Relatório: Sincronização de Acessos à Conta Bancária

1. Objetivo

Este exercício visa demonstrar como sincronizar o acesso a recursos compartilhados em um ambiente multithread. Através de duas threads (`Banco` e `Companhia`), que manipulam o saldo de uma conta bancária de forma concorrente, mostramos como usar o mecanismo de sincronização do Java para garantir a consistência dos dados.

2. Estrutura do Código

2.1. Classe `Banco`

- **Descrição:**
  - Implementa a interface `Runnable` para criar uma thread que subtrai valores do saldo da conta.
- **Funcionamento:**
  - A cada iteração no loop, subtrai 1000 da conta. O número de iterações é 100.

2.2. Classe `Companhia`

- **Descrição:**
  - Implementa a interface `Runnable` para criar uma thread que adiciona valores ao saldo da conta.
- **Funcionamento:**
  - A cada iteração no loop, adiciona 1000 à conta. O número de iterações é 100.

2.3. Classe `Conta`

- **Descrição:**
  - Contém o saldo da conta e métodos sincronizados para adicionar ou subtrair valores.
- **Métodos Sincronizados:**
  - O uso da palavra-chave `synchronized` garante que apenas uma thread manipule o saldo ao mesmo tempo, evitando problemas de concorrência.

2.4. Classe `Principal`

- **Descrição:**
  - Orquestra a execução das threads `Banco` e `Companhia`, inicializa a conta e exibe o saldo inicial e final após a execução das threads.

3. Exemplo de Execução

Saída do Console:

Conta : Saldo Inicial: 1000.000000 Conta : Saldo Final: 1000.000000

4. Explicação do Resultado

A sincronização de métodos na classe `Conta` garante que, enquanto uma thread estiver manipulando o saldo, outra thread não possa fazer o mesmo. Isso previne inconsistências, como ler o saldo parcialmente alterado por uma thread. Mesmo com operações de adição e subtração ocorrendo simultaneamente, o saldo final permanece consistente devido à sincronização das operações.

5. Testes Realizados

- **Cenário 1:** Adicionar e subtrair valores de uma conta simultaneamente utilizando múltiplas threads.
  - Resultado esperado: O saldo final deve ser consistente, sem alterações inesperadas.
  - O saldo inicial foi de 1000 e, após 100 iterações de adição e subtração de 1000, o saldo final permanece 1000 devido à sincronização.

6. Conclusão

A sincronização de métodos foi fundamental para garantir a consistência do saldo da conta bancária em um ambiente multithread. Ao usar o modificador `synchronized`, evitamos problemas como condições de corrida, onde duas threads tentam acessar e modificar o saldo simultaneamente, levando a inconsistências. O código funciona corretamente, mantendo a integridade dos dados.