**Java HashMap**

HashMap 是一个散列表，它存储的内容是键值对(key-value)映射。

HashMap 实现了 Map 接口，根据键的 HashCode 值存储数据，具有很快的访问速度，最多允许一条记录的键为 null，不支持线程同步。

HashMap 是无序的，即不会记录插入的顺序。

HashMap 继承于AbstractMap，实现了 Map、Cloneable、java.io.Serializable 接口。

HashMap 的 key 与 value 类型可以相同也可以不同，可以是字符串（String）类型的 key 和 value，也可以是整型（Integer）的 key 和字符串（String）类型的 value。

HashMap 中的元素实际上是对象，一些常见的基本类型可以使用它的包装类。

基本类型对应的包装类表如下：

基本类型 引用类型

boolean Boolean

byte Byte

short Short

int Integer

long Long

float Float

double Double

char Character

HashMap 类位于 java.util 包中，使用前需要引入它，语法格式如下：

import java.util.HashMap; // 引入 HashMap 类

以下实例我们创建一个 HashMap 对象 Sites， 整型（Integer）的 key 和字符串（String）类型的 value：

HashMap<Integer, String> Sites = new HashMap<Integer, String>();

添加元素

HashMap 类提供了很多有用的方法，添加键值对(key-value)可以使用 put() 方法:

实例

// 引入 HashMap 类

import java.util.HashMap;

public class RunoobTest {

public static void main(String[] args) {

// 创建 HashMap 对象 Sites

HashMap<Integer, String> Sites = new HashMap<Integer, String>();

// 添加键值对

Sites.put(1, "Google");

Sites.put(2, "Runoob");

Sites.put(3, "Taobao");

Sites.put(4, "Zhihu");

System.out.println(Sites);

}

}

执行以上代码，输出结果如下：

{1=Google, 2=Runoob, 3=Taobao, 4=Zhihu}

以下实例创建一个字符串（String）类型的 key 和字符串（String）类型的 value：

实例

// 引入 HashMap 类

import java.util.HashMap;

public class RunoobTest {

public static void main(String[] args) {

// 创建 HashMap 对象 Sites

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

// 添加键值对

Sites.put("one", "Google");

Sites.put("two", "Runoob");

Sites.put("three", "Taobao");

Sites.put("four", "Zhihu");

System.out.println(Sites);

}

}

执行以上代码，输出结果如下：

{four=Zhihu, one=Google, two=Runoob, three=Taobao}

访问元素

我们可以使用 get(key) 方法来获取 key 对应的 value:

实例

// 引入 HashMap 类

import java.util.HashMap;

public class RunoobTest {

public static void main(String[] args) {

// 创建 HashMap 对象 Sites

HashMap<Integer, String> Sites = new HashMap<Integer, String>();

// 添加键值对

Sites.put(1, "Google");

Sites.put(2, "Runoob");

Sites.put(3, "Taobao");

Sites.put(4, "Zhihu");

System.out.println(Sites.get(3));

}

}

执行以上代码，输出结果如下：

Taobao

删除元素

我们可以使用 remove(key) 方法来删除 key 对应的键值对(key-value):

实例

// 引入 HashMap 类

import java.util.HashMap;

public class RunoobTest {

public static void main(String[] args) {

// 创建 HashMap 对象 Sites

HashMap<Integer, String> Sites = new HashMap<Integer, String>();

// 添加键值对

Sites.put(1, "Google");

Sites.put(2, "Runoob");

Sites.put(3, "Taobao");

Sites.put(4, "Zhihu");

Sites.remove(4);

System.out.println(Sites);

}

}

执行以上代码，输出结果如下：

{1=Google, 2=Runoob, 3=Taobao}

删除所有键值对(key-value)可以使用 clear 方法：

实例

// 引入 HashMap 类

import java.util.HashMap;

public class RunoobTest {

public static void main(String[] args) {

// 创建 HashMap 对象 Sites

HashMap<Integer, String> Sites = new HashMap<Integer, String>();

// 添加键值对

Sites.put(1, "Google");

Sites.put(2, "Runoob");

Sites.put(3, "Taobao");

Sites.put(4, "Zhihu");

Sites.clear();

System.out.println(Sites);

}

}

执行以上代码，输出结果如下：

{}

计算大小

如果要计算 HashMap 中的元素数量可以使用 size() 方法：

实例

// 引入 HashMap 类

import java.util.HashMap;

public class RunoobTest {

public static void main(String[] args) {

// 创建 HashMap 对象 Sites

HashMap<Integer, String> Sites = new HashMap<Integer, String>();

// 添加键值对

Sites.put(1, "Google");

Sites.put(2, "Runoob");

Sites.put(3, "Taobao");

Sites.put(4, "Zhihu");

System.out.println(Sites.size());

}

}

执行以上代码，输出结果如下：

4

迭代 HashMap

可以使用 for-each 来迭代 HashMap 中的元素。

如果你只想获取 key，可以使用 keySet() 方法，然后可以通过 get(key) 获取对应的 value，如果你只想获取 value，可以使用 values() 方法。

实例

// 引入 HashMap 类

import java.util.HashMap;

public class RunoobTest {

public static void main(String[] args) {

// 创建 HashMap 对象 Sites

HashMap<Integer, String> Sites = new HashMap<Integer, String>();

// 添加键值对

Sites.put(1, "Google");

Sites.put(2, "Runoob");

Sites.put(3, "Taobao");

Sites.put(4, "Zhihu");

// 输出 key 和 value

for (Integer i : Sites.keySet()) {

System.out.println("key: " + i + " value: " + Sites.get(i));

}

// 返回所有 value 值

for(String value: Sites.values()) {

// 输出每一个value

System.out.print(value + ", ");

}

}

}

执行以上代码，输出结果如下：

key: 1 value: Google

key: 2 value: Runoob

key: 3 value: Taobao

key: 4 value: Zhihu

Google, Runoob, Taobao, Zhihu,