```
class RandomizedSet {

    Map<Integer, Integer> mp;
    List<Integer> arr;
    java.util.Random rand;
    /** Initialize your data structure here. */
    public RandomizedSet() {
        mp = new HashMap<>();
        arr = new ArrayList<>();
        rand = new Random();
    }

    /** Inserts a value to the set. Returns true if the set did not already contain the specified
element. */
    public boolean insert(int val) {
        if (!mp.containsKey(val)) {
            mp.put(val, arr.size());
            arr.add(val);
            return true;
        }
        return false;
    }

    /** Removes a value from the set. Returns true if the set contained the specified element. */
    public boolean remove(int val) {
        if (mp.containsKey(val)) {
            if (mp.get(val) < arr.size()-1) {
                int last = arr.get(arr.size() - 1);
                arr.set(mp.get(val), last);
                mp.put(last, mp.get(val));
            }
            mp.remove(val);
            arr.remove(arr.size() - 1);
            return true;
        }
        return false;
    }

    /** Get a random element from the set. */
    public int getRandom() {
        int idx = rand.nextInt(arr.size());
        return arr.get(idx);
    }
}
```

Design a data structure that supports all following operations in *average* **O(1)** time

1. `insert(val)`: Inserts an item val to the set if not already present.
2. `remove(val)`: Removes an item val from the set if present.
3. `getRandom`: Returns a random element from current set of elements. Each element must have the **same probability** of being returned.