Coroutines Map

```
public interface CoroutineMap<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?


    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>>
}
```