CopyOnWriteArrayList vs Collections.synchronizedList()

Uso do CopyOnWriteArray:

```java
import java.util.concurrent.CopyOnWriteArrayList;

public class ExemploConcorrente {
    public static void main(String[] args) {
        CopyOnWriteArrayList<String> listaSegura = new CopyOnWriteArrayList<>();

        // Adição thread-safe
        listaSegura.add("Item 1");
        listaSegura.add("Item 2");

        // Iteração thread-safe - não precisa de sincronização explícita
        for (String item : listaSegura) {
            System.out.println(item);
        }
    }
}
```

Comparação:

Característica	CopyOnWriteArrayList	Collections.synchronizedList()
Estratégia	Cópia do array em modificações	Sincronização tradicional
Performance em leitura	Muito rápida (sem bloqueio)	Rápida (com bloqueio mínimo)
Performance em escrita	Lenta (copia todo o array)	Rápida (bloqueio apenas durante operação)
Iteração segura	Sim	Não
Consistência	Snapshot consistente	Consistência momentânea
Uso ideal	Leitura frequente, escrita pouco frequente	Escrita frequente

add e remove de CopyOnWriteArrayList:

```java
public boolean add(E e) {
    synchronized (lock) {
        Object[] es = getArray();
        int len = es.length;
        es = Arrays.copyOf(es, len + 1);  // 1. Cria nova cópia com tamanho aumentado
        es[len] = e;                      // 2. Adiciona o novo elemento no final
        setArray(es);                     // 3. Substitui a referência do array
        return true;
    }
}
```

1. Bloqueio: Entra em um bloco sincronizado usando um lock interno
2. Cópia: Cria uma nova cópia do array atual com tamanho +1
3. Inserção: Adiciona o novo elemento na última posição
4. Troca: Atualiza a referência do array interno para a nova cópia
5. Liberação: Libera o lock após a operação completa

```java
public E remove(int index) {
    synchronized (lock) {
        Object[] es = getArray();
        int len = es.length;
        E oldValue = elementAt(es, index);  // 1. Pega o valor a ser removido
        int numMoved = len - index - 1;
        Object[] newElements;
        if (numMoved == 0) {
            newElements = Arrays.copyOf(es, len - 1);  // 2a. Caso simples (último elemento)
        } else {
            newElements = new Object[len - 1];
            System.arraycopy(es, 0, newElements, 0, index);  // 2b. Copia parte anterior
            System.arraycopy(es, index + 1, newElements, index, numMoved);  // 2c. Copia parte posterior
        }
        setArray(newElements);  // 3. Substitui o array
        return oldValue;
    }
}
```

1. Bloqueio: Entra no bloco sincronizado
2. Identificação: Localiza o elemento a ser removido
3. Cópia:
    - Se for o último elemento: cópia simples reduzindo tamanho
    - Caso contrário: cópia em duas partes (antes/depois do índice)
4. Troca: Atualiza a referência do array interno
5. Liberação: Libera o lock