FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Professor Eduardo Marques

Programação Concorrente

Trabalho 2: Implementação de Conjuntos Baseados em Tabelas de Dispersão

Junho de 2022

Trabalho realizado por:

Hugo Almeida - 201905410

Pedro Leite - 201906697

Introdução

Para este projeto, foi nos proposto o desenvolvimento de implementações de conjuntos baseados em tabelas de dispersão (*hash sets*), através de programação concorrente. Os elementos do conjunto estão dispersados por entradas numa tabela de *hashing*.

Foi nos fornecido um código base e 4 exercícios com o objetivo de aplicar diferentes implementações.

Hset1

Para realizar este exercício utilizamos como base o código fornecido no *HSet0*. Em vez de utilizarmos o *synchronized*, utilizamos *locks*, o que significa que temos de definir quando é que os processos são fechados e abertos.

Começamos por adicionar 2 campos privados: um *ReetrantLock* e outro *Condition*.

Nas funções *size*, *add*, *remove*, *contains*, *waitFor* e *rehash*, utilizamos a *ReetrantLock* para dar *lock* e *unlock*.. Damos *unlock* quando o *thread* está a ocupar a *lock*, através da função *isHeldByCurrentThread*.

Na função *waitFor* utilizamos a *Condition* na função *awai*t, para o processo aguardar até ser chamado o *signalAll*, na função *add*, que vai acordar o processo.

Hset2

Para realizar este exercício utilizamos como base o código do exercício anterior. Em vez do *ReentrantLock*, utilizamos o *ReentrantReadWriteLock*, e adicionamos mais 2 campos privados: um *ReadLock* e o outro *WriteLoc*k. Agora podemos ser mais específicos com os tipos de *locks*, que podem ser para ler ou para escrever, assim se tivermos um processo que requer o *lock write* e outro o *lock read*, estes podem ser executados ao mesmo tempo, melhorando a eficiência em relação ao exercício anterior.

Nas funções *size* e *contains*, utilizamos a *ReadLock*, já que estas funções apenas leem. Nas funções *add*, *remove*, *waitFor* e *rehash*, utilizamos a *WriteLock*, já que estas funções apenas escrevem.

Hset3

Para realizar este exercício utilizamos como base o código do exercício anterior. Mas o *ReetrantReadWriteLock* e o *Condition* passam a ser *arrays*. Na função *HSet3*, definimos o tamanho dos *arrays* como o tamanho da tabela de *hash* e inicializamos todas as entradas do *ReetrantReadWriteLock* como um novo *lock* e do *Condition* como uma nova *condition*.

No *rehash* e no *size*, damos *lock* e *unlock* (de escrita) a todos os elementos do *array ReetrantReadWriteLock*. No *size*, definimos uma variável à qual vamos incrementando o tamanho de cada entrada da tabela, no final temos de dar *unlock*, a todas essas entradas.

No *add*, *remove*, *contains*, *waitFor*, os *locks*, o *unlock*, o *isHeldByCurrentThread*, o *signAll* e o *await*, utilizamos sempre o *lock*: *Math.abs(elem.hashCode() % tamanho)*.

Conclusão

Conseguimos realizar com sucesso os 3 primeiros exercícios, passando em todos os testes fornecidos. Tivemos dificuldades na realização do exercício 4, este que não foi entregue.