ArrayList

2018年7月1日 14:58

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

/*
 *        使用List的各个子类遍历集合
 * ArrayList
 */
public class ArrayListDemo {
        public static void main(String[] args) {
                //创建集合对象
                ArrayList array = new ArrayList();

                //添加元素
                array.add("hello");
                array.add("world");
                array.add("java");

                //遍历
                Iterator it = array.iterator();
                while(it.hasNext()) {
                        String s =(String)it.next();
                        System.out.println(s);
                }
                System.out.println("-----------");

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

        }
}
```

Vector类

2018年7月14日 20:54

```
import java.util.Enumeration;
import java.util.Vector;

/*
 * Vector 的特有功能:
 *        1.添加功能
 *                public void addElement(Object obj) ---add
 *        2.获取功能
 *                public Object elementAt(int index) ---get
 *                public Enumeration elements()//等同于迭代器 -----Iterator
 */
public class VectorDemo {
        public static void main(String[] args) {
                Vector v = new Vector();

                v.addElement("Hello");
                v.addElement("world");
                v.addElement("java");

                for (int x = 0; x < v.size(); x++) {
                        String s = (String) v.elementAt(x);
                        System.out.println(s);
                }
                System.out.println("------------");

                Enumeration en = v.elements();// 返回实现类的对象
                while (en.hasMoreElements()) {
                        String s = (String) en.nextElement();
                        System.out.println(s);
                }

        }
}
```

LinkedList

2018年7月14日 21:12

```
import java.util.LinkedList;

/*
 * LinkedList的特有功能
 *		A.添加功能
 *			public void addFirst(Object e)
 *			public void addLast(Object e)
 *		B.获取功能
 *			public Object getFirst()
 *			public Object getLast()
 *		D.删除功能
 *			public Object removeFirst()
 *			public Object removeLast()
 */
public class LinkedListDemo {
	public static void main(String[] args) {
		LinkedList link = new LinkedList();

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

		link.addFirst("javaee");
		link.addLast("android");// 无意义----与add一样
		System.out.println(link.getFirst());
		System.out.println(link.getLast());

		System.out.println(link.removeFirst());
		System.out.println(link.removeLast());// 删除并返回

		System.out.println("link:" + link);
	}
}
```

集合去重1

2018年7月14日 21:52

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

/*
 * ArrayList中去除集合中字符串的重复值
 */
public class ArrayListDemo {
	public static void main(String[] args) {
		ArrayList array = new ArrayList();
		array.add("hello");
		array.add("world");
		array.add("java");
		array.add("hello");
		array.add("world");
		array.add("java");

		//创建新集合
		ArrayList newArray =new ArrayList();

		//遍历旧集合，获取每一个元素
		Iterator it = array.iterator();
		while(it.hasNext()) {
			String s = (String)it.next();

			if(!newArray.contains(s)) {
				newArray.add(s);
			}
		}

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

集合去重2

2018年7月15日 8:59

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

/*
 * ArrayList中去除集合中字符串的重复值
 *       不创建新数组
 */
public class ArrayListDemo2 {
	public static void main(String[] args) {
		ArrayList array = new ArrayList();
		array.add("hello");
		array.add("world");
		array.add("java");
		array.add("hello");
		array.add("world");
		array.add("java");

		// 拿0索引依次与后面比较
		for (int x = 0; x < array.size() - 1; x++)
			for (int y = x + 1; y < array.size(); y++) {
				if (array.get(x).equals(array.get(y))) {
					array.remove(y);
					y--;
				}

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

link模拟栈

2018年7月15日 9:22

```
import java.util.LinkedList;

/*
 *      自定义的栈集合
 */
public class MyStack {
	private LinkedList link;

	public MyStack() {
		link = new LinkedList();
	}

	public void add(Object obj) {
		link.addFirst(obj);
	}

	public Object get() {
		return link.removeFirst();
	}

	public boolean isEmpty() {
		return link.isEmpty();
	}
}
```

```
import java.util.Iterator;
import java.util.LinkedList;

/*
 *      用LinkedList模拟栈数据结构的集合
 */
public class MyStackDemo {
	public static void main(String[] args) {
		MyStack ms = new MyStack();

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

		while (!ms.isEmpty()) {
			System.out.println(ms.get());
		}
	}
}
```

泛型

2018年7月15日 10:48 看API，如果类、接口后有<E>,可加入泛型

```
package cn.itcast_06;

/*
 *      为避免出错，集合在创建对象的时候就明确元素的数据类型，这种技术为泛型
 *      泛型：是一种把类型明确的工作推迟到建对象或调方法的时候才去明确的特殊类型
 *              也被称为参数化类型，把类型像参数一样传递
 *      格式：
 *              <数据类型> 只能是引用类型
 *      好处：  1.把运行时的问题提前到编译期间
 *                      2.避免了强制类型转换
 *                      3.优化了程序设计，解决了黄色警告线
 */
import java.util.ArrayList;
import java.util.Iterator;

public class GenericDemo {
        public static void main(String[] args) {
                // 创建集合对象
                ArrayList<String> array = new ArrayList<String>();

                // 添加元素
                array.add("hello");
                array.add("world");
                array.add("java");
                // array.add(10);//JDK5以后自动装箱
                // 等价于：array.add(Integer.valueOf(10))

                // 遍历
                Iterator<String> it = array.iterator();
                while (it.hasNext()) {
                        // String s = (String) it.next();
                        String s = it.next();
                        System.out.println(s);
                }
        }
}
```

泛型

2018年7月15日 11:05

```
package cn.itcast_06;

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

/*
 *      存储自定义对象并遍历
 */
public class ArrayListDemo {
	public static void main(String[] args) {
		//JDK7的新特性：泛型推断（不建议）
		ArrayList<Student> array = new ArrayList<Student>();

		array.add(new Student("王若潇",22));
		array.add(new Student("曹操",260));
		array.add(new Student("吕布",32));

		Iterator<Student> it=array.iterator();
		while(it.hasNext()) {
			Student s=it.next();
			System.out.println(s.getName()+"----"+s.getAge());
		}
	}
}
```

泛型产生原因

2018年7月15日 11:13

```
package cn.itcast_07;
/*
 *      早期的时候。使用Object类代表任意类型
 *      向上转型没问题，向下转型时存在类型转换的问题，不安全
 *      Java在JDK5后引入泛型，提高程序安全性
 */
public class ObjectToolDemo {
	public static void main(String[] args) {
		//正常使用
		ObjectTool ot = new ObjectTool();
		ot.setObj(new Integer(27));
		Integer i =(Integer)ot.getObj();
		System.out.println("i="+i);

		ot.setObj(new String("王若潇"));
		String s=(String)ot.getObj();
		System.out.println("s="+s);
		System.out.println();

		/*ot.setObj(new Integer(27));
		String ss=(String)ot.getObj();
		System.out.println("s="+s);*/

	}
}
```

泛型类

2018年7月15日 11:39

```
package cn.itcast_08;
//泛型类
public class ObjectTool<T> {
	private T obj;

	public T getObj() {
		return obj;
	}

	public void setObj(T obj) {
		this.obj = obj;
	}

}
```

```
package cn.itcast_08;

/*
 * 泛型类的测试
 */
public class ObjectToolDemo {
	public static void main(String[] args) {
		ObjectTool<String> ot = new ObjectTool<String>();

		ot.setObj(new String("王若潇"));
		// String s = (String)ot.getObj();
		String s = ot.getObj();// 不再需要强转
		System.out.println("姓名是:" + s);

		ObjectTool<Integer> ot2 = new ObjectTool<Integer>();
		ot2.setObj(new Integer(27));
		Integer i = (Integer) ot2.getObj();
		System.out.println("i=" + i);
	}
}
```

泛型方法

2018年7月15日 11:39

把泛型定义在方法上

```
/*
 *      泛型方法
 */
public class ObjectTool {
	public <T> void show(T t) {
		System.out.println(t);
	}
}
```

```
public class ObjectToolDemo {
	public static void main(String[] args) {
		ObjectTool ot = new ObjectTool();
		ot.show("hello");
		ot.show(100);
		ot.show(true);
	}
}
```

泛型接口

2018年7月15日 17:52

```
package cn.itcast_10;
/*
 *      泛型接口：把泛型定义在接口上
 */
public interface Inter<T> {
	public abstract void show(T t);
}
```

```
/*
 *      实现类在实现接口的时候：
 *              1.已知类型
 *              2.未知类型
 */
/*public class InterImpl implements Inter<String>{

	@Override
	public void show(String t) {
		System.out.println(t);
	}
}*/

//第二种情况
public class InterImpl<T> implements Inter<T>{

	@Override
	public void show(T t) {
		System.out.println(t);
	}

}
```

```
public class InterDemo {
	public static void main(String[]
	args) {
		/*Inter<String> i = new
		InterImpl();
		i.show("hello");*/

		//第二种情况
		Inter<String> i =new
		InterImpl<String>();
		i.show("hello");

		Inter<Integer> ii = new
		InterImpl<Integer>();
		ii.show(100);
	}
}
```

泛型高级之通配符

2018年7月15日 18:10

```
/*
 *      通配符：
 *      ?：任意类型：如果没有明确，那么就是Object及任意java类
 *      ? extends E:向下限定，E及其子类
 * ? super E:向上限定，E及其父类
 */
import java.util.ArrayList;
import java.util.Collection;

public class GenericDemo {
	public static void main(String[] args) {
		// 泛型如果明确写的时候，前后必须一致
		Collection<Object> c = new ArrayList<Object>();
		// Collection<Object> c2=new ArrayList<Animal>();

		// ?表示任意类型都是可以的
		Collection<?> c3 = new ArrayList<Object>();
		Collection<?> c4 = new ArrayList<Animal>();
		Collection<?> c5 = new ArrayList<Cat>();
		Collection<?> c6 = new ArrayList<Dog>();

		// ? extends E
		// Collection<? extends Animal> c7=new ArrayList<Object>();
		Collection<? extends Animal> c8 = new ArrayList<Animal>();
		Collection<? extends Animal> c9 = new ArrayList<Cat>();
		Collection<? extends Animal> c10 = new ArrayList<Dog>();

		// ? extends E
		Collection<? super Animal> c11=new ArrayList<Object>();
		Collection<? super Animal> c12 = new ArrayList<Animal>();
		//Collection<? super Animal> c13 = new ArrayList<Cat>();
		//Collection<? super Animal> c14 = new ArrayList<Dog>();
	}
}

class Animal {
```

```
}

class Dog extends Animal {
}

class Cat extends Animal {
}
```

增强for

2018年7月15日 19:18

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

/*
 * JDK5的新特性：自动拆装箱、泛型、增强for、静态导入、可变参数、枚举
 *      增强for是for循环的一种
 *              for(元素数据类型 变量:数组或者Collection集合){}
 *
 *      好处：简化了数组和集合的遍历
 *      弊端：增强for的目标不能为null
 *                      先进行不为null的判断
 *      增强for的实质是代替迭代器的
 */
public class ForDemo {
    public static void main(String[] args) {
        int[] arr = { 1, 2, 3, 4, 5 };
        for (int x = 0; x < arr.length; x++) {
            System.out.println(arr[x]);
        }
        System.out.println("-----------");

        // 增强for
        for (int x : arr) {
            System.out.println(x);
        }
        System.out.println("-----------");

        ArrayList<String> array = new ArrayList<>();
        array.add("hello");
        array.add("world");
        array.add("java");

        for (String s : array) {
            System.out.println(s);
        }
```

```
            List<String> list = null;
            if (list != null) {
                for (String s : list) {
                    System.out.println(s);
                } // NullPointerException
            }
        }
}
```

练习

2018年7月15日 20:05

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

//ArrayList储存数组并遍历，加入泛型，增强for
public class ArrayListDemo {
	public static void main(String[] args) {
		ArrayList<String> array = new ArrayList<String>();
		array.add("hello");
		array.add("world");
		array.add("java");

		// 迭代器
		Iterator<String> it = array.iterator();
		while (it.hasNext()) {
			String s = it.next();
			System.out.println(s);
		}
		System.out.println("---------");

		// 普通for
		for (int x = 0; x < array.size(); x++) {
			String s = array.get(x);
			System.out.println(s);
		}
		System.out.println("---------");

		// 增强for(常用)
		for (String s : array) {
			System.out.println(s);
		}
	}
}
```

静态导入

2018年7月15日 20:42

```
/*
 *      静态导入：import static 包名...类名.方法名
 *              可直接导入方法的级别
 *      注意事项：
 *                      1.方法是静态的
 *                      2.多个同名方法，必须加前缀
 */
import static java.lang.Math.abs;
import static java.lang.Math.pow;
import static java.lang.Math.max;

public class StaticImportDemo {
    public static void main(String[] args) {
        System.out.println(java.lang.Math.abs(-100));
        System.out.println(java.lang.Math.pow(2, 3));
        System.out.println(java.lang.Math.max(10, 20));

        // 太复杂，引入import
        System.out.println(Math.abs(-100));
        System.out.println(Math.pow(2, 3));
        System.out.println(Math.max(10, 20));

        //
        System.out.println(abs(-100));
        System.out.println(pow(2, 3));
        System.out.println(max(10, 20));

    }
}
```

可变参数

2018年7月15日 20:05

```
/*
 *      可变参数：定义参数时不知道有多少参数
 *      格式：
 *                      修饰符 返回值类型 方法名(数据类型... 变量名){}
 *      这里的变量是一个数组
 *      可变参数一定是最后一个
 */
public class ArgsDemo {
	public static void main(String[] args) {
		int a = 10;
		int b = 20;
		int result = sum(a, b);
		System.out.println("result=" + result);

		int c = 30;
		result = sum(a, b, c);
		System.out.println("result=" + result);
	}

	public static int sum(int... a) {
		int s=0;
		for(int x:a) {
			s+=s;
		}
		return s;
	}
}
```

asList

2018年7月15日 20:43

```
import java.util.Arrays;
import java.util.List;

/*
 * public static <T> List<T> asList(T ... a):把数组转为集合
 *       集合的长度不能改变，因为该集合的本质是数组
 */
public class ArraysDemo {
	public static void main(String[] args) {
		String[] strArray = { "hello", "world", "java" };

		// List<String> list=Arrays.asList(strArray);
		List<String> list = Arrays.asList("hello", "world", "java");
		// list.add("javaee");//UnsupportedOperationException
		// list.remove("java");//UnsupportedOperationException
		list.set(1, "welcome");
		for (String s : list) {
			System.out.println(s);
		}
	}
}
```

集合嵌套遍历

2018年7月15日 21:06

```
package cn.itcast_12;

import java.util.ArrayList;

import cn.itcast_06.Student;

/*
 *      集合的嵌套遍历
 */
public class ArrayListDemo {
	public static void main(String[] args) {
		// 创建大集合
		ArrayList<ArrayList<Student>> bigArray = new
		ArrayList<ArrayList<Student>>();

		// 创建第一个小集合
		ArrayList<Student> firstArray = new ArrayList<Student>();
		Student s1 = new Student("唐僧", 30);
		Student s2 = new Student("孙悟空", 29);
		Student s3 = new Student("猪八戒", 28);
		Student s4 = new Student("沙僧", 27);
		Student s5 = new Student("白龙马", 26);

		// 学生进班
		firstArray.add(s1);
		firstArray.add(s2);
		firstArray.add(s3);
		firstArray.add(s4);
		firstArray.add(s5);

		// 班进系统
		bigArray.add(firstArray);

		// 创建第一个小集合
		ArrayList<Student> secondArray = new ArrayList<Student>();
		Student s11 = new Student("诸葛亮", 30);
```

```
        Student s22 = new Student("周瑜", 29);
        Student s33 = new Student("司马懿", 28);

        // 学生进班
        secondArray.add(s11);
        secondArray.add(s22);
        secondArray.add(s33);

        // 班进系统
        bigArray.add(secondArray);

        // 创建第一个小集合
        ArrayList<Student> thirdArray = new ArrayList<Student>();
        Student s111 = new Student("宋江", 40);
        Student s222 = new Student("吴用", 29);

        // 学生进班
        thirdArray.add(s111);
        thirdArray.add(s222);

        // 班进系统
        bigArray.add(thirdArray);

        // 遍历
        for (ArrayList<Student> arr : bigArray)
            for (Student s : arr) {
                System.out.println(s.getName() + "---" + s.getAge());
            }

    }
}
```

练习

2018年7月15日 21:33

```
import java.util.ArrayList;
import java.util.Random;

//获取10个1~20之间的随机数，要求不能重复
public class RandomDemo {
	public static void main(String[] args) {
		Random r = new Random();
		ArrayList<Integer> array = new ArrayList<Integer>();

		int count = 0;
		while (count < 10) {
			int number = r.nextInt(20) + 1;
			if (!array.contains(number)) {
				array.add(number);
				count++;
			}
		}

		for (Integer i : array) {
			System.out.println(i);
		}
	}
}
```

练习

2018年7月15日 22:19

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

/*
 *      键盘录入多个数据，以0结束，要求在控制台输出这些数据中最大值
 */
public class ArrayListDemo {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);

		ArrayList<Integer> array = new ArrayList<Integer>();

		while(true) {
			System.out.println("请输入数据");
			int num=sc.nextInt();
			if(num!=0) {
				array.add(num);
			}else {
				break;
			}
		}

		Integer[] i = new Integer[array.size()];
		//Integer[] ii=array.toArray(i);
		array.toArray(i);

		Arrays.sort(i);
		System.out.println(i[i.length-1]);
	}
}
```