# Java HashMap

[❮ Previous](https://www.w3schools.com/java/java_arraylist.asp)[Next ❯](https://www.w3schools.com/java/java_wrapper_classes.asp)

## **Java HashMap**

You learned from the previous chapter, that Arrays store items as an ordered collection, and you have to access them with an index number (int type). A HashMap however, store items in "**key**/**value**" pairs, and you can access them by an index of another type (e.g. a String).

One object is used as a key (index) to another object (value). It can store different types: String keys and Integer values, or the same type, like: String keys and String values:

### **Example**

Create a HashMap object called **capitalCities** that will store String **keys** and String **values**:

import java.util.HashMap; // import the HashMap class

HashMap<String, String> capitalCities = new HashMap<String, String>();

## **Add Items**

The HashMap class has many useful methods. For example, to add items to it, use the put() method:

### **Example**

// Import the HashMap class

import java.util.HashMap;

public class MyClass {

public static void main(String[] args) {

// Create a HashMap object called capitalCities

HashMap<String, String> capitalCities = new HashMap<String, String>();

// Add keys and values (Country, City)

capitalCities.put("England", "London");

capitalCities.put("Germany", "Berlin");

capitalCities.put("Norway", "Oslo");

capitalCities.put("USA", "Washington DC");

System.out.println(capitalCities);

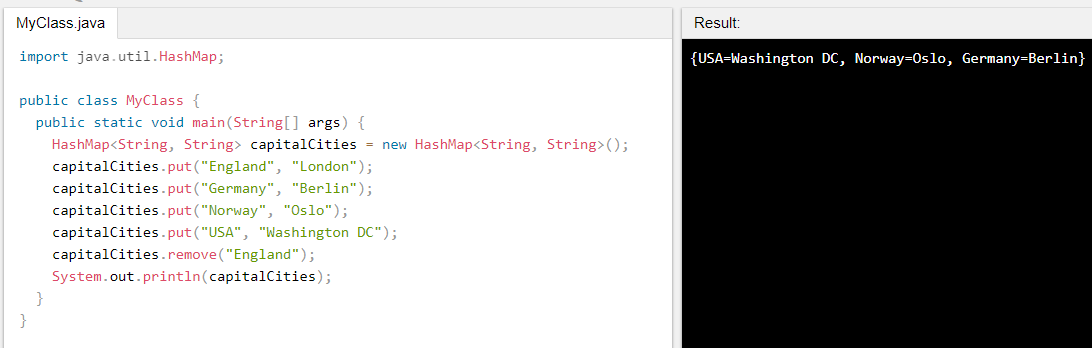
}

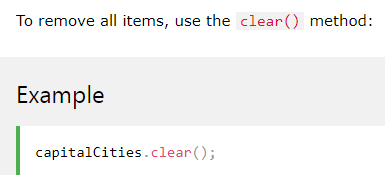
}

{USA=Washington DC, Norway=Oslo, England=London, Germany=Berlin}











**HashMap put() Method in Java**

The java.util.HashMap.put() method of HashMap is used to insert a mapping into a map. This means we can insert a specific key and the value it is mapping to into a particular map. If an existing key is passed then the previous value gets replaced by the new value. If a new pair is passed, then the pair gets inserted as a whole.

**Syntax:**

Hash\_Map.put(*key, value*)

**Parameters:** The method takes two parameters, both are of the Object type of the HashMap.

* *key:* This refers to the key element that needs to be inserted into the Map for mapping.
* *value:* This refers to the value that the above key would map into.

**Return Value:** If an existing key is passed then the previous value gets returned. If a new pair is passed, then NULL is returned.

Below programs are used to illustrate the working of java.util.HashMap.put() Method:  
**Program 1:** When passing an existing key.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the put() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<Integer, String> hash\_map = new HashMap<Integer, String>();            // Mapping string values to int keys          hash\_map.put(10, "Geeks");          hash\_map.put(15, "4");          hash\_map.put(20, "Geeks");          hash\_map.put(25, "Welcomes");          hash\_map.put(30, "You");            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Inserting existing key along with new value          String returned\_value = (String)hash\_map.put(20, "All");            // Verifying the returned value          System.out.println("Returned value is: " + returned\_value);            // Displayin the new map          System.out.println("New map is: " + hash\_map);      }  } |

**Output:**

Initial Mappings are: {20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

Returned value is: Geeks

New map is: {20=All, 25=Welcomes, 10=Geeks, 30=You, 15=4}

**Program 2:** When passing a new key.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the put() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<Integer, String> hash\_map = new HashMap<Integer, String>();            // Mapping string values to int keys          hash\_map.put(10, "Geeks");          hash\_map.put(15, "4");          hash\_map.put(20, "Geeks");          hash\_map.put(25, "Welcomes");          hash\_map.put(30, "You");            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Inserting existing key along with new value          String returned\_value = (String)hash\_map.put(50, "All");            // Verifying the returned value          System.out.println("Returned value is: " + returned\_value);            // Displayin the new map          System.out.println("New map is: " + hash\_map);      }  } |

**Output:**

Initial Mappings are: {20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

Returned value is: null

New map is: {50=All, 20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

**Note:** The same operation can be performed with any type of Mappings with variation and combination of different data types.

# HashMap get() Method in Java

The java.util.HashMap.get() method of HashMap class is used to retrieve or fetch the value mapped by a particular key mentioned in the parameter. It returns NULL when the map contains no such mapping for the key.

**Syntax:**

Hash\_Map.get(Object key\_element)

**Parameter:** The method takes one parameter key\_element of object type and refers to the key whose associated value is supposed to be fetched.

**Return Value:** The method returns the value associated with the key\_element in the parameter.

Below programs illustrates the working of java.util.HashMap.get() method:  
**Program 1:** Mapping String Values to Integer Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the get() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<Integer, String> hash\_map = new HashMap<Integer, String>();            // Mapping string values to int keys          hash\_map.put(10, "Geeks");          hash\_map.put(15, "4");          hash\_map.put(20, "Geeks");          hash\_map.put(25, "Welcomes");          hash\_map.put(30, "You");            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Getting the value of 25          System.out.println("The Value is: " + hash\_map.get(25));            // Getting the value of 10          System.out.println("The Value is: " + hash\_map.get(10));      }  } |

**Output:**

Initial Mappings are: {20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

The Value is: Welcomes

The Value is: Geeks

**Program 2:** Mapping Integer Values to String Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the get() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<String, Integer> hash\_map = new HashMap<String, Integer>();            // Mapping int values to string keys          hash\_map.put("Geeks", 10);          hash\_map.put("4", 15);          hash\_map.put("Geeks", 20);          hash\_map.put("Welcomes", 25);          hash\_map.put("You", 30);            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Getting the value of "Geeks"          System.out.println("The Value is: " + hash\_map.get("Geeks"));            // Getting the value of "You"          System.out.println("The Value is: " + hash\_map.get("You"));      }  } |

**Output:**

Initial Mappings are: {4=15, Geeks=20, You=30, Welcomes=25}

The Value is: 20

The Value is: 30

**Note:** The same operation can be performed with any type of Mappings with variation and combination of different data types.

# HashMap putAll() Method in Java

The java.util.HashMap.putAll() is an inbuilt method of HashMap class that is used for the copy operation. The method copies all of the elements i.e., the mappings, from one map into another.

**Syntax:**

new\_hash\_map.putAll(exist\_hash\_map)

**Parameters:** The method takes one parameter exist\_hash\_map that refers to the existing map we want to copy from.

**Return Value:** The method does not return any values.

**Exception:** The method throws NullPointerException if the map we want to copy from is NULL.

Below programs illustrates the working of java.util.HashMap.putAll() method:  
**Program 1:** Mapping String Values to Integer Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the putAll() method  import java.util.\*;    public class Hash\_Map\_Demo {  public static void main(String[] args) {        // Creating an empty HashMap      HashMap<Integer, String> hash\_map = new HashMap<Integer, String>();        // Mapping string values to int keys      hash\_map.put(10, "Geeks");      hash\_map.put(15, "4");      hash\_map.put(20, "Geeks");      hash\_map.put(25, "Welcomes");      hash\_map.put(30, "You");        // Displaying the HashMap      System.out.println("Initial Mappings are: " + hash\_map);        // Creating a new hash map and copying      HashMap<Integer, String> new\_hash\_map = new HashMap<Integer, String>();      new\_hash\_map.putAll(hash\_map);        // Displaying the final HashMap      System.out.println("The new map looks like this: " + new\_hash\_map);  }  } |

**Output:**

Initial Mappings are: {20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

The new map looks like this: {25=Welcomes, 10=Geeks, 20=Geeks, 30=You, 15=4}

**Program 2:** Mapping Integer Values to String Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the putAll() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<String, Integer> hash\_map = new HashMap<String, Integer>();            // Mapping int values to string keys          hash\_map.put("Geeks", 10);          hash\_map.put("4", 15);          hash\_map.put("Geeks", 20);          hash\_map.put("Welcomes", 25);          hash\_map.put("You", 30);            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Creating a new hash map and copying          HashMap<String, Integer> new\_hash\_map = new HashMap<String, Integer>();          new\_hash\_map.putAll(hash\_map);            // Displaying the final HashMap          System.out.println("The new map looks like this: " + new\_hash\_map);      }  } |

**Output:**

Initial Mappings are: {4=15, Geeks=20, You=30, Welcomes=25}

The new map looks like this: {Geeks=20, 4=15, You=30, Welcomes=25}

**Note:** The same operation can be performed with any type of Mappings with variation and combination of different data types.

# HashMap keySet() Method in Java

The java.util.HashMap.keySet() method in Java is used to create a set out of the key elements contained in the hash map. It basically returns a set view of the keys or we can create a new set and store the key elements in them.

**Syntax:**

hash\_map.keySet()

**Parameters:** The method does not take any parameter.

**Return Value:** The method returns a set having the keys of the hash map.

Below programs are used to illustrate the working of java.util.HashMap.keySet() Method:  
**Program 1:** Mapping String Values to Integer Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the keySet() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<Integer, String> hash\_map = new HashMap<Integer, String>();            // Mapping string values to int keys          hash\_map.put(10, "Geeks");          hash\_map.put(15, "4");          hash\_map.put(20, "Geeks");          hash\_map.put(25, "Welcomes");          hash\_map.put(30, "You");            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Using keySet() to get the set view of keys          System.out.println("The set is: " + hash\_map.keySet());      }  } |

**Output:**

Initial Mappings are: {20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

The set is: [20, 25, 10, 30, 15]

**Program 2:** Mapping Integer Values to String Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the keySet() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<String, Integer> hash\_map = new HashMap<String, Integer>();            // Mapping int values to string keys          hash\_map.put("Geeks", 10);          hash\_map.put("4", 15);          hash\_map.put("Geeks", 20);          hash\_map.put("Welcomes", 25);          hash\_map.put("You", 30);            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Using keySet() to get the set view of keys          System.out.println("The set is: " + hash\_map.keySet());      }  } |

**Output:**

Initial Mappings are: {4=15, Geeks=20, You=30, Welcomes=25}

The set is: [4, Geeks, You, Welcomes]

**Note:** The same operation can be performed with any type of Mappings with variation and combination of different data types.

# HashMap values() Method in Java

The java.util.HashMap.values() method of HashMap class in Java is used to create a collection out of the values of the map. It basically returns a Collection view of the values in the HashMap.

**Syntax:**

Hash\_Map.values()

**Parameters:** The method does not accept any parameters.

**Return Value:** The method is used to return a collection view containing all the values of the map.

Below programs are used to illustrate the working of java.util.HashMap.values() Method:  
**Program 1:** Mapping String Values to Integer Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the values() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<Integer, String> hash\_map = new HashMap<Integer, String>();            // Mapping string values to int keys          hash\_map.put(10, "Geeks");          hash\_map.put(15, "4");          hash\_map.put(20, "Geeks");          hash\_map.put(25, "Welcomes");          hash\_map.put(30, "You");            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Using values() to get the set view of values          System.out.println("The collection is: " + hash\_map.values());      }  } |

**Output:**

Initial Mappings are: {20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

The collection is: [Geeks, Welcomes, Geeks, You, 4]

**Program 2:** Mapping Integer Values to String Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the values() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<String, Integer> hash\_map = new HashMap<String, Integer>();            // Mapping int values to string keys          hash\_map.put("Geeks", 10);          hash\_map.put("4", 15);          hash\_map.put("Geeks", 20);          hash\_map.put("Welcomes", 25);          hash\_map.put("You", 30);            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Using values() to get the set view of values          System.out.println("The collection is: " + hash\_map.values());      }  } |

**Output:**

Initial Mappings are: {4=15, Geeks=20, You=30, Welcomes=25}

The collection is: [15, 20, 30, 25]

**Note:** The same operation can be performed with any type of Mappings with variation and combination of different data types.

# HashMap size() Method in Java

The java.util.HashMap.size() method of HashMap class is used to get the size of the map which refers to the number of the key-value pair or mappings in the Map.

**Syntax:**

Hash\_Map.size()

**Parameters:** The method does not take any parameters.

**Return Value:** The method returns the size of the map which also means the number of key-value pairs present in the map.

Below programs illustrates the working of java.util.HashMap.size():  
Program 1: Mapping String Values to Integer Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the size() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<Integer, String> hash\_map = new HashMap<Integer, String>();            // Mapping string values to int keys          hash\_map.put(10, "Geeks");          hash\_map.put(15, "4");          hash\_map.put(20, "Geeks");          hash\_map.put(25, "Welcomes");          hash\_map.put(30, "You");            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Displaying the size of the map          System.out.println("The size of the map is " + hash\_map.size());      }  } |

**Output:**

Initial Mappings are: {20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

The size of the map is 5

**Program 2:** Mapping Integer Values to String Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the size() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<String, Integer> hash\_map = new HashMap<String, Integer>();            // Mapping int values to string keys          hash\_map.put("Geeks", 10);          hash\_map.put("4", 15);          hash\_map.put("Geeks", 20);          hash\_map.put("Welcomes", 25);          hash\_map.put("You", 30);            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Displaying the size of the map          System.out.println("The size of the map is " + hash\_map.size());      }  } |

**Output:**

Initial Mappings are: {4=15, Geeks=20, You=30, Welcomes=25}

The size of the map is 4

**Note:** The same operation can be performed with any type of Mappings with variation and combination of different data types.

# HashMap entrySet() Method in Java

The java.util.HashMap.entrySet() method in Java is used to create a set out of the same elements contained in the hash map. It basically returns a set view of the hash map or we can create a new set and store the map elements into them.

**Syntax:**

hash\_map.entrySet()

**Parameters:** The method does not take any parameter.

**Return Value:** The method returns a set having same elements as the hash map.

Below programs are used to illustrate the working of java.util.HashMap.entrySet() Method:  
**Program 1:** Mapping String Values to Integer Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the entrySet() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<Integer, String> hash\_map = new HashMap<Integer, String>();            // Mapping string values to int keys          hash\_map.put(10, "Geeks");          hash\_map.put(15, "4");          hash\_map.put(20, "Geeks");          hash\_map.put(25, "Welcomes");          hash\_map.put(30, "You");            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Using entrySet() to get the set view          System.out.println("The set is: " + hash\_map.entrySet());      }  } |

**Output:**

Initial Mappings are: {20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

The set is: [20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4]

**Program 2:** Mapping Integer Values to String Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the entrySet() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<String, Integer> hash\_map = new HashMap<String, Integer>();            // Mapping int values to string keys          hash\_map.put("Geeks", 10);          hash\_map.put("4", 15);          hash\_map.put("Geeks", 20);          hash\_map.put("Welcomes", 25);          hash\_map.put("You", 30);            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Using entrySet() to get the set view          System.out.println("The set is: " + hash\_map.entrySet());      }  } |

**Output:**

Initial Mappings are: {4=15, Geeks=20, You=30, Welcomes=25}

The set is: [4=15, Geeks=20, You=30, Welcomes=25]

**Note:** The same operation can be performed with any type of Mappings with variation and combination of different data types.

# HashMap size() Method in Java

The java.util.HashMap.size() method of HashMap class is used to get the size of the map which refers to the number of the key-value pair or mappings in the Map.

**Syntax:**

Hash\_Map.size()

**Parameters:** The method does not take any parameters.

**Return Value:** The method returns the size of the map which also means the number of key-value pairs present in the map.

Below programs illustrates the working of java.util.HashMap.size():  
Program 1: Mapping String Values to Integer Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the size() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<Integer, String> hash\_map = new HashMap<Integer, String>();            // Mapping string values to int keys          hash\_map.put(10, "Geeks");          hash\_map.put(15, "4");          hash\_map.put(20, "Geeks");          hash\_map.put(25, "Welcomes");          hash\_map.put(30, "You");            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Displaying the size of the map          System.out.println("The size of the map is " + hash\_map.size());      }  } |

**Output:**

Initial Mappings are: {20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

The size of the map is 5

**Program 2:** Mapping Integer Values to String Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the size() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<String, Integer> hash\_map = new HashMap<String, Integer>();            // Mapping int values to string keys          hash\_map.put("Geeks", 10);          hash\_map.put("4", 15);          hash\_map.put("Geeks", 20);          hash\_map.put("Welcomes", 25);          hash\_map.put("You", 30);            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Displaying the size of the map          System.out.println("The size of the map is " + hash\_map.size());      }  } |

**Output:**

Initial Mappings are: {4=15, Geeks=20, You=30, Welcomes=25}

The size of the map is 4

**Note:** The same operation can be performed with any type of Mappings with variation and combination of different data types.

**HashMap containsValue() Method in Java**

The java.util.HashMap.containsValue() method is used to check whether a particular value is being mapped by a single or more than one key in the HashMap. It takes the Value as a parameter and returns True if that value is mapped by any of the key in the map.

**Syntax:**

Hash\_Map.containsValue(*Object Value*)

**Parameters:** The method takes just one parameter *Value* of Object type and refers to the value whose mapping is supposed to be checked by any key inside the map.

**Return Value:** The method returns boolean true if the mapping of the value is detected else false.

Below programs are used to illustrate the working of java.util.HashMap.containsValue() Method:  
**Program 1:** Mapping String Values to Integer Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the containsValue() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<Integer, String> hash\_map = new HashMap<Integer, String>();            // Mapping string values to int keys          hash\_map.put(10, "Geeks");          hash\_map.put(15, "4");          hash\_map.put(20, "Geeks");          hash\_map.put(25, "Welcomes");          hash\_map.put(30, "You");            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Checking for the Value 'Geeks'          System.out.println("Is the value 'Geeks' present? " +          hash\_map.containsValue("Geeks"));            // Checking for the Value 'World'          System.out.println("Is the value 'World' present? " +          hash\_map.containsValue("World"));      }  } |

**Output:**

Initial Mappings are: {20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

Is the value 'Geeks' present? true

Is the value 'World' present? false

**Program 2:** Mapping Integer Values to String Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the containsValue() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<String, Integer> hash\_map = new HashMap<String, Integer>();            // Mapping int values to string keys          hash\_map.put("Geeks", 10);          hash\_map.put("4", 15);          hash\_map.put("Geeks", 20);          hash\_map.put("Welcomes", 25);          hash\_map.put("You", 30);            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Checking for the Value '10'          System.out.println("Is the value '10' present? " +          hash\_map.containsValue(10));            // Checking for the Value '30'          System.out.println("Is the value '30' present? " +          hash\_map.containsValue(30));            // Checking for the Value '40'          System.out.println("Is the value '40' present? " +          hash\_map.containsValue(40));      }  } |

**Output:**

Initial Mappings are: {4=15, Geeks=20, You=30, Welcomes=25}

Is the value '10' present? false

Is the value '30' present? true

Is the value '40' present? false

**Time Complexity :** O(n)

**Note:** The same operation can be performed with any type of Mappings with variation and combination of different data types.

**HashMap containsKey() Method in Java**

The java.util.HashMap.containsKey() method is used to check whether a particular key is being mapped into the HashMap or not. It takes the key element as a parameter and returns True if that element is mapped in the map.

**Syntax:**

Hash\_Map.containsKey(*key\_element*)

**Parameters:** The method takes just one parameter *key\_element* that refers to the key whose mapping is supposed to be checked inside a map.

**Return Value:** The method returns boolean true if the presence of the key is detected else false .

Below programs are used to illustrate the working of java.util.HashMap.containsKey() Method:  
**Program 1:** Mapping String Values to Integer Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the containsKey() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<Integer, String> hash\_map = new HashMap<Integer, String>();            // Mapping string values to int keys          hash\_map.put(10, "Geeks");          hash\_map.put(15, "4");          hash\_map.put(20, "Geeks");          hash\_map.put(25, "Welcomes");          hash\_map.put(30, "You");            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Checking for the key\_element '20'          System.out.println("Is the key '20' present? " +          hash\_map.containsKey(20));            // Checking for the key\_element '5'          System.out.println("Is the key '5' present? " +          hash\_map.containsKey(5));      }  } |

**Output:**

Initial Mappings are: {20=Geeks, 25=Welcomes, 10=Geeks, 30=You, 15=4}

Is the key '20' present? true

Is the key '5' present? false

**Program 2:** Mapping Integer Values to String Keys.

*filter\_none*

*edit*

*play\_arrow*

*brightness\_4*

|  |
| --- |
| // Java code to illustrate the containsKey() method  import java.util.\*;    public class Hash\_Map\_Demo {      public static void main(String[] args)      {            // Creating an empty HashMap          HashMap<String, Integer> hash\_map = new HashMap<String, Integer>();            // Mapping int values to string keys          hash\_map.put("Geeks", 10);          hash\_map.put("4", 15);          hash\_map.put("Geeks", 20);          hash\_map.put("Welcomes", 25);          hash\_map.put("You", 30);            // Displaying the HashMap          System.out.println("Initial Mappings are: " + hash\_map);            // Checking for the key\_element 'Welcomes'          System.out.println("Is the key 'Welcomes' present? " +          hash\_map.containsKey("Welcomes"));            // Checking for the key\_element 'World'          System.out.println("Is the key 'World' present? " +          hash\_map.containsKey("World"));      }  } |

**Output:**

Initial Mappings are: {4=15, Geeks=20, You=30, Welcomes=25}

Is the key 'Welcomes' present? true

Is the key 'World' present? false

**Note:** The same operation can be performed with any type of Mappings with variation and combination of different data types.