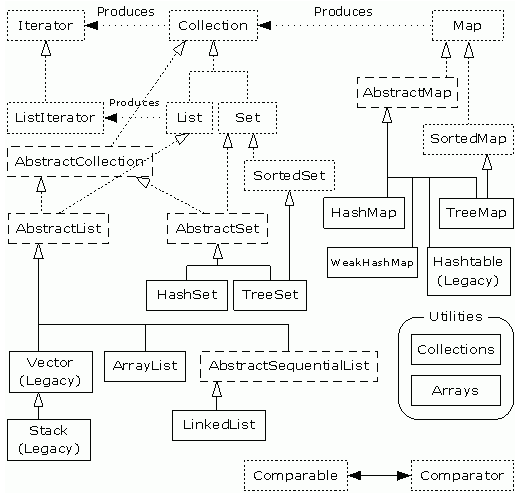
# **[Java 集合类详解](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/2526822.html)**

## **0.参考文献**

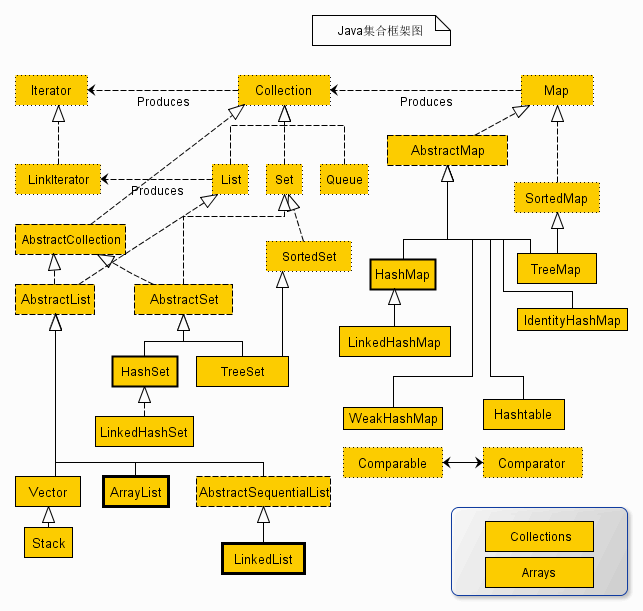
<http://blog.csdn.net/liulin_good/article/details/6213815>

## **1.java集合类图**

1.1



1.2



　　上述类图中，**实线边框**的是**实现类**，比如ArrayList，LinkedList，HashMap等，**折线边框**的是**抽象类**，比如AbstractCollection，AbstractList，AbstractMap等，而**点线边框**的是**接口**，比如Collection，Iterator，List等。

　　发现一个特点，上述所有的集合类，都实现了Iterator接口，这是一个用于遍历集合中元素的接口，主要包含hashNext(),next(),remove()三种方法。它的一个子接口LinkedIterator在它的基础上又添加了三种方法，分别是add(),previous(),hasPrevious()。也就是说如果是先Iterator接口，那么在遍历集合中元素的时候，只能往后遍历，被遍历后的元素不会在遍历到，通常无序集合实现的都是这个接口，比如HashSet，HashMap；而那些元素有序的集合，实现的一般都是LinkedIterator接口，实现这个接口的集合可以双向遍历，既可以通过next()访问下一个元素，又可以通过previous()访问前一个元素，比如ArrayList。

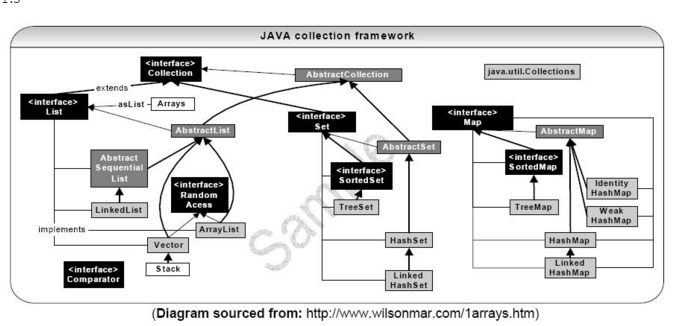
　　还有一个特点就是抽象类的使用。如果要自己实现一个集合类，去实现那些抽象的接口会非常麻烦，工作量很大。这个时候就可以使用抽象类，这些抽象类中给我们提供了许多现成的实现，我们只需要根据自己的需求重写一些方法或者添加一些方法就可以实现自己需要的集合类，工作流昂大大降低。

1.3

## **2.详解**

### **2.1HashSet**

HashSet是Set接口的一个子类，主要的特点是：里面不能存放重复元素，而且采用散列的存储方法，所以没有顺序。这里所说的没有顺序是指：元素插入的顺序与输出的顺序不一致。



### **代码实例：HashSetDemo**

IMG_259

[IMG_260](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

package edu.sjtu.erplab.collection;

import java.util.HashSet;import java.util.Iterator;import java.util.Set;

public class HashSetDemo {

public static void main(String[] args) {

Set<String> set=new HashSet<String>();

set.add("a");

set.add("b");

set.add("c");

set.add("c");

set.add("d");

//使用Iterator输出集合

Iterator<String> iter=set.iterator();

while(iter.hasNext())

{

System.out.print(iter.next()+" ");

}

System.out.println();

//使用For Each输出结合

for(String e:set)

{

System.out.print(e+" ");

}

System.out.println();

//使用toString输出集合 System.out.println(set);

}

}

[IMG_261](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

### **代码实例：SetTest**

IMG_262

[IMG_263](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

package edu.sjtu.erplab.collection;

import java.io.FileInputStream;import java.io.FileNotFoundException;import java.io.InputStream;import java.util.HashSet;import java.util.Iterator;import java.util.Scanner;import java.util.Set;

public class SetTest {

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

Set<String> words=new HashSet<String>();

//通过输入流代开文献

//方法1:这个方法不需要抛出异常

InputStream inStream=SetTest.class.getResourceAsStream("Alice.txt");

//方法2：这个方法需要抛出异常

//InputStream inStream = new FileInputStream("D:\\Documents\\workspace\\JAVAStudy\\src\\edu\\sjtu\\erplab\\collection\\Alice.txt");

Scanner in=new Scanner(inStream);

while(in.hasNext())

{

words.add(in.next());

}

Iterator<String> iter=words.iterator();

for(int i=0;i<5;i++)

{

if(iter.hasNext())

System.out.println(iter.next());

}

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

}

}

[IMG_264](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

### **2.2ArrayList**

ArrayList是List的子类，它和HashSet想法，允许存放重复元素，因此有序。集合中元素被访问的顺序取决于集合的类型。如果对ArrayList进行访问，迭代器将从索引0开始，每迭代一次，索引值加1。然而，如果访问HashSet中的元素，每个元素将会按照某种随机的次序出现。虽然可以确定在迭代过程中能够遍历到集合中的所有元素，但却无法预知元素被访问的次序。

### **代码实例：ArrayListDemo**

IMG_265

[IMG_266](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

package edu.sjtu.erplab.collection;

import java.util.ArrayList;import java.util.Iterator;import java.util.List;

public class ArrayListDemo {

public static void main(String[] args) {

List<String> arrList=new ArrayList<String>();

arrList.add("a");

arrList.add("b");

arrList.add("c");

arrList.add("c");

arrList.add("d");

//使用Iterator输出集合

Iterator<String> iter=arrList.iterator();

while(iter.hasNext())

{

System.out.print(iter.next()+" ");

}

System.out.println();

//使用For Each输出结合

for(String e:arrList)

{

System.out.print(e+" ");

}

System.out.println();

//使用toString输出集合 System.out.println(arrList);

}

}

[IMG_267](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

### **2.3LinkedList**

LinkedList是一种可以在任何位置进行高效地插入和删除操作的有序序列。

### **代码实例：LinkedListTest**

IMG_268

[IMG_269](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

package edu.sjtu.erplab.collection;

import java.util.ArrayList;import java.util.Iterator;import java.util.List;import java.util.ListIterator;

public class LinkedListTest {

public static void main(String[] args) {

List<String> a=new ArrayList<String>();

a.add("a");

a.add("b");

a.add("c");

System.out.println(a);

List<String> b=new ArrayList<String>();

b.add("d");

b.add("e");

b.add("f");

b.add("g");

System.out.println(b);

//ListIterator在Iterator基础上添加了add(),previous()和hasPrevious()方法

ListIterator<String> aIter=a.listIterator();

//普通的Iterator只有三个方法，hasNext()，next()和remove()

Iterator<String> bIter=b.iterator();

//b归并入a当中，间隔交叉得插入b中的元素

while(bIter.hasNext())

{

if(aIter.hasNext())

aIter.next();

aIter.add(bIter.next());

}

System.out.println(a);

//在b中每隔两个元素删除一个

bIter=b.iterator();

while(bIter.hasNext())

{

bIter.next();

if(bIter.hasNext())

{

bIter.next();//remove跟next是成对出现的，remove总是删除前序 bIter.remove();

}

}

System.out.println(b);

//删除a中所有的b中的元素 a.removeAll(b);

System.out.println(a);

}

}

[IMG_270](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

### **2.4HashMap**

参考之前的一篇博客：[Hashmap实现原理](http://www.cnblogs.com/xwdreamer/archive/2012/05/14/2499339.html)

### **2.5WeekHashMapDemo**

IMG_271

[IMG_272](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

package edu.sjtu.erplab.collection;

import java.util.WeakHashMap;

public class WeekHashMapDemo {

public static void main(String[] args) {

int size = 100;

if (args.length > 0) {

size = Integer.parseInt(args[0]);

}

Key[] keys = new Key[size];

WeakHashMap<Key, Value> whm = new WeakHashMap<Key, Value>();

for (int i = 0; i < size; i++) {

Key k = new Key(Integer.toString(i));

Value v = new Value(Integer.toString(i));

if (i % 3 == 0) {

keys[i] = k;//强引用 }

whm.put(k, v);//所有键值放入WeakHashMap中 }

System.out.println(whm);

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

System.gc();

try {

// 把处理器的时间让给垃圾回收器进行垃圾回收

Thread.sleep(4000);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(whm);

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

}

}

class Key {

String id;

public Key(String id) {

this.id = id;

}

public String toString() {

return id;

}

public int hashCode() {

return id.hashCode();

}

public boolean equals(Object r) {

return (r instanceof Key) && id.equals(((Key) r).id);

}

public void finalize() {

System.out.println("Finalizing Key " + id);

}

}

class Value {

String id;

public Value(String id) {

this.id = id;

}

public String toString() {

return id;

}

public void finalize() {

System.out.println("Finalizing Value " + id);

}

}

[IMG_273](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

输出结果

IMG_274

[IMG_275](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

{50=50, 54=54, 53=53, 52=52, 51=51, 46=46, 47=47, 44=44, 45=45, 48=48, 49=49, 61=61, 60=60, 63=63, 62=62, 65=65, 64=64, 55=55, 56=56, 57=57, 58=58, 59=59, 76=76, 75=75, 74=74, 73=73, 72=72, 71=71, 70=70, 68=68, 69=69, 66=66, 67=67, 85=85, 84=84, 87=87, 86=86, 81=81, 80=80, 83=83, 82=82, 77=77, 78=78, 79=79, 89=89, 88=88, 10=10, 90=90, 91=91, 92=92, 93=93, 94=94, 95=95, 96=96, 97=97, 98=98, 99=99, 20=20, 21=21, 12=12, 11=11, 14=14, 13=13, 16=16, 15=15, 18=18, 17=17, 19=19, 8=8, 9=9, 31=31, 4=4, 32=32, 5=5, 6=6, 30=30, 7=7, 0=0, 1=1, 2=2, 3=3, 29=29, 28=28, 27=27, 26=26, 25=25, 24=24, 23=23, 22=22, 40=40, 41=41, 42=42, 43=43, 38=38, 37=37, 39=39, 34=34, 33=33, 36=36, 35=35}

100

Finalizing Key 98

Finalizing Key 97

Finalizing Key 95

Finalizing Key 94

Finalizing Key 92

Finalizing Key 91

Finalizing Key 89

Finalizing Key 88

Finalizing Key 86

Finalizing Key 85

Finalizing Key 83

Finalizing Key 82

Finalizing Key 80

Finalizing Key 79

Finalizing Key 77

Finalizing Key 76

Finalizing Key 74

Finalizing Key 73

Finalizing Key 71

Finalizing Key 70

Finalizing Key 68

Finalizing Key 67

Finalizing Key 65

Finalizing Key 64

Finalizing Key 62

Finalizing Key 61

Finalizing Key 59

Finalizing Key 58

Finalizing Key 56

Finalizing Key 55

Finalizing Key 53

Finalizing Key 52

Finalizing Key 50

Finalizing Key 49

Finalizing Key 47

Finalizing Key 46

Finalizing Key 44

Finalizing Key 43

Finalizing Key 41

Finalizing Key 40

Finalizing Key 38

Finalizing Key 37

Finalizing Key 35

Finalizing Key 34

Finalizing Key 32

Finalizing Key 31

Finalizing Key 29

Finalizing Key 28

Finalizing Key 26

Finalizing Key 25

Finalizing Key 23

Finalizing Key 22

Finalizing Key 20

Finalizing Key 19

Finalizing Key 17

Finalizing Key 16

Finalizing Key 14

Finalizing Key 13

Finalizing Key 11

Finalizing Key 10

Finalizing Key 8

Finalizing Key 7

Finalizing Key 5

Finalizing Key 4

Finalizing Key 2

Finalizing Key 1

{54=54, 51=51, 45=45, 48=48, 60=60, 63=63, 57=57, 75=75, 72=72, 69=69, 66=66, 84=84, 87=87, 81=81, 78=78, 90=90, 93=93, 96=96, 99=99, 21=21, 12=12, 15=15, 18=18, 9=9, 6=6, 30=30, 0=0, 3=3, 27=27, 24=24, 42=42, 39=39, 33=33, 36=36}

34

[IMG_276](http://www.cnblogs.com/xwdreamer/archive/2012/05/30/javascript:void(0);)

 疑问：为什么value没有被回收。

## **3.比较**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 是否有序 | 是否允许元素重复 |
| Collection | | 否 | 是 |
| List | | 是 | 是 |
| Set | AbstractSet | 否 | 否 |
|  | HashSet |
|  | TreeSet | 是（用二叉排序树） |
| Map | AbstractMap | 否 | 使用key-value来映射和存储数据，key必须唯一，value可以重复 |
|  | HashMap |
|  | TreeMap | 是（用二叉排序树） |