- Welcome

- Introduction to HashMap and Set

**Agenda**

- Overview of HashMap and Set

- HashMap vs. Set

- Beginner-level concepts

- Basics concepts

**Overview of HashMap**

- Definition and purpose: HashMap is a data structure in Java that stores key-value pairs. It provides fast retrieval of values based on keys.

- Key-value pairs: Each element in a HashMap consists of a key and a corresponding value. The key is unique within the HashMap, and the value can be any object.

- Time complexity: The average time complexity for operations like put, get, and remove is O(1), making HashMap efficient for large datasets.

- Common methods: Examples of commonly used methods include put(key, value), get(key), and remove(key).

Example:

```

HashMap<String, Integer> studentGrades = new HashMap<>();

studentGrades.put("John", 85);

studentGrades.put("Alice", 92);

studentGrades.put("Bob", 78);

int aliceGrade = studentGrades.get("Alice");

System.out.println("Alice's grade: " + aliceGrade);

studentGrades.remove("John");

System.out.println("John's grade removed.");

```

**Overview of Set**

- Definition and purpose: Set is a collection in Java that stores unique elements. It ensures that no duplicates are present in the collection.

- Uniqueness and no duplication: Set guarantees that each element it contains is unique. It achieves this by using the equals() and hashCode() methods of the objects being stored.

- Time complexity: The time complexity for operations like add, remove, and contains is generally O(1) or O(log n), depending on the implementation.

- Common methods: Examples of commonly used methods include add(element), remove(element), and contains(element).

Example:

```

Set<String> fruitSet = new HashSet<>();

fruitSet.add("Apple");

fruitSet.add("Banana");

fruitSet.add("Orange");

fruitSet.add("Apple"); // Ignored since "Apple" already exists in the Set

System.out.println("Number of fruits in the set: " + fruitSet.size());

fruitSet.remove("Banana");

System.out.println("Banana removed from the set.");

boolean containsApple = fruitSet.contains("Apple");

System.out.println("Set contains Apple: " + containsApple);

```

**HashMap vs. Set**

- Differences and use cases: HashMap is used when we need to store and retrieve values based on unique keys. Set is used when we need to store a collection of unique elements without any specific associated values.

- HashMap: Ideal for scenarios where we want to associate a value with each key, such as storing student grades, mapping user IDs to names, etc.

- Set: Suitable for scenarios where we only care about uniqueness and don't need to associate any values, such as removing duplicates from a list, keeping track of unique visitors, etc.

**Basic concepts**

- Creating HashMap and Set objects: To create a HashMap or a Set, we declare a variable of the appropriate type and instantiate it using the respective class constructors.

- Adding elements to HashMap and Set: We can add elements to a HashMap using the put() method and to a Set using the add() method.

- Retrieving elements from HashMap and Set: We can retrieve elements from a HashMap using the get() method, which takes a key as an argument, and from a Set using iteration or the contains() method.

- Updating elements in HashMap and Set: Elements in a HashMap can be updated by using the put() method with an existing key. In Set, since the elements are unique, updating is done by removing the old element and adding the updated one.

- Removing elements from HashMap and Set: Elements can be removed from a HashMap using the remove() method with the corresponding key. In Set, elements can be removed using the remove() method with the element itself.

- Iterating over HashMap and Set: We can iterate over the elements of a HashMap using a for-each loop or by using an iterator. Similarly, we can iterate over a Set using a for-each loop or by using an iterator.

***https://pastebin.com/X7PEGXCX***

**Start to solve interesting problem 🧑‍💻**