` para executar operações de venda e devolução de ingressos.
- **Operações Executadas:**
  - `Bilheteria1` realiza uma sequência de vendas e devoluções que afetam ambas as salas.
  - `Bilheteria2` executa outra sequência de operações de venda e devolução.

2.3. Classe `Principal`

- **Descrição:**
  - Inicializa o objeto `Cinema` e as bilheteiras.

- Cria e inicia as threads correspondentes a `Bilheteria1` e `Bilheteria2`.
- Aguarda a conclusão das threads e, em seguida, exibe o número final de vagas disponíveis em cada sala.

3. Fluxo de Execução

1. **Inicialização:**
   - O objeto `Cinema` é criado com 20 vagas em cada sala.
   - As bilheteiras (`Bilheteria1` e `Bilheteria2`) são instanciadas com referência ao mesmo objeto `Cinema`.
2. **Execução Concorrente:**
   - As threads das bilheteiras são iniciadas, executando operações de venda e devolução de ingressos.
   - A sincronização nos métodos da classe `Cinema` garante que apenas uma thread acesse as vagas de cada sala por vez.
3. **Finalização:**
   - Após a execução das threads (garantida pelo `join`), o programa exibe as vagas restantes em cada sala.

4. Exemplo de Execução

Saída do Console:

Vagas na Sala 1: 6 Vagas na Sala 2: 10

5. Conclusão

O exercício evidencia a importância da sincronização em ambientes multithread para evitar condições de corrida e garantir a consistência dos dados compartilhados. Apesar das operações concorrentes de venda e devolução de ingressos, o uso correto de blocos sincronizados mantém a integridade do número de vagas disponíveis, conforme demonstrado na saída do programa.