JAVA - THE IDENTITYHASHMAP CLASS

This class implements AbstractMap. It is similar to HashMap except that it uses reference equality when comparing elements.

This class is not a general-purpose Map implementation. While this class implements the Map interface, it intentionally violates Map's general contract, which mandates the use of the equals method when comparing objects.

This class is designed for use only in the rare cases wherein reference-equality semantics are required.

This class provides constant-time performance for the basic operations (get and put), assuming the system identity hash function (System.identityHashCode(Object)) disperses elements properly among the buckets.

This class has one tuning parameter (which affects performance but not semantics): expected maximum size. This parameter is the maximum number of key-value mappings that the map is expected to hold.

The IdentityHashMap class supports three constructors. The first form constructs a new, empty identity hash map with a default expected maximum size (21):

```
IdentityHashMap()
```

The second form constructs a new, empty map with the specified expected maximum size:

```
IdentityHashMap(int expectedMaxSize)
```

The third form constructs a new identity hash map containing the keys-value mappings in the specified map:

```
IdentityHashMap(Map m)
```

Apart from the methods inherited from its parent classes, IdentityHashMap defines following methods:

SN	Methods with Description
1	**void clear()** Removes all mappings from this map.
2	**Object clone()** Returns a shallow copy of this identity hash map: the keys and values themselves are not cloned.
3	**boolean containsKey(Object key)** Tests whether the specified object reference is a key in this identity hash map.
4	**boolean containsValue(Object value)** Tests whether the specified object reference is a value in this identity hash map.
5	**Set entrySet()**

Returns a set view of the mappings contained in this map.

6 **boolean equals(Object o)**

Compares the specified object with this map for equality.

7 **Object get(Object key)**

Returns the value to which the specified key is mapped in this identity hash map, or null if the map contains no mapping for this key.

8 **int hashCode()**

Returns the hash code value for this map.

9 **boolean isEmpty()**

Returns true if this identity hash map contains no key-value mappings.

10 **Set keySet()**

Returns an identity-based set view of the keys contained in this map.

11 **Object put(Object key, Object value)**

Associates the specified value with the specified key in this identity hash map.

12 **void putAll(Map t)**

Copies all of the mappings from the specified map to this map These mappings will replace any mappings that this map had for any of the keys currently in the specified map.

13 **Object remove(Object key)**

Removes the mapping for this key from this map if present.

14 **int size()**

Returns the number of key-value mappings in this identity hash map.

15 **Collection values()**

Returns a collection view of the values contained in this map.

Example:

The following program illustrates several of the methods supported by this collection:

```
import java.util.*;

public class IdentityHashMapDemo {

   public static void main(String args[]) {
      // Create a hash map
      IdentityHashMap ihm = new IdentityHashMap();
      // Put elements to the map
      ihm.put("Zara", new Double(3434.34));
      ihm.put("Mahnaz", new Double(123.22));
      ihm.put("Ayan", new Double(1378.00));
      ihm.put("Daisy", new Double(99.22));
      ihm.put("Qadir", new Double(-19.08));

      // Get a set of the entries
      Set set = ihm.entrySet();
      // Get an iterator
      Iterator i = set.iterator();
      // Display elements
      while(i.hasNext()) {
         Map.Entry me = (Map.Entry)i.next();
         System.out.print(me.getKey() + ": ");
         System.out.println(me.getValue());
      }
      System.out.println();
      // Deposit 1000 into Zara's account
      double balance = ((Double)ihm.get("Zara")).doubleValue();
      ihm.put("Zara", new Double(balance + 1000));
      System.out.println("Zara's new balance: " +
      ihm.get("Zara"));
   }
}
```

This would produce the following result:

```
Ayan: 1378.0
Zara: 3434.34
Qadir: -19.08
Mahnaz: 123.22
Daisy: 99.22

Zara's new balance: 4434.34
```