Programação Concorrente em Java

Aula 05

Condição de Corrida e Seção Crítica

Condição de Corrida e Seção Crítica

- **Condição de Corrida**: é uma condição especial que ocorre dentro da seção crítica.
- **Seção Crítica:** é um pedaço do código que é executado por múltiplas threads e onde a sequencia de execução das threads faz diferença no resultado.

  Quando o resultado de múltiplas threads executando a seção crítica difere, dependendo da ordem das threads, a seção crítica é dita em *condição de corrida*.

Seção Crítica

- Rodar mais de uma thread dentro de uma aplicação geralmente não casa problemas. O problema surge quando as threads competem pelos mesmos recursos (salvo em caso de leitura).

```
public class Counter {
    protected long count = 0;
    public void add(long value){
        this.count = this.count + value;
    }
 }
```

Seção Crítica

- Análise do Código
  - Imagine duas threads, A e B executando o método add na mesma instância da classe Counter.
  - É impossível dizer qual thread irá executar primeiro (entrar na seção crítica).
  - O código na seção crítica (método add) não é “atômico” e sim executado em partes:
    - Ler this.count da memória para o registrador;
    - Add value para o registrador;
    - Finalmente, escrever o valor calculado no registrado na memória.

Seção Crítica

- Entrelaçamento (interleaving) - Cenário

```
this.count = 0;
   A:  Reads this.count into a register (0)
   B:  Reads this.count into a register (0)
   B:  Adds value 2 to register
   B:  Writes register value (2) back to memory. this.count now equals 2
   A:  Adds value 3 to register
   A:  Writes register value (3) back to memory. this.count now equals 3
```

Qual valor deveria ser o correto para o contador?

Seção Crítica

- Condição de Corrida
  - O código em add é uma seção crítica da aplicação.
  - Quando múltiplas threads concorrem por seu acesso, temos uma condição de corrida. A ordem que essas threads “entram” na seção, afeta o resultado.
- *“More formally, the situation where two threads compete for the same resource, where the sequence in which the resource is accessed is significant, is called* ***race conditions****. A code section that leads to race conditions is called a* ***critical section****.”*

Seção Crítica

- Previnindo Condição de Corrida.
  - O programador deve ter certeza de executar a seção crítica como uma operação **atômica.**
  - Ou seja, apenas uma thread pode entrar na seção crítica por vez. As outras threads devem esperar.
  - Como fazer isso em Java?
    - Sincronização de blocos ou métodos.
    - Locks.
    - Variáveis atômicas.

Seção Crítica

- Para seções críticas pequenas, torna o bloco sincronizado muitas vezes pode funcionar bem.
- Para seções críticas maiores, é benéfico quebrar a seção crítica em seções menores, permitindo assim que várias threads executem um pequeno pedaço da seção crítica.
- Isso reduz a contenção e aumenta o **throughput** total da seção crítica.

Seção Crítica

```
public class TwoSums {

    private int sum1 = 0;
    private int sum2 = 0;

    public void add(int val1, int val2){
        synchronized(this){
            this.sum1 += val1;
            this.sum2 += val2;
        }
    }
}
```

Seção Crítica

```
public class TwoSums {

    private int sum1 = 0;
    private int sum2 = 0;
    private Integer sum1Lock = new Integer(1);
    private Integer sum2Lock = new Integer(2);
    public void add(int val1, int val2){
        synchronized(this.sum1Lock){
            this.sum1 += val1;
        }
        synchronized(this.sum2Lock){
            this.sum2 += val2;
        }
    }
}
```

Referências

- http://tutorials.jenkov.com/java-concurrency/race-conditions-and-critical-sections.html