



Java HashMap

[< Previous](#)[Next >](#)

Java HashMap

In the [ArrayList](#) chapter, you learned 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 Main {
    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);
    }
}
```

Try it Yourself »

Access an Item

To access a value in the `HashMap`, use the `get()` method and refer to its key:

Example

```
capitalCities.get("England");
```

Try it Yourself »

Remove an Item

To remove an item, use the `remove()` method and refer to the key:

Example

```
capitalCities.remove("England");
```

Try it Yourself »

To remove all items, use the `clear()` method:

Example

```
capitalCities.clear();
```

Try it Yourself »

ADVERTISEMENT



HashMap Size

To find out how many items there are, use the `size()` method:

Example

```
capitalCities.size();
```

Try it Yourself »

Loop Through a HashMap

Loop through the items of a `HashMap` with a **for-each** loop.

Note: Use the `keySet()` method if you only want the keys, and use the `values()` method if you only want the values:

Example

```
// Print keys
for (String i : capitalCities.keySet()) {
    System.out.println(i);
}
```

Try it Yourself »

Example

```
// Print values
for (String i : capitalCities.values()) {
    System.out.println(i);
}
```

Try it Yourself »

Example

```
// Print keys and values
for (String i : capitalCities.keySet()) {
    System.out.println("key: " + i + " value: " + capitalCities.get(i));
}
```

Try it Yourself »

Other Types

Keys and values in a HashMap are actually objects. In the examples above, we used objects of type "String". Remember that a String in Java is an object (not a primitive type). To use other types, such as int, you must specify an equivalent wrapper class:

Integer . For other primitive types, use: **Boolean** for boolean, **Character** for char, **Double** for double, etc:

Example

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

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

public class Main {
    public static void main(String[] args) {

        // Create a HashMap object called people
        HashMap<String, Integer> people = new HashMap<String, Integer>();

        // Add keys and values (Name, Age)
        people.put("John", 32);
        people.put("Steve", 30);
        people.put("Angie", 33);

        for (String i : people.keySet()) {
            System.out.println("key: " + i + " value: " + people.get(i));
        }
    }
}
```

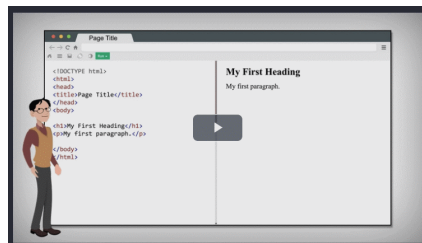
Try it Yourself »

[< Previous](#)[Next >](#)

ADVERTISEMENT

NEW

We just launched
W3Schools videos



Explore now

COLOR PICKER



Get certified
by completing
a Java
course today!



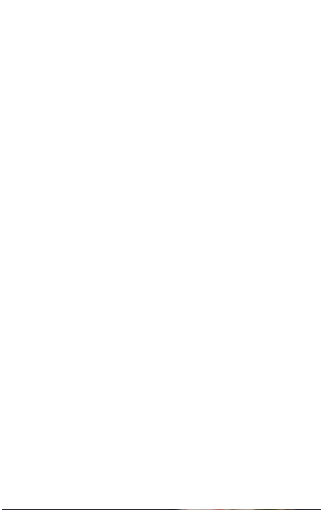
Get started

CODE GAME



Play Game

ADVERTISEMENT



ADVERTISEMENT

ADVERTISEMENT



[Report Error](#)

[Spaces](#)

[Pro](#)

[Buy Certificate](#)

Top Tutorials

[HTML Tutorial](#)
[CSS Tutorial](#)
[JavaScript Tutorial](#)
[How To Tutorial](#)
[SQL Tutorial](#)
[Python Tutorial](#)
[W3.CSS Tutorial](#)
[Bootstrap Tutorial](#)

[PHP Tutorial](#)
[Java Tutorial](#)
[C++ Tutorial](#)
[jQuery Tutorial](#)

Top References

[HTML Reference](#)
[CSS Reference](#)
[JavaScript Reference](#)
[SQL Reference](#)
[Python Reference](#)
[W3.CSS Reference](#)
[Bootstrap Reference](#)
[PHP Reference](#)
[HTML Colors](#)
[Java Reference](#)
[Angular Reference](#)
[jQuery Reference](#)

Top Examples

[HTML Examples](#)
[CSS Examples](#)
[JavaScript Examples](#)
[How To Examples](#)
[SQL Examples](#)
[Python Examples](#)
[W3.CSS Examples](#)
[Bootstrap Examples](#)
[PHP Examples](#)
[Java Examples](#)
[XML Examples](#)
[jQuery Examples](#)

Get Certified

[HTML Certificate](#)
[CSS Certificate](#)
[JavaScript Certificate](#)
[Front End Certificate](#)
[SQL Certificate](#)
[Python Certificate](#)
[PHP Certificate](#)
[jQuery Certificate](#)
[Java Certificate](#)
[C++ Certificate](#)
[C# Certificate](#)
[XML Certificate](#)

[FORUM](#) | [ABOUT](#)

W3Schools is optimized for learning and training. Examples might be simplified to improve reading and learning. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant

full correctness of all content. While using W3Schools, you agree to have read and accepted our terms of use, cookie and privacy policy.

Copyright 1999-2022 by Refsnes Data. All Rights Reserved.
W3Schools is Powered by W3.CSS.

