Lock-free контейнер

Необходимо реализовать в lock-free стиле следующий интерфейс:

```
/**
 * Lock-Free множество.
 * @param <T> Тип ключей
 */
public interface Set<T extends Comparable<T>> {
    /**
   * Добавить ключ к множеству
     *
   * Алгоритм должен быть как минимум lock-free
     *
     * @param value значение ключа
     * @return false если value уже существует в множестве, true если элемент
был добавлен
     */
    boolean add(T value);


    /**
   * Удалить ключ из множества
     *
   * Алгоритм должен быть как минимум lock-free
     *
     * @param value значение ключа
     * @return false если ключ не был найден, true если ключ успешно удален
     */
    boolean remove(T value);


    /**
   * Проверка наличия ключа в множестве
     *
   * Алгоритм должен быть как минимум wait-free для типов конечной размерноости и
lock-free для отальных
     *
     * @param value значение ключа
     * @return true если элемент содержится в множестве, иначе - false
     */
    boolean contains(T value);


    /**
   * Проверка множества на пустоту
     *
   * Алгоритм должен быть как минимум lock-free
     *
```

```
     * @return true если множество пусто, иначе - false
     */
    boolean isEmpty();

    /**
   * Возвращает lock-free итератор для множества
     *
     * @return новый экземпляр итератор для множества
     */
    java.util.Iterator<T> iterator();
}
```

Дополнительные условности:

1. Имя класса реализации - *SetImpl*
2. Класс должен иметь конструктор без параметров
3. Pull Request должен содержать в части тестирования проходящие:
    - Нагрузочные тесты на основе jcstress
    - Тесты корректности на основе lincheck
4. В реализации не предполагается увидеть стандартные контейнеры из java.util.concurrent
5. Использовать только JDK 11

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

Permanent link:
http://wiki.osll.ru/doku.php/courses:high_performance_computing:lock_free?rev=1591324355

Last update: **2020/06/05 05:32**