UNIVERSIDADE FEDERAL DE GOIÁS

ENGENHARIA DE SOFTWARE

SOFTWARE CONCORRENTE E DISTRIBUIDO – INF0298

Discente:

Adriel Lenner Vinhal Mori

Docente:

Vagner Jose Do Sacramento Rodrigues

Exercícios sobre concorrência e threads - complementar 1 – 4 jul 2023

Goiânia

2023

Exercícios avançados de programação concorrente e locks

1.

Ao rodar várias vezes os resultados não se mantiveram constantes. Como o acesso a classe pelo thread é de forma concorrente e não sincronizadas à ArrayList, algumas mensagens não serão adicionadas corretamente, e mesmo duplicadas algumas vezes. Esses motivos ocorrem por conta do ArrayList não ser utilizado como um thread-safe em um ambiente multi-thread, elando a problemas de concorrência.

3.

O Vector cuida da sincronização internamente em seus métodos compostos no seu pacote java.util.Vector, o que um torna um tipo de abordagem thread-safe. Avaliando a documentação do Vector disponível em https://docs.oracle.com/javase/8/docs/api/java/util/Vector.html, e código-fonte opensource disponibilizado pela JDK7 em https://github.com/openjdk-mirror/jdk7u-jdk/blob/master/src/share/classes/java/util/Vector.java, pode-se ver os métodos que modificam o conteúdo do vetor são marcados como “synchonized” o que garante que apenas uma thread pode executar esses métodos por vez, impedindo condições de corrida.

Diante ao supracitado, adicionar o ‘Vector’ evita-se a necessidade de adicionar manualmente o ‘synchronized’, como se pede na questão de número 2.

4.

Sabendo que as implementações de HashSet e LinkedLisnt não são thread-safe, e o acesso concorrente leva a problemas de concorrência, onde algumas execuções os threads se entrelaçam de maneira que o código funcione corretamente, mas não é garantida.