

Coroutines Map

Необходимо реализовать следующий интерфейс lock-free map в стиле coroutines:

```
public interface CoroutinesMap<K: Comparable<K>, V> {  
    /**  
    * Добавить (key,value) к ассоциативному контейнеру  
    *  
    * Алгоритм должен быть как минимум lock-free  
    *  
    * @param key ключ  
    * @param value значение  
    * @return вернуть существующее value если key уже существует в множестве,  
    null если элемент был добавлен  
    */  
    suspend fun put(key: K, value: V) : V?  
  
    /**  
    * Проверка наличия ключа в множестве  
    *  
    * Алгоритм должен быть как минимум wait-free  
    *  
    * @param key значение ключа  
    * @return значение если элемент содержится в множестве, иначе - null  
    */  
    suspend fun get(key: K) : V?  
  
    /**  
    * Удалить ключ из ассоциативного контейнера  
    *  
    * Алгоритм должен быть как минимум lock-free  
    *  
    * @param key значение ключа  
    * @return вернуть существующее value если key существовал в множестве, null  
    если элемента не было в контейнере  
    */  
    suspend fun remove(key: K) : V?  
  
    /**  
    * Проверка ассоциативного контейнера на пустоту  
    *  
    * Алгоритм должен быть как минимум lock-free  
    *  
    * @return true если множество пусто, иначе - false  
    */  
    suspend fun isEmpty(): Boolean  
  
    /**  
    * Возвращает lock-free Set элементов для ассоциативного контейнера
```

```
*  
* @return новый экземпляр Set для ассоциативного контейнера  
*/  
suspend fun entrySet(): Set<Map.Entry<K,V>>  
}
```

В работе можно использовать только JDK 11 и язык программирования Kotlin. Имя класса реализации `CoroutinesMapImpl`. В работе нужно доработать `lock-free-set` из прошлой работы до `map` и использовать как вложенное поле в `CoroutinesMapImpl`. В конструкторе должно передаваться число потоков в которых будут выполняться запросы для доступа к ассоциативному контейнеру. Сама работа должна выполняться в `Executors.fixedThreadPool`. Все реализованные функции должны быть асинхронными.

При использовании `async` действия должны выполняться асинхронно

```
val m = CoroutinesMapImpl<String, String>(10)  
runBlocking {  
    val a = async { m.put("Moscow", "very big city") }  
    val b = async { m.put("Rostov", "city") }  
    print(a.await());  
    print(b.await());  
}
```

Если не использовать `async`, то каждое действие будет блокироваться до получения результата

```
val m = CoroutinesMapImpl<String, String>(10)  
runBlocking {  
    val a = m.put("Moscow", "very big city")  
    val b = m.put("Rostov", "city")  
    print(a);  
    print(b);  
}
```

From:
<http://wiki.osll.ru/> - Open Source & Linux Lab

Permanent link:
http://wiki.osll.ru/doku.php/courses:high_performance_computing:coroutines_map?rev=1591118115

Last update: 2020/06/02 20:15

