Fetching an element from a HashMap

There are two ways to get an element from a HashMap.

Using the `get()` method

The `get(Object key)` method takes a key as a parameter and returns the value corresponding to that key. If the key is not present, it returns `null`.

Using the `getOrDefault()` method

The `getOrDefault(Object key, V defaultValue)` method is useful if are not sure whether a key is present in the Map or not. If the key is present then this method returns the value corresponding to the key and if the key is not present then the default value is returned.

```
import java.util.HashMap;
import java.util.Map;

public class HashMapDemo {

    public static void main(String args[]) {

        Map<String, Integer> stockPrice = new HashMap<>();

        stockPrice.put("Oracle", 56);
        stockPrice.put("Fiserv", 117);
        stockPrice.put("BMW", 73);
        stockPrice.put("Microsoft", 213);

        System.out.println(stockPrice.get("Oracle"));
        System.out.println(stockPrice.get("Google")); //This will return null.

        //Since Google is not present in our Map, this will insert it with default value of 100.
        System.out.println(stockPrice.getOrDefault("Google", 100));
    }
}
```

Run Save Reset

Replacing a value in HashMap

When we insert a key-value pair in **HashMap** and the key is already present then its value gets updated. But if we only want to update the value of a key that is present in the Map, then we can use the `replace()` method. There are two overloaded versions of the `replace()` method and one `replaceAll()` method. All three methods were added in Java 8.

Using the `replace(K key, V oldValue, V newValue)` method

The `replace(K key, V oldValue, V newValue)` method takes three parameters: the key, the old value, and a new value. It checks if the current value of the key is equal to the `oldValue` provided in the parameter. If yes then it replaces the value with `newValue` and returns `true`; otherwise, it returns `false`.

Using the `replace(K key, V value)` method

This method takes only two parameters: a key and a value. It replaces the value of the key with the new value provided as a parameter and returns the old value. If the key is not present, then it returns `null`.

Using the `replaceAll(BiFunction<? super K, ? super V, ? extends V> function)` method

This method takes a BiFunction as input and replaces the values of all the keys with the result of the given function. Suppose we need to add ten to the stock price of each company. Instead of updating the value for each stock one by one, we can use this method. The lambda expression to do this task will look like this:

```
(key, value) -> value + 10
```

```
import java.util.HashMap;
import java.util.Map;

public class HashMapDemo {

    public static void main(String args[]) {

        Map<String, Integer> stockPrice = new HashMap<>();

        stockPrice.put("Oracle", 56);
        stockPrice.put("Fiserv", 117);
        stockPrice.put("BMW", 73);
        stockPrice.put("Microsoft", 213);

        // This will not replace the value of Oracle because current value is not 70.
        stockPrice.replace("Oracle", 70, 76);
        System.out.println(stockPrice.get("Oracle"));

        // This will replace the value of Oracle because current value is 56.
        stockPrice.replace("Oracle", 56, 76);
        System.out.println(stockPrice.get("Oracle"));

        // This will replace the value of Fiserv as current value does not matter.
        stockPrice.replace("Fiserv", 100);
        System.out.println(stockPrice.get("Fiserv"));

        // Using the replaceAll() method to add 10 to the price of each stock.
        stockPrice.replaceAll((k,v) -> v + 10);
```

Run Save Reset

Removing an element from a HashMap

An element can be easily removed for the **HashMap** using the `remove(Object key)` method. It takes a key as a parameter and removes that key from the map. This method returns the value of the key that was removed. If the key is not present, then it returns `null`.

Another overloaded version of this method `remove(Object key, Object value)` was added in Java 8. This method removes a key only if it is currently mapped to the specified value. This method returns `true` if the key is removed; otherwise, it returns `false`.

```
import java.util.HashMap;
import java.util.Map;

public class HashMapDemo {

    public static void main(String args[]) {

        Map<String, Integer> stockPrice = new HashMap<>();

        stockPrice.put("Oracle", 56);
        stockPrice.put("Fiserv", 117);
        stockPrice.put("BMW", 73);
        stockPrice.put("Microsoft", 213);

        //This will remove Oracle from the Map and return 56.
        System.out.println(stockPrice.remove("Oracle"));

        //This will return null as Google is not present in the Map.
        System.out.println(stockPrice.remove("Google"));

        //This will return false because the value passed does not match the value of BMW in the Map.
        System.out.println(stockPrice.remove("BMW", 45));


    }
}
```

Run Save Reset