# 1.映射---Map<K,V>

## 1.1.定义

Map是Java中的映射接口。K-Key---健，V-Value---值---存储的时候是一个键对应一个值---键值对---要求键必须唯一，值随意。

一个Map中存储了很多的键值对。

Map<String,Integer> map = new HashMap<>(); //创建Map

Map.put(“abc”,3); //添加元素。

## 1.2.特点

1.Map中toString(); 方法也做了重写。

2.Map中不保证元素的存入顺序。

3.Map中如果出现了重复的键，则对应的值覆盖，键不变。

## 1.3.Map.Entry<K,V>

代表键值对的接口，每一个Entry对象代表一个键值对。一个Map对象实际上由多个Entry对象组成。是一个内部接口。

### 1.3.1.重要方法

1)equals(object o);

比较指定对象与此项的相等性。

2)getKey();

返回此项对应的键。

3)getValue();

返回此项对应的值。

4)setValue(V value);

用指定的值替换与此项对应的值。

## 1.4.重要方法

1>clear();

清空映射。

2>containsKey(K key);

判断是否包含指定的键。

3>containsValues(V value);

判断是否包含指定的值。

4>entrySet();

将所有的键值对放到一个Set集合中。

5>equals(Object o);

比较指定的对象与此映射是否相等。

6>get(K key);

获取键值对。如果键不存在，则返回null。

7>isEmpty();

判断是否为空。

8>keySet();

获取键。

9>put(K k,V v);

添加键和值。

10>remove(K key);

移除键值对。如果移除的键不存在，则直接忽略。

11>size();

获取键值对的个数。

12>values();

将所有的值放入一个Collection中。

Collection<Integer> c =map.values();

Map是集合吗？---不是

## 1.5.遍历

### 1.5.1.foreach

利用键来获取值。

Set<String> keys = map.keySet();

for(String key : keys){

System.out.println( key + ” = ” + map.get(key));

}

### 1.5.2.使用迭代器

获取所有的键值对组成的Set集合，Set中集合的元素类型是键值对类型，确定Set的泛型Map.Entry。

每一个Map.Entry对象都代表一个键值对。

Set<Map.Entry<String,Integer>> entrys = map.entrySet();

Iterator<Map.Entry<String,Integer>> it = entrys.iterator();

while (it.hasNext()) {

Map.Entry<String, Integer> entry = it.next();

System.out.println(entry.getKey() + "=" + entry.getValue());

}

## 1.6.子类

HashMap---允许键或者值为null。默认的初始容量是16，加载因子是0.75f，每次扩容一倍。是一个异步式线程不安全的映射

Hashtable---不允许键或者值为null。默认的初始容量是11，加载因子是0.75f。是一个同步式线程安全的映射

ConcurrentHashMap---异步式线程安全的映射

练习：输入一个数以及对应个数的键值对->键升序排序，如果出现了重复的键，对应的值求和

5

2 6

1 7

3 9

2 8

5 0

--->

1 7

2 14

3 9

5 0

# 2.Properties映射

## 2.1.定义

继承了Hashtable<Object o,Object o> class。

1.Properties是一个可以进行持久化的映射。键和值默认类型是String，而且只能是String类型。

2.当序列化Properties对象的时候，数据必须存储在properties文件中。

3.在properties文件中，如果输入中文，则自动的转会为对应的Unicode编码（范围：\u0000-\uffff 216）。Properties中不允许存储中文，因为properties的默认编码是西欧码表。

## 2.2.构造函数

1>Properties();

创建一个无默认值的空属性列表。

2>Properties(Properties defaults);

创建一个带有指定默认值的空属性列表。

## 2.3.重要方法

1>setProperty(String K,String V);

添加键值。

2>store(OutputStream out,String comments);

序列化，第二个参数表示向properties文件中添加注释，解释当前properties文件的作用。

3>load(InputStream inStream);

加载

4>getProperty(String k,String duf);

根据键获取值，如果键不存在返回null。我们可以指定一个默认值，如果没有找到，则返回默认值。

示例一：

public static void main(String[] args) throws Exception {

// 创建一个Properties对象

Properties prop = new Properties();

// 添加键值对

prop.setProperty("name", "Amy");

prop.setProperty("age", "18");

// 创建一个输出流

// 当序列化properties对象的时候，数据必须存储在properties文件中

FileOutputStream out = new FileOutputStream("p.properties");

// 序列化properties对象

// 第二个参数表示向properties文件中添加注释---解释当前properties文件的作用

prop.store(out, "描述了一个人~~~");

// 关流

out.close();

}

示例二：

public static void main(String[] args) throws Exception {

// 创建Properties对象存储反序列化之后的结果

Properties prop = new Properties();

// 读取properties文件

FileInputStream in = new FileInputStream("p.properties");

// 将properties对象反序列化回来

prop.load(in);

// 关流

in.close();

// 根据键获取对应的值

// 如果键不存在，则返回一个null

// System.out.println(prop.getProperty("gender"));

// 可以指定默认值；如果没有找到，则返回默认值

System.out.println(prop.getProperty("gender","男"));

}

# 3.File

代表文件或者目录(文件夹)的类

E:\\a.txt

File(String pathname)---代表了一个File对象；如果文件不存在，不会创建。在创建file对象的时候，并不会去计算机中检查文件是否存在，只是把传入的路径标志为一个file对象

## 2.1.重要方法

### 2.1.1.创建

1>createNewFile();

创建文件。只有当文件不存在的时候才会创建。此方法需要有一个boolean标记，创建成功返回一个true，创建失败返回一个file。在创建文件的时候，要求存放文件的路径要真实存在，如果不存在则报错，不会创建父目录。只能用于创建文件，不能用于创建目录。

boolean b = file.createNewFile();

2>mkdir();

创建目录。只能创建目录，不能创建文件。只能创建一层目录，不能一次性创建多层。

3>mkdirs();

创建多层目录。

### 2.1.2.删除

delete()---用于删除文件或者目录。---从计算机中真正移除，无法撤销。---如果删除的是目录且该目录下存在子目录或者子文件，此时无法删除

### 2.1.3.判断

1>canExecute();

判断是否是可执行文件。

2>canRead();

判断是否可读文件。

3>canWrite();

判断文件是否可写。

4>exists();

判断路径是否真实存在。

5>isAbsoute();

判断是否是绝对路径。

绝对路径：以盘符或者/开头的路径。和当前路径没有任何关系。

相对路径：不以盘符或者/开头的路径。以当前路径为基准进行计算。..表示的是上一层目录。

6>isDirectory();

判断是否是一个目录。

7>isFile();

判断是否是一个文件。

8>isHidden();

判断是否是隐藏文件。

### 2.1.4.获取

1>getAbsoluteFile();

获取File对象的绝对路径。

2>getAbsolutePath();

获取传入的File对象的路径。

3>listFiles();

获取所有的子目录和子文件组成的File数组。

File[] fl = file.listFiles();

4>listFiles(new Filefileter);

获取所有子目录。参数不可变。

5>listFiles(new FilenameFilter);

获取所有文件目录的名字。参数不可变。

6>lastModified();

获取文件最后一次修改时间。返回值表示是修改时间到1970-01-01 00:00:00的毫秒值。

7>getName();

获取文件的名字。

8>getParent();

获取父目录

### 2.1.5.设置

1>setLastModified(long l);

设置文件最后一次修改时间。参数也是long类型的毫秒值。

2>setExecutable(boolean b);

设置此文件是否可执行文件。

3>setReadable(boolean readable);

设置文件是否为可读文件。

4>setReadable(boolean readable,boolean ownerOnly);

设置用户对文件的可读权限。

5>setReadOnly();

设置文件是否为只读文件。

6>setWritable(boolean writable);

设置文件是否为可写文件。

7>setWritable(boolean writable,boolean ownerOnly);

设置用户对文件的可写权限。

8>renameTo(new File());

重命名。如果路径没有发生改变，则表示重命名此文件；如果路径发生来改变，则表示的是剪切此文件。

练习：删除目录(写成了一个方法：获取这个目录下的子文件和子目录---遍历数组---如果是子文件的话，直接删除；如果是子目录的话，后续的功能和当前方法的功能一致的，直接调用当前方法--递归)

public static void main(String[] args) {

File file = new File("E:\\aaa");

delFile(file);

}

private static void delFile(File file) {

// 判断file是否为null

if (file == null) {

throw new NullPointerException("亲，文件不能为空啊~~~");

}

// 判断是一个目录还是一个文件

if (file.isDirectory()) {

// 如果是目录，获取当前目录下的所有的子文件和子目录

File[] fs = file.listFiles();

// 遍历数组

for (File f : fs) {

// 获取到该file对象之后，需要再次判断这个f是一个子文件还是一个子目录

delFile(f);

}

}

// 无论是子目录还是子文件，都需要删除

file.delete();

}

练习：统计Java文件的个数和class文件的个数(写一个方法：获取工作空间下的所有的子文件和子目录---如果是子目录的话，递归；如果是子文件，需要判断是否是一个Java文件或者class文件)

public class FileExer2 {

// 记录Java文件的个数

static int javaCount = 0;

// 记录class文件的个数

static int classCount = 0;

public static void main(String[] args) {

File file = new File("E:\\workspace");

statistic(file);

System.out.println("Java文件：" + javaCount);

System.out.println("class文件：" + classCount);

}

private static void statistic(File file) {

// 判断file是否为空

if (file == null) {

throw new NullPointerException();

}

// 判断是否是一个目录

if (file.isDirectory()) {

// 获取这个目录下的所有的子文件和子目录

File[] fs = file.listFiles();

for (File f : fs) {

statistic(f);

}

} /\* 在计算机中一个file如果不是目录，那么就是文件 \*/ else if (file.getName().endsWith(".java")) {

// 如果是文件，判断是否是一个Java文件

javaCount++;

} else if (file.getName().endsWith(".class")) {

classCount++;

}

}

}

练习：提取所有的docx文件->E:\\DOCX

import java.io.File;

import java.io.FilenameFilter;

public class FileExer {

static File docx = new File("E:\\DOCX");

public static void main(String[] args) {

File file = new File("F:\\aaa");

get(file);

}

private static void get(File file) {

// 判断file是否为空

if (file == null) {

throw new NullPointerException();

}

// 判断file是否是一个目录

if (file.isDirectory()) {

// 获取这个目录下的所有的子目录和子文件

File[] fs = file.listFiles();

for (File f : fs) {

get(f);

}

} else if (file.getName().endsWith(".docx")) {

// 获取要提取的文件的名字

// 12.4test.docx

String name = file.getName();

// 后去 . 最后一次出现的位置

int index = name.lastIndexOf(".");

// 获取的是文件真正的名字

final String fileName = name.substring(0, index);

File[] fs = docx.listFiles(new FilenameFilter() {

@Override

public boolean accept(File dir, String name) {

return name.matches(fileName + "-\\d+\\.docx") || name.matches(fileName + "\\.docx");

}

});

// 提取文件

file.renameTo(new File(docx, fileName + (fs.length == 0 ? "" : "-" + fs.length) + ".docx"));

}

}

}