对象数组

2018年7月1日 14:58

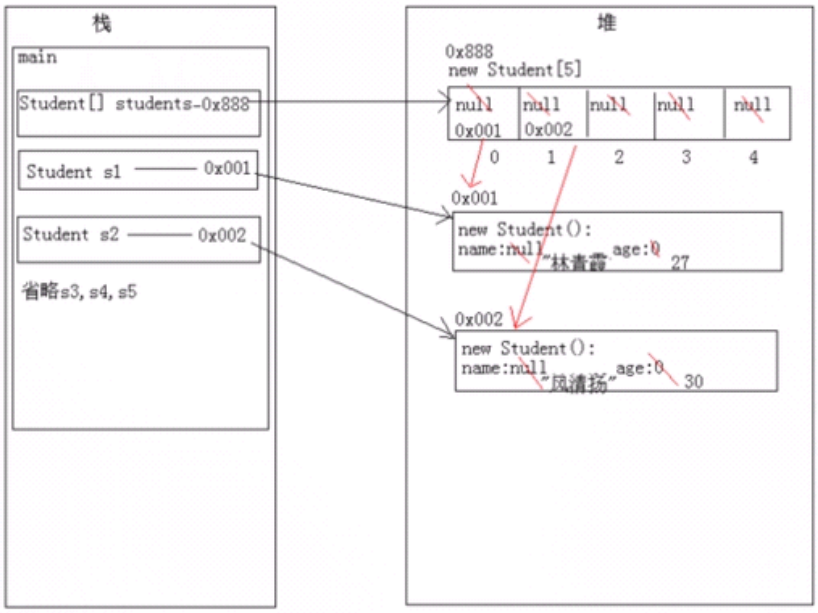
```
package cn.itcast_01;

public class Student {
	private String Name;
	private int age;

	public Student() {
		super();
		// TODO Auto-generated constructor stub
	}

	public Student(String name, int age) {
		super();
		Name = name;
		this.age = age;
	}

	public String getName() {
		return Name;
	}

	public void setName(String name) {
		Name = name;
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

	@Override
	public String toString() {
		return "Student [Name=" + Name + ", age=" + age + "]";
	}

}
```

```
public class ObjectArrayDemo {
	public static void main(String[] args) {
		//创建学生数组
		Student[] students=new Student[5];

		Student s1 = new Student("王若潇",22);
		Student s2 = new Student("周杰伦",40);
		Student s3 = new Student("赵磊",12);
		Student s4 = new Student("孙权",62);
		Student s5 = new Student("C罗",33);

		students[0]=s1;
		students[1]=s2;
		students[2]=s3;
		students[3]=s4;
		students[4]=s5;

		for(int x=0;x<students.length;x++) {
			System.out.println(students[x].getName()+"---"+studen
			ts[x].getAge());
		}

	}
}
```

内存图

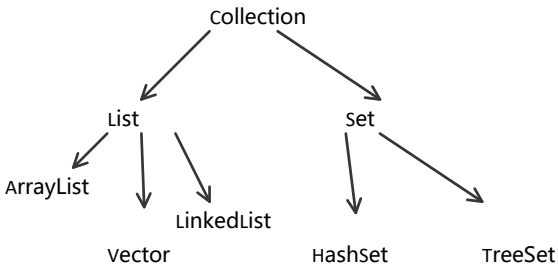
2018年7月13日 16:40

```
// 创建学生数组(对象数组)。
Student[] students = new Student[5];

// 创建5个学生对象，并赋值。
Student s1 = new Student("林青霞", 27);
Student s2 = new Student("风清扬", 30);
Student s3 = new Student("刘意", 30);
Student s4 = new Student("赵雅芝", 60);
Student s5 = new Student("王力宏", 35);

// 把C步骤的元素，放到数组中。
students[0] = s1;
students[1] = s2;
students[2] = s3;
students[3] = s4;
students[4] = s5;


// 遍历
for (int x = 0; x < students.length; x++) {
        //System.out.println(students[x]);

        Student s = students[x];
        System.out.println(s.getName()+"---"+s.getAge());
}
```



API-集合

2018年7月13日 16:18



集合类：方便对一组对象的操作

数组与集合的区别：
1.长度的区别：数组的长度固定，集合的长度可变
2.内容不同
　　数组存储的是同一种类型的元素
　　集合可以存储不同类型的元素
3.元素的数据类型问题：
　　数组可以存储基本类型和引用类型
　　集合只能存储引用类型

针对不同的需求，Java提供了不同的集合类，多个集合类和数据结构不同
数据结构：数据的存储方式
多个集合类是有共性的，把集合的共性内容不断向上提取，形成集合的继承体系

Collection类

2018年7月13日 17:01

Collection是接口，对象为元素

```
import java.util.ArrayList;
import java.util.Collection;

/*
 * Collection 是集合的顶级接口，它的子体系有重复有唯一、有有序有无序
 *                功能：
 *                1.添加 boolean add(Object o):添加一个元素
 *                                boolean addAll(Collection c):添加一个集合
 *                2.删除 void clear():移除所有元素
 *                                boolean remove(Object o):移除一个元素
 *                                boolean remove(Collection c)：移除一个集合的元素
 *                3.判断 boolean contains(Object o):判断集合中是否包含指定的集合元素
 *                                boolean contains(Collection c)
 *                                boolean isEmpty():判断集合是否为空
 *                4.获取 Iterator<E> iterator():重点
 *                5.长度 int size() :元素的个数
 *                6.交集 boolean retainAll(Collection c):两个集合都有的元素
 *                7.把集合转为数组       Object[] toArray()
 */
public class CollectionDemo {
        public static void main(String[] args) {
                //创建集合对象
                Collection c =new ArrayList();

                //boolean add(Object o)
                //System.out.println("add:"+c.add("hello"));//永远都为true
                c.add("hello");
                c.add("world");
                c.add("java");
                System.out.println("c="+c);

                //void clear()
                //c.clear();
                //System.out.println("c="+c);
```

```
            //boolean remove(Object o)
            System.out.println(c.remove("hello"));
            System.out.println(c.remove("javaee"));
            System.out.println("c="+c);
            c.add("hello");

            //boolean contains(Object o)
            System.out.println("contains="+c.contains("hello"));
            System.out.println("contains="+c.contains("android"));

            //boolean isEmpty()
            System.out.println("isempty="+c.isEmpty());

            //int size()
            System.out.println("size="+c.size());
        }
}
```

高级方法

2018年7月13日 21:54

```
import java.util.ArrayList;
import java.util.Collection;

/*
 *boolean addAll(Collection c):添加一个集合
 *boolean removeAll(Collection c)：移除一个集合的元素
 *boolean containsAll(Collection c)
 *boolean retainAll(Collection c):两个集合都有的元素
 */
public class CollectionDemo2 {
	public static void main(String[] args) {
		// 创建集合1
		Collection c1 = new ArrayList();
		c1.add("abc1");
		c1.add("abc2");
		c1.add("abc3");
		c1.add("abc4");

		// 创建集合2
		Collection c2 = new ArrayList();
		c2.add("abc4");
		c2.add("abc5");
		c2.add("abc6");
		c2.add("abc7");

		//boolean addAll(Collection c):可以有重复
		//c1.addAll(c2);

		//boolean removeAll(Collection c):移除共有元素
		System.out.println(c1.removeAll(c2)); //c1=[abc1, abc2, abc3]
		c1.add("abc4");

		//boolean containsAll(Collection c):只有包含所有的元素才返回true
		System.out.println("containsAll="+c1.contains(c2));//false

		//boolean retainAll(Collection c)：把共有的付给c1
		System.out.println("retainAll="+c1.retainAll(c2));//c1=[abc4]

		System.out.println("c1="+c1);
		System.out.println("c2="+c2);
	}
```

假设有两个集合A，B
A对B作交集，最终保存进A中，B不变
返回值表示的是A是否发生改变

```
}
```

集合的遍历

2018年7月13日 22:36

```
import java.util.ArrayList;
import java.util.Collection;

/*
 *      遍历数组
 *              Object[] toArray()
 */
public class CollectionDemo3 {
    public static void main(String[] args) {
        // 创建集合对象
        Collection c = new ArrayList();
        c.add("hello");
        c.add("world");
        c.add("java");

        // 遍历
        // Object[] toArray() 把集合变为数组，可以实现集合遍历
        Object[] objs = c.toArray();
        for (int x = 0; x < objs.length; x++) {
            System.out.println(objs[x]);
            // System.out.println(objs[x].length());
            // object中没有length()方法，要把元素还原为字符串
            // 向下转型
            String s = (String) objs[x];
            System.out.println(s.length());

        }
    }
}
```

```
import java.util.ArrayList;
import java.util.Collection;

import cn.itcast_01.Student;

public class StudentDemo {
    public static void main(String[] args) {
        Collection c = new ArrayList();
        Student s1 = new Student("王若潇",22);
        Student s2 = new Student("周杰伦",40);
        Student s3 = new Student("赵磊",12);
        Student s4 = new Student("孙权",62);
        Student s5 = new Student("C罗",33);

        c.add(s1);
        c.add(s2);
        c.add(s3);
        c.add(s4);
        c.add(s5);

        Object[] objs = c.toArray();
        for(int x=0;x<objs.length;x++) {
            Student s = (Student)objs[x];
            System.out.println(s.getName()+"---"+s.getAge());
        }
    }
}
```

Iterator

2018年7月13日 23:30

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;

/*
 * Iterator<E> iterator():迭代器，集合的专用遍历方法
 *                获取元素并移动到下一个位置：Object next()
 *                        NoSuchElementException 没有元素
 *                判断是否还有下一个元素
 *                        boolean hasNext()
 */
public class IteratorDemo {
        public static void main(String[] args) {
                // 创建集合对象
                Collection c = new ArrayList();
                c.add("hello");
                c.add("world");
                c.add("java");

                // Iterator iterator()
                Iterator it = c.iterator();// 实际返回值肯定为子类对象，多态
//                  System.out.println(it.next());
//                  System.out.println(it.next());
//                  System.out.println(it.next());
//                  System.out.println(it.next());

                // 标准代码
                while (it.hasNext()) {
                        System.out.println(it.next());
                }
        }
}
```

练习

2018年7月13日 23:42

```
package cn.itcast_04;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;

import cn.itcast_01.Student;

/*
 *        用迭代器遍历集合存储的学生对象
 *       不要多次使用it.next()方法
 */
public class InteratorTest {
	public static void main(String[] args) {
		Collection c = new ArrayList();
		Student s1 = new Student("王若潇", 22);
		Student s2 = new Student("周杰伦", 40);
		Student s3 = new Student("赵磊", 12);
		Student s4 = new Student("孙权", 62);
		Student s5 = new Student("C罗", 33);

		c.add(s1);
		c.add(s2);
		c.add(s3);
		c.add(s4);
		c.add(s5);

		// Iterator it = c.iterator();
		// while(it.hasNext()) {
		// Student s = (Student) it.next();
		// System.out.println(s.getName()+"---"+s.getAge());
		// }

		// for循环改进
		for (Iterator it = c.iterator(); it.hasNext();) {
			Student s = (Student) it.next();
```

```
                System.out.println(s.getName() + "---" + s.getAge());
            }
        }
    }
```

集合的使用步骤

2018年7月13日 23:59

1.创建集合对象
2.创建元素对象
3.把元素添加到集合
4.遍历集合 Iterator
    a.通过集合对象获取迭代器对象
    b.通过迭代器对象的hasNext（）方法判断是否有元素
    c.通过迭代器对象的Next（）方法获取元素并移动到下一个位置

迭代器原理

2018年7月14日 9:57

迭代器为什么不定义为类，而是一个接口？

java中提供了很多的集合类，而这些集合类的数据结构是不同的，数据的存储和遍历方式是不同的，
因此没有定义迭代器类
而无论是哪种集合，都应该具备获取元素的操作，并辅助于判断功能，在获取前先判断。
判断功能和获取功能应该是一个集合遍历必须的，把这两个功能提取出来，而不是具体实现，这种方式就是接口
在真正的具体子类中，以内部类的方式体现

List类

2018年7月14日 10:58

```
package cn.itcast_01;

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

/*
 * List集合存储字符串并遍历
 */
public class ListDemo {
	public static void main(String[] args) {
		List list = new ArrayList();

		list.add("hello");
		list.add("world");
		list.add("java");

		Iterator it = list.iterator();
		while (it.hasNext()) {
			String s = (String) it.next();
			System.out.println(s);
		}
	}
}
```

List特点

2018年7月14日 11:03

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

/*
 * List集合的特点:
 *                  有序、可重复
 */
public class ListDemo2 {
	public static void main(String[] args) {
		List list = new ArrayList();

		list.add("hello");
		list.add("world");
		list.add("java");
		list.add("javaee");
		list.add("javaee");// 可重复
		list.add("javaee");

		Iterator it = list.iterator();
		while (it.hasNext()) {
			String s = (String) it.next();
			System.out.println(s);
		}
	}
}
```

List集合的特有功能

2018年7月14日 11:20

```
import java.util.ArrayList;
import java.util.List;

/*
 * List集合的特有功能
 * A.添加功能：
 *          void add(int index,Object element):在指定位置添加元素
 * B.获取功能
 *          Object obj get(int index):获取指定位置元素
 * C.列表迭代器
 *          ListIterator listIterator():列表集合特有的迭代器
 * D.删除功能
 *          Object remove(int index):根据索引删除元素
 * E.修改功能
 *          Object set(int index,Object element):根据索引修改元素
 */
public class ListDemo {
    public static void main(String[] args) {
        List list = new ArrayList();

        list.add("hello");
        list.add("world");
        list.add("java");

        // void add(int index,Object element)
        list.add(1, "Android");// list=[hello, Android, world, java]

        // Object obj get(int index)
        System.out.println("get=" + list.get(1));

        // Object remove(int index)
        System.out.println("remove=" + list.remove(1));// remove=Android

        // Object set(int index,Object element)
        System.out.println("set=" + list.set(1, "javaee"));// set=world
```

```
            System.out.println("list=" + list);
        }
    }
```

List遍历

2018年7月14日 12:04

```
//List遍历
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class ListDemo2 {
	public static void main(String[] args) {
		List list = new ArrayList();

		list.add("hello");
		list.add("world");
		list.add("java");

		for (int x = 0; x < list.size(); x++) {
			String s = (String) list.get(x);
			System.out.println(s);
		}

		// 迭代器遍历
		Iterator it = list.iterator();
		while (it.hasNext()) {
			String s = (String) it.next();
			System.out.println(s);
		}
	}
}
```

ListIterator

2018年7月14日 14:27

List集合特有的列表迭代器

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

/*列表迭代器
	ListIterator listIterator()
	该迭代器继承自Iterator迭代器，由hasnext和next方法

	特有功能：Object previous():逆向遍历
	*/
public class ListIteratorDemo {
	public static void main(String[] args) {
		List list = new ArrayList();

		list.add("hello");
		list.add("world");
		list.add("java");

		ListIterator lit = list.listIterator();// 子类对象
		while (lit.hasNext()) {
			String s = (String) lit.next();
			System.out.println(s);
		}
		while (lit.hasPrevious()) {
			System.out.println(lit.previous());// 逆向打印
		}
	}
}
```

并发异常

2018年7月14日 15:15

```
/*
 * ConcurrentModificationException
 *		当方法检测到对象的并发修改，但不允许这种修改时，抛出次异常
 *		产生原因：迭代器是依赖于集合存在的，集合发生改变后，迭代器不知道，成为并发修改异常
 *		如何解决：
 *			1.迭代器迭代元素，迭代器修改元素
 *			2.集合遍历元素，集合修改元素
 */
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;

public class ListIteratorDemo2 {
	public static void main(String[] args) {
		List list = new ArrayList();

		list.add("hello");
		list.add("world");
		list.add("java");

		/*
		 * Iterator it = list.Iterator(); while (lit.hasNext()) { String s = (String)
		 * it.next(); if ("world".equals(s)) { list.add("javaee"); } }
		 */

		// 迭代器迭代元素，迭代器修改元素
		// 添加到删除元素之后
		// 使用子接口ListIterator
		ListIterator lit = list.listIterator();
		while (lit.hasNext()) {
			String s = (String) lit.next();
			if ("world".equals(s)) {
				lit.add("javaee");
			}
		}
```

```
            System.out.println("list=" + list);// list=[hello, world, javaee, java]

            // 集合遍历元素，集合修改元素
            // 在最后添加
            for (int x = 0; x < list.size(); x++) {
                String s = (String) list.get(x);
                if ("world".equals(s)) {
                    list.add("javaee");
                }
            }
            System.out.println("list=" + list);//list=[hello, world, javaee, java, javaee]
        }
}
```

数据结构

2018年7月14日 15:16

数据结构：数据的组织方式

栈：先进后出
压栈、弹栈

队列：先进先出
入口、出口

数组：有索引、方便获取
查询快、增删慢

链表：由一个链子把多个结点连起来组成的数据
由数据域和指针域组成
查询慢、增删快

List的子类特点

2018年7月14日 16:27

List：

ArrayList:底层数据结构是数组，查询快，增删慢；线程不安全，效率高

Vector：底层数据结构是数组，查询快，增删慢；线程安全，效率低 ——→ 基本不用

LinkedList：底层数据结构为链表，查询慢，增删快；线程不安全，效率高

看需求使用

查询多：ArrayList

增删多：LinkedList