04. Coleções thread-safe

Date	@26/08/2022
Categoria	Java
Curso	Curso Threads em Java 1: programação paralela

Tópicos

1. Threads manipulando listas
2. Listas sincorinizadas
3. Palavra chave: synchronized
4. Acesso concorrente
5. A calsse java.util.Vector
6. Para saber mais: Outras coleções threads-safe
7. Mãos à obra: ArrayList vs Vector
8. O que aprendemos?

Acesso concorrente

Descreva, com suas próprias palavras, por qual motivo foi necessário sincronizar os métodos da nossa implementação de lista.

O problema é que precisamos garantir que todo o código abaixo será executado de maneira atômica, de uma vez só. Uma thread não pode parar na primeira linha, sem ter incrementado o índice. Se isso acontece, uma outra thread poderá executar esse código e adicionar um elemento na mesma posição.

```
public void adicionaElementos(String elemento) {
    this.elementos[indice] = elemento;
    this.indice++;
}COPIAR CÓDIGO
```

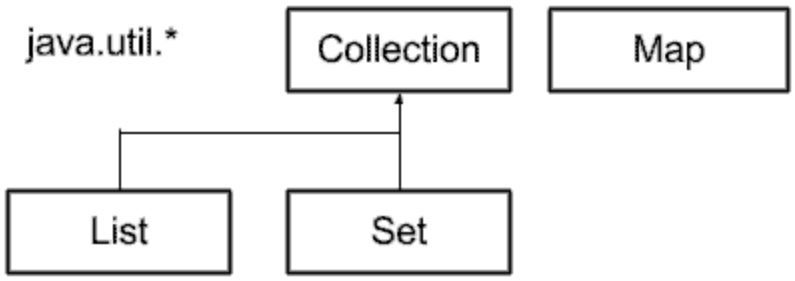
Para piorar, essas duas linhas desse método se tornam várias linhas no código compilado (bytecode). Ou seja, a chance que der algo errado aumenta.

Esse problema não só acontece nos métodos que modificam os dados. Isso também se aplica aos métodos acessores, `get` por exemplo. Eles também devem usar `synchronized`.

Para saber mais: Outras coleções threads-safe

Vimos na aula a classe `java.util.Vector` em ação, mas a API de Collections possuem muito mais interfaces e implementações.

Dentro dessa API existem 4 interfaces principais: `java.util.Collection`, `java.util.List`, `java.util.Set` e `java.util.Map`.



Se a classe `Vector` é a versão thread-safe de uma lista, será que existem para as outras interfaces implementações thread-safe? Claro que sim!

Para os mapas (`Map`) podemos usar a antiga classe `Hashtable`:

```
Map mapaThreadSafe = new Hashtable();COPIAR CÓDIGO
```

E também temos uma implementação mais recente e performática de mapas, a classe `ConcurrentHashMap`:

```
//do pacote java.util.concurrent
Map mapaThreadSafe = new ConcurrentHashMap();COPIAR CÓDIGO
```

Para o `Set` (conjunto) não existe uma implementação pronta na API padrão do Java mas podemos utilizar a classe `Collections` para construir um `Set` sincronizado:

```
Set conjunto = Collections.synchronizedSet(new HashSet());
```

O que aprendemos?

- Ao modificar o objeto concorrentemente, coisas inesperadas podem aparecer;
- *Thread safe* significa que o código funciona corretamente mesmo com vários threads compartilhando o objeto;
- Há coleções *thread safe*, como o `java.util.Vector` para lista ou `java.util.Hashtable` para mapas.