



Java 核心技术

第十章 Java数据结构

第六节 工具类

华东师范大学 陈良育



JCF的工具类

- JCF中工具类
 - 不存储数据，而是在数据容器上，实现高效操作
 - 排序
 - 搜索
 - Arrays 类
 - Collections 类

Arrays



- Arrays: 处理对象是数组
 - 排序: 对数组排序, sort/parallelSort。
 - 查找: 从数组中查找一个元素, binarySearch。
 - 批量拷贝: 从源数组批量复制元素到目标数组, copyOf。
 - 批量赋值: 对数组进行批量赋值, fill。
 - 等价性比较: 判定两个数组内容是否相同, equals。
 - 查看ArraysTest.java。

包装器类



- Collections: 处理对象是 Collection 及其子类
 - 排序：对List进行排序，sort。
 - 搜索：从List中搜索元素，binarySearch
 - 批量赋值：对List批量赋值，fill。
 - 最大、最小：查找集合中最大/小值，max，min
 - 反序：将List 反序排列，reverse
 - 查看CollectionsTest.java。



对象比较

- 对象实现Comparable接口（需要修改对象类）
 - compareTo方法
 - > 返回1, ==返回0, <返回-1
 - Arrays和Collections在进行对象sort时, 自动调用该方法
- 新建Comparator（适用于对象类不可更改的情况）
 - compare方法
 - > 返回1, ==返回0, <返回-1
 - Comparator比较器将作为参数提交给工具类的sort方法
- 查看Person.java和Person2Comparator.java

总结



- Arrays和Collections功能强大，不需要重复造轮子
- 对象比较方法Comparable/Comparator



代码(1) ArrayTest.java

```
public class ArraysTest {
    public static void main(String[] args) {
        testSort();
        testSearch();
        testCopy();
        testFill();
        testEquality();
    }
    public static void testSort() {
        Random r = new Random();
        int[] a = new int[10];
        for(int i=0;i<a.length;i++) {
            a[i] = r.nextInt();
        }
        System.out.println("=====测试排序=====");
        System.out.println("排序前");
        for(int i=0;i<a.length;i++) {
            System.out.print(a[i] + ",");
        }
        System.out.println();
        System.out.println("排序后");
        Arrays.sort(a);
        for(int i=0;i<a.length;i++) {
            System.out.print(a[i] + ",");
        }
        System.out.println();
    }
}
```



代码(2) ArrayTest.java

```
public static void testSearch() {  
    Random r = new Random();  
    int[] a = new int[10];  
    for(int i=0;i<a.length;i++)  
    {  
        a[i] = r.nextInt();  
    }  
    a[a.length-1] = 10000;  
    System.out.println("=====测试查找=====");  
    System.out.println("10000 的位置是" + Arrays.binarySearch(a, 10000));  
}
```




代码(3) ArrayTest.java

```
public static void testCopy() {  
    Random r = new Random();  
    int[] a = new int[10];  
    for(int i=0;i<a.length;i++)  
    {  
        a[i] = r.nextInt();  
    }  
    int[] b = Arrays.copyOf(a, 5);  
    System.out.println("=====测试拷贝前五个元素=====");  
    System.out.print("源数组: ");  
    for(int i=0;i<a.length;i++)  
    {  
        System.out.print(a[i] + ",");  
    }  
    System.out.println();  
    System.out.print("目标数组: ");  
    for(int i=0;i<b.length;i++)  
    {  
        System.out.print(b[i] + ",");  
    }  
    System.out.println();  
}
```



代码(4) ArrayTest.java

```
public static void testFill() {  
    int[] a = new int[10];  
    Arrays.fill(a, 100);  
    Arrays.fill(a, 2, 8, 200);  
    System.out.println("=====测试批量赋值=====");  
    System.out.print("数组赋值后: ");  
    for(int i=0;i<a.length;i++)  
    {  
        System.out.print(a[i] + ",");  
    }  
    System.out.println();  
}  
public static void testEquality() {  
    int[] a = new int[10];  
    Arrays.fill(a, 100);  
    int[] b = new int[10];  
    Arrays.fill(b, 100);  
    System.out.println(Arrays.equals(a, b));  
    b[9] = 200;  
    System.out.println(Arrays.equals(a, b));  
}  
}
```



代码(5) CollectionsTest.java

```
public class CollectionsTest {  
  
    public static void main(String[] args) {  
        ArrayList<Integer> list = new ArrayList<Integer>();  
        list.add(1);  
        list.add(12);  
        list.add(2);  
        list.add(19);  
  
        // 排序  
        Collections.sort(list);  
        // 检索  
        System.out.println("元素所在的索引值是: " + Collections.binarySearch(list, 12));  
        // 最大最小  
        System.out.println("最大值: " + Collections.max(list));  
        System.out.println("最小值: " + Collections.min(list));  
        Collections.reverse(list); // 翻转不需要用到排序  
  
        Collections.fill(list, 100); // 全部赋值为100  
    }  
}
```


代码(6) Person.java



```
public class Person implements Comparable<Person> {  
    String name;  
    int age;  
  
    public String getName() {  
        return name;  
    }  
  
    public int getAge() {  
        return age;  
    }  
  
    public Person(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
}
```


代码(7) Person.java



```
public int compareTo(Person another) {
    int i = 0;
    i = name.compareTo(another.name); // 使用字符串的比较
    if (i == 0) {
        // 如果名字一样,比较年龄, 返回比较年龄结果
        return age - another.age;
    } else {
        return i; // 名字不一样, 返回比较名字的结果.
    }
}

public static void main(String... a) {
    Person[] ps = new Person[3];
    ps[0] = new Person("Tom", 20);
    ps[1] = new Person("Mike", 18);
    ps[2] = new Person("Mike", 20);

    Arrays.sort(ps);
    for (Person p : ps) {
        System.out.println(p.getName() + "," + p.getAge());
    }
}
```

代码(8) Person2.java



```
public class Person2 {  
    private String name;  
    private int age;  
    public String getName() {  
        return name;  
    }  
    public int getAge() {  
        return age;  
    }  
    public Person2(String name, int age)  
    {  
        this.name = name;  
        this.age = age;  
    }  
}
```



代码(9) Person2Comparator.java

```
public class Person2Comparator implements Comparator<Person2> {  
    public int compare(Person2 one, Person2 another) {  
        int i = 0;  
        i = one.getName().compareTo(another.getName());  
        if (i == 0) {  
            // 如果名字一样,比较年龄,返回比较年龄结果  
            return one.getAge() - another.getAge();  
        } else {  
            return i; // 名字不一样,返回比较名字的结果.  
        }  
    }  
}
```


代码(10) Person2Comparator.java



```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    Person2[] ps = new Person2[3];  
    ps[0] = new Person2("Tom", 20);  
    ps[1] = new Person2("Mike", 18);  
    ps[2] = new Person2("Mike", 20);  
  
    Arrays.sort(ps, new Person2Comparator());  
    for (Person2 p : ps) {  
        System.out.println(p.getName() + "," + p.getAge());  
    }  
}
```




谢谢!