Contents

HashMap and Object Class

HashMap

Implementation:
A hash table-based implementation of the Map interface.

Functionality: Stores key-value pairs. Keys are used for fast access and retrieval of values.

Key Features:

1. **Unique Keys:** Keys must be unique within the HashMap.
2. **Hashing:** Keys are hashed to determine their location in the HashMap.
3. **Collisions:** If two keys hash to the same location (collision), a collision resolution mechanism (e.g., chaining) is used.
4. **Performance:** Offers fast average-case access time (O(1)), but may degrade with frequent collisions.
5. **Functions:**
    a. HashMap(int initialCapacity, float loadFactor): Constructor to initialise the HashMap with a specific initial capacity and load factor (threshold for resizing).
    Link:
    https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/
    b. **put(K key, V value):** Adds a key-value pair to the HashMap. If the key already exists, the value is replaced.
    Link:
    https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/

c. **get(Object key)**: Returns the value associated with the given key, or null if the key is not present.
Link:
https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/

d. **remove(Object key)**: Removes the key-value pair associated with the given key and returns the value. Returns null if the key is not present.
Link:
https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/

e. **containsKey(Object key)**: Checks if the HashMap contains the given key.
Link:
https://www.geeksforgeeks.org/hashmap-containskey-method-in-java/

f. **containsValue(Object value)**: Checks if the HashMap contains the given value.
Link:
https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/

g. **size()**: Returns the number of key-value pairs in the HashMap.
Link:
https://www.geeksforgeeks.org/hashmap-size-method-in-java/

h. **isEmpty()**: Checks if the HashMap is empty.
Link:
https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/

i. **entrySet()**: Returns a set of all key-value mappings in the HashMap.
Link:
https://www.geeksforgeeks.org/hashmap-entryset-method-in-java/

j. **keySet()**: Returns a set of all keys in the HashMap.
Link:

https://www.geeksforgeeks.org/hashmap-keyset-method-in-java/

k. **values():** Returns a collection of all values in the HashMap.
Link:
https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/

Object Class

Fundamentals:
Base class for all Java objects.

Functionality:
Provides essential methods for object behaviour.

Key Functions:

1. **equals(Object obj):** Compares two objects for equality. Subclasses can override this method to define their own equality criteria.
https://www.geeksforgeeks.org/equals-hashcode-methods-java/

2. **hashCode():** Generates a hash code for the object. Used for hashing-based collections like HashMap. Subclasses can override this method to define a custom hash function.
https://www.geeksforgeeks.org/importance-hashcode-method-java/

3. **toString():** Returns a String representation of the object. Can be overridden by subclasses for a more informative representation.
https://www.geeksforgeeks.org/equals-hashcode-methods-java/

4. **finalize():** Garbage collection method. Invoked by the garbage collector when an object is about to be reclaimed. Generally not

recommended to override for most use cases.
https://www.geeksforgeeks.org/finalize-method-in-java-and-how-to-override-it/

Additional Resources:

1. **HashMap:**
   https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/

2. **Object Class:**
   https://www.geeksforgeeks.org/understanding-classes-and-objects-in-java/