L3 Task

1. Examine the code in the l3-assignment-hash-tables project.

2. (**7 points**) Implement the unimplemented methods and test them.

   The following methods need to be implemented in the HashMap and HashMapOa classes:

   - remove(K key);
   - containsValue(Object value) - checks if exists at least one key, which value is equal to the value specified in the method argument;
   - replace(K key, V oldValue, V newValue) - replaces the *oldValue* of the key with the *newValue* and returns true. If the key does not exist or its value does not match to the *oldValue*, the replacement is not performed and false is returned.

3. (**3 points**) Perform benchmark (using JMH) with one pair of methods from the table below:

No.	Method 1	Method 2
1	Class HashMap: remove()	Class HashMapOa: remove()
2	Class HashMap: contains()	Class HashMapOa: contains()
3	Class HashMap: put()	Class HashMapOa: put()
4	Class HashMap: get()	Class HashMapOa: get()
5	Class HashMap: containsValue()	Class HashMapOa: containsValue()
6	Class HashMap: remove()	Class java.util.HashMap <E>: remove()
7	Class HashMap: contains()	Class java.util.HashMap <E>: contains()
8	Class HashMap: containsValue()	Class java.util.HashMap <E>: containsValue()
9	Class HashMap: put()	Class java.util.HashMap <E>: put()
10	Class HashMap: get()	Class java.util.HashMap <E>: get()
11	Class HashMapOa: remove()	Class java.util.HashMap <E>: remove()
12	Class HashMapOa: contains()	Class java.util.HashMap <E>: contains()
13	Class HashMapOa: containsValue()	Class java.util.HashMap <E>: containsValue()
14	Class HashMapOa: put()	Class java.util.HashMap <E>: put()
15	Class HashMapOa: get()	Class java.util.HashMap <E>: get()

Note: The actual task number will be assigned to you by your lecturer.

Prepare a free-form report. The report must include:

- Description the benchmarked methods.
- The computational complexity of the benchmarked methods.
- The description of benchmark methodology (How benchmarking was carried out).
- The Graph whereby the dependence of the execution time and the input data size is depicted.
- Conclusions. The conclusions, among others, should answer the following questions in free form:
    - Whether the experimentally determined dependence of the execution time and the input size is consistent with the computational complexity of the algorithm/method. If not, why?
    - Which of the methods studied is better in terms of the running time. Why?

L3 Defence

During the defence, lecturers may ask students to complete the following tasks and answer the questions below. The list of questions and exercises is not exhaustive - lecturers may ask questions or practical exercises related to the topic during the defence that are not included in this list.

Possible practical tasks

Implement one or more of the methods from the following list:

Value returned	**Description of the method**
java.util.Set<K>	keySet() Returns the keyset for this image.
java.util.List<V>	values() Returns a list of all image values.
int	getNumberOfCollisions() Returns the number of collisions.
int	averageChainSize() Beautifies the average chain length.
V	replace(K key, V value) Replaces the value corresponding to the key *key* existing in the image with the value *value* and returns the old value. If the key *key* does not exist in the image, the pair (*key, value*) is cached in the image and null is returned.
void	replaceAll(V oldValue, V newValue) Replaces the values of all image pairs with the new value *newValue* if the old value is equal to *oldValue*.

K	putIfAbsent(K key, V value) If the pair (key, value) does not exist in the image, it is written and returned null. Otherwise, the value corresponding to the key key that already exists in the image is returned
void	putAll(Map<K,V> map) Copies all key-value pairs from the map into this image.

Possible theoretical questions

1. What is the computational complexity of various operations implemented in the lab?
2. What are the advantages and disadvantages of solving hash table collisions using separate chaining?
3. What are the advantages and disadvantages of resolving hash table conflicts using open addressing?
4. Be able to explain the various nuances involved in the programming code.