GTU Department of Computer Engineering
CSE 222/505 - Spring 2021
Homework 5
Due date: May 12 2021– 23:55 AM

PART 1 - 40 pts:

Write a custom iterator class MapIterator to iterate through the keys in a HashMap data structure in Java. This class should have the following methods:

- next(): The function returns the next key in the Map. It returns the first key when there is no not-iterated key in the Map.

- prev(): The iterator points to the previous key in the Map. It returns the last key when the iterator is at the first key.

- hasNext(): The method returns True if there are still not-iterated key/s in the Map, otherwise returns False.

- MapIterator (K key): The iterator should start from the given key and still iterate though all the keys in the Map. The iterator starts from any key in the Map when the starting key is not in the Map or not specified (zero parameter constructor).

PART 2 – 60 pts:

Implement KWHashMap interface in the book using the following hashing methods to organize hash table:

- Use the chaining technique for hashing by using linked lists (available in the text book) to chain items on the same table slot.

- Use the chaining technique for hashing by using TreeSet (instead of linked list) to chain items on the same table slot.

- Use the Coalesced hashing technique. This technique uses the concept of Open Addressing to find first empty place for colliding element by using the quadratic probing and the concept of Separate Chaining to link the colliding elements to each other through pointers (indices in the table). The deletion of a key is performed by linking its next entry to the entry that points the deleted key by replacing deleted entry by the next entry. See the following illustration as an example:

Input = {3, 12, 13, 25, 23, 51, 42}

Hash function = data % 10

Insert 3: insert 12: insert 13:

Hash Value	Key	Next	Hash Value	Key	Next	Hash Value	Key	Next
0		NULL	0		NULL	0		NULL
1		NULL	1		NULL	1		NULL
2		NULL	2	12	NULL	2	12	NULL
3	3	NULL	3	3	NULL	3	3	4
4		NULL	4		NULL	4	13	NULL
5		NULL	5		NULL	5		NULL
6		NULL	6		NULL	6		NULL
7		NULL	7		NULL	7		NULL
8		NULL	8		NULL	8		NULL
9		NULL	9		NULL	9		NULL

Insert 25: insert 23: insert 51:

Hash Value	Key	Next	Hash Value	Key	Next	Hash Value	Key	Next
0		NULL	0		NULL	0		NULL
1		NULL	1		NULL	1	51	NULL
2	12	NULL	2	12	NULL	2	12	NULL
3	3	4	3	3	4	3	3	4
4	13	NULL	4	13	7	4	13	7
5	25	NULL	5	25	NULL	5	25	NULL
6		NULL	6		NULL	6		NULL
7		NULL	7	23	NULL	7	23	NULL
8		NULL	8		NULL	8		NULL
9		NULL	9		NULL	9		NULL

Insert 42:

Hash Value	Key	Next
0		NULL
1	51	NULL
2	12	6
3	3	4
4	13	7
5	25	NULL
6	42	NULL
7	23	NULL
8		NULL
9		NULL

Delete 13:

Hash Value	Key	Next
0		NULL
1	51	NULL
2	12	6
3	3	4
4	23	NULL
5	25	NULL
6	42	NULL
7		NULL
8		NULL
9		NULL

Test all the three hash table implementations empirically. Use small, medium, and large-sized data and hash tables in suitable sizes for testing. Perform different tasks over the tables to compare their performance results (like accessing existing/non-existing items or adding/removing items).

RESTRICTIONS:

- Can be only one main class in project
- Don't use any other third part library

GENERAL RULES:

- For any question firstly use course news forum in Moodle, and then the contact TA.
- You can submit assignment one day late and will be evaluated over sixty percent (%60).

TECHNICAL RULES:

- You must write a driver function that demonstrates all possible actions in your homework. For example, if you are asked to implement an array list and perform an iterative search on the list then, you must at least provide the following in the driver function:
    - Create an array list and add items to the list. Append items to head, tail, and k^{th} index of the list.
    - Perform at least two different searches by using two items in the list and print the index of the items.
    - Perform another search with an item that isn't in the array list and inform the user that the item doesn't exist in the array list.
    - Delete an existing item from the list and repeat the searches.
    - Try to delete an item that is not on the array list and throw an exception for this situation.

  The driver function should run when the code file is executed.

- Implement clean code standards in your code;
    - Classes, methods and variables names must be meaningful and related with the functionality.
    - Your functions and classes must be simple, general, reusable and focus on one topic.
    - Use standard java code name conventions.

REPORT RULES:

- Add all javadoc documentations for classes, methods, variables …etc. All explanation must be meaningful and understandable.
- You should submit your homework code, Javadoc and report to Moodle in a "studentid_hw5.tar.gz" file.
- Use the given homework format including selected parts from the table below:

Detailed system requirements	
The Project use case diagrams (extra points)	
Class diagrams	
Other diagrams	
Problem solutions approach	X

Test cases	X
Running command and results	X

GRADING :

- **No OOP design:** **-100**
- **No interface:** **-95**
- **No method overriding:** **-95**
- **No error handling:** **-50**
- **No inheritance:** **-95**
- **No polymorphism:** **-95**
- No javadoc documentation: -50
- No report: -90
- Disobey restrictions: -100
- **Cheating:** **-200**
- Your solution is evaluated over 100 as your performance.

CONTACT:

- Teaching Assistant: Mehmet Burak Koca
- b.koca@gtu.edu.tr