Map API Improvements: Replace Operations

This lessons explains the new methods for replacing the values that have been added to Map API.

We'll cover the following

In the previous lesson, we discussed a few new methods that have been added to the Map interface. In this lesson, we will look at some more improvements that have been done in Map API.

1. `replace()`, `replaceAll()`, and `remove()`

Sometimes we are required to change certain values from `Hashmap`. Before Java 8, the only way to do this was to iterate over the `Map` and change each value one by one.

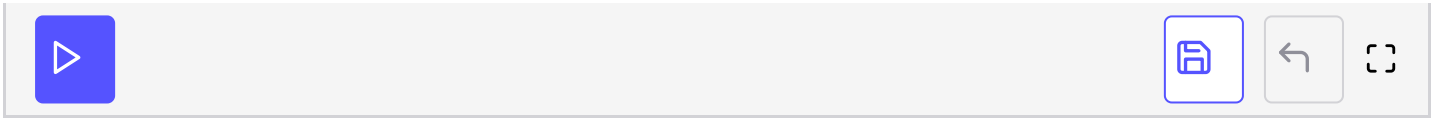
This is a cumbersome process, and it is prone to errors if the logic is not written properly. To overcome this issue, a few new methods have been introduced in Java 8.

a) `replace(K key, V value)`

This method replaces the entry for the specified key only if it is currently mapped to some value. If the key is not present or if the key is present but the value is null, then nothing is done.

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

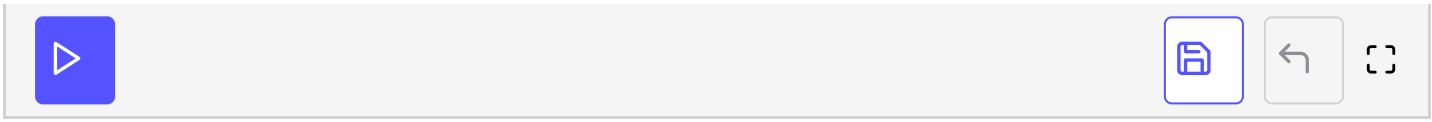
```
public class MapImprovements {

    public static void main(String args[]){
        Map<String, Integer> fruits = new HashMap<>();
        fruits.put("apple", 20);
        fruits.put("banana", 20);

        fruits.replace("apple", 50);

        System.out.println(fruits.get("apple"));
    }
}
```



b) `replace(K key, V oldValue, V newValue)`

This method replaces the entry for the specified key only if it iscurrently mapped to the specified value.

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

public class MapImprovements {

    public static void main(String args[]){
        Map<String, Integer> fruits = new HashMap<>();
        fruits.put("apple", 20);
        fruits.put("banana", 20);

        fruits.replace("apple", 30, 50);

        System.out.println(fruits.get("apple"));

        fruits.replace("apple", 20, 50);

        System.out.println(fruits.get("apple"));
    }
}
```

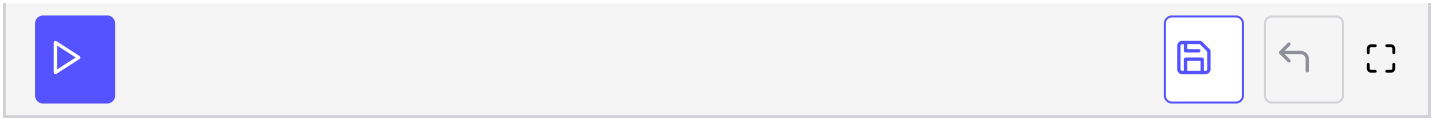


c) `replaceAll(BiFunction<? super K, ? super V, ? extends V> function)`

This method replaces each entry's value with the result of invoking the given function on that entry until all of theentries have been processed or the function throws an exception.

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

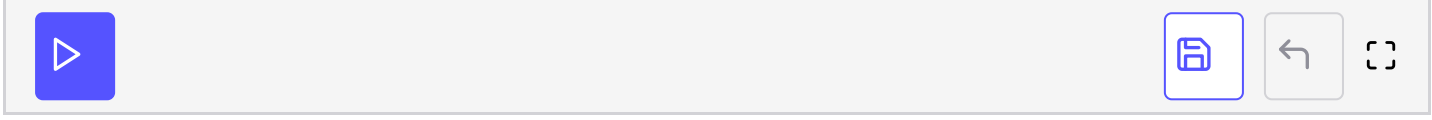
```
public class MapImprovements {

    public static void main(String args[]) {
        Map<String, Integer> fruits = new HashMap<>();
        fruits.put("apple", 20);
        fruits.put("banana", 20);

        fruits.replaceAll((k, v) -> 50);  //Value becomes 50 for all keys

        System.out.println(fruits.get("apple"));
        System.out.println(fruits.get("banana"));
    }
}
```



d) `remove(Object key)`

This method removes the mapping for a key from this map if it is present.

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

public class MapImprovements {

    public static void main(String args[]) {
        Map<String, Integer> fruits = new HashMap<>();
        fruits.put("apple", 20);
        fruits.put("banana", 20);

        fruits.remove("apple");  //apple will be removed

        System.out.println(fruits.get("apple"));
    }
}
```

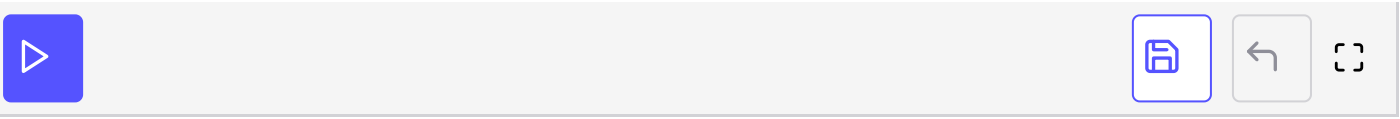


e) `remove(Object key, Object value)`

This method removes the entry for the specified key only if it is currently mapped to the specified value.

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

public class MapImprovements {

    public static void main(String args[]) {
        Map<String, Integer> fruits = new HashMap<>();
        fruits.put("apple", 20);
        fruits.put("banana", 20);
```

```
        fruits.remove("apple" , 30);  //apple will not be removed because the value is 20

        System.out.println(fruits.get("apple"));

        fruits.remove("apple" , 20);  //apple will be removed

        System.out.println(fruits.get("apple"));
    }
}
```



2) Iterating over the map using forEach()

Before Java 8, if you needed to iterate over a `HashMap`, then there were quite a few ways to do so, like getting the keyset or entry set. However, these methods were not very flexible and needed some practice to get a hold of.

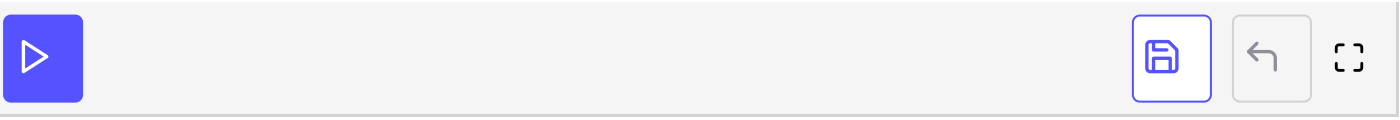
Now you can easily iterate over a `Map` using the `forEach()` method added in Java 8.

Here is the syntax of the `forEach()` method.

```
forEach(BiConsumer<? super K,? super V> action)
```

It takes a `BiConsumer` as a parameter.

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

public class MapImprovements {

    public static void main(String args[]) {
        Map<String, Integer> fruits = new HashMap<>();
        fruits.put("apple", 10);
        fruits.put("banana", 20);
        fruits.put("orange", 30);

        fruits.forEach((k,v) -> System.out.println("Key: " + k + " Value: " + v));

    }
}
```



In the next lesson, we will discuss the renewed comparator.