JAVA容器

果核科技
CORE TECH.

JAVA SE基础

- 容器的概念和API
- Collection 接口
- Iterator 接口
- 增强的for循环
- Set 接口
- List 接口 和 Comparable 接口
- Collections类

容器的概念

阅读如下程序：

```
public class Name {
    private String firstName,lastName;
    public Name(String firstName, String lastName) {
        this.firstName = firstName; this.lastName = lastName;
    }
    public String getFirstName() {  return firstName;   }
    public String getLastName() {   return lastName;   }
    public String toString() {  return firstName + ""+ lastName;  }
}
```

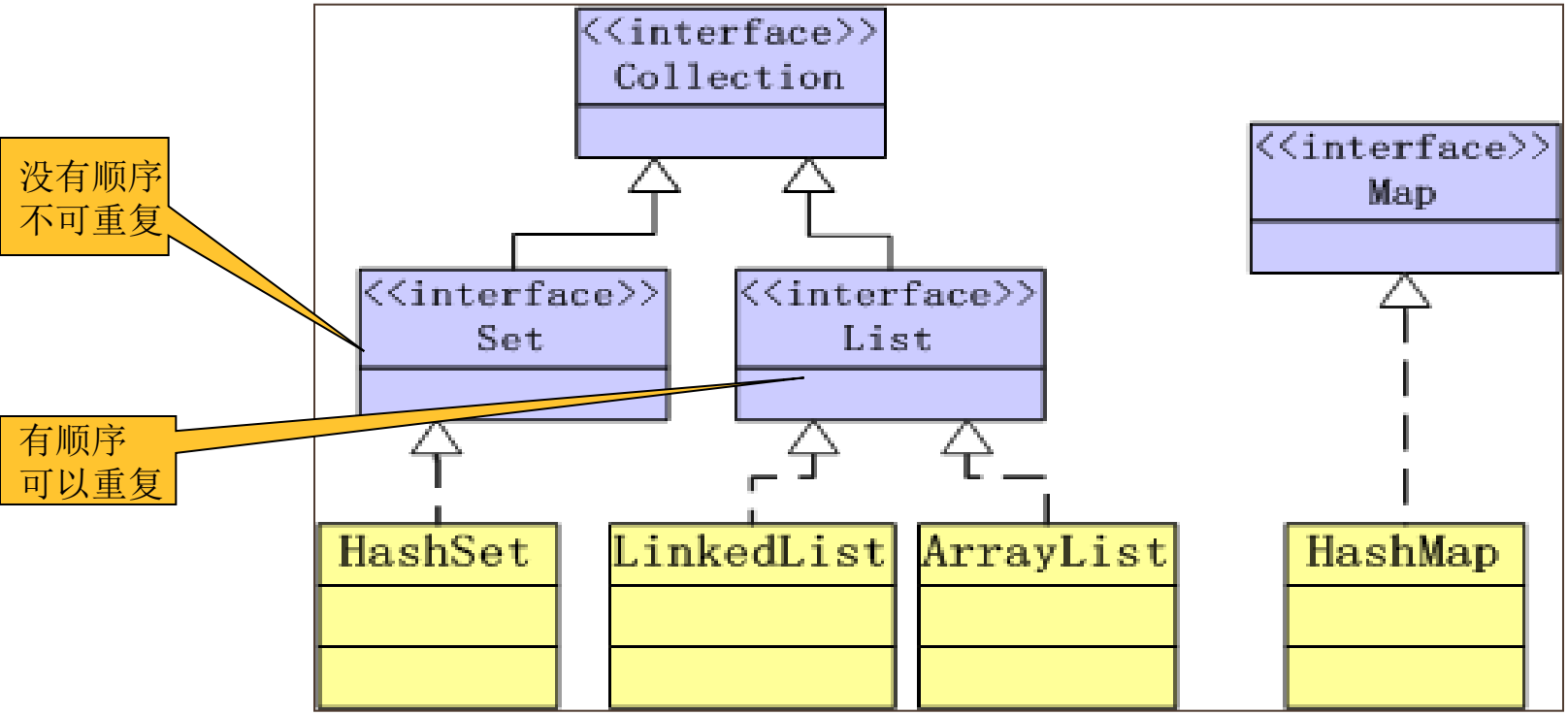
```
public class Test {
    public static void main(String arg[]) {
        Name name1 = new Name("f1","l1");
        Name name2 = new Name("f2","l2");
        Name name3 = new Name("f3","l3");
                        … … …
    }
}
```

容器：

Java API 所提供的一系列类的实例，用于在程序中存放对象。

容器 API

J2SDK所提供的容器API位于 java.util 包内。
容器API的类图结构如下图所示：



容器 API

- Collection 接口一定义了存取一组对象的方法，其子接口Set和List分别定义了存储方式。
  - **Set 中的数据对象没有顺序且不可以重复。**
  - **List 中的数据对象有顺序且可以重复。（即互相equals）**
- Map 接口定义了存储“键（key）— 值（value）映射对”的方法。

Collection 接口

Collection 接口中所定义的方法：

```
int size();
boolean isEmpty();
void clear();
boolean contains(Object element);   //equals
boolean add(Object element);
boolean remove(Object element);

boolean containsAll(Collection c);
boolean addAll(Collection c);
boolean removeAll(Collection c);
boolean retainAll(Collection c);   //求交集
```

- boolean remove(Object element);

  add自定义对象后，remove如何删除？

  自定义对象实现equals必须实现hashcode方法

- Object[] toArray();

  使用每个元素都要强制转换 泛型
- Iterator iterator();
- 容器内元素遍历

Set 接口

- **Set 接口是Collection的子接口，Set接口没有提供额外的方法，但实现 Set 接口的容器类中的元素是没有有顺序的，而且不可以重复。**
- **Set 容器可以与数学中“集合”的概念相对应**
- **J2SDK API中 所提供的 Set 容器类有 HashSet，TreeSet 等。**

Set 方法举例


```
public static void main(String[] args) {
    Set s = new HashSet();
    s.add("hello");
    s.add("world");
    s.add(new Name("f1","f2"));
    s.add(new Integer(100));
    s.add(new Name("f1","f2")); //相同元素不会被加入
    s.add("hello"); //相同的元素不会被加入
    System.out.println(s);
}
```

输出结果：

```
[100, hello, world, f1 f2]
```

Set 方法举例

果核科技 CORE TECH.

```
public static void main(String[] args) {
   Set s1 = new HashSet();
   Set s2 = new HashSet();
   s1.add("a");s1.add("b");s1.add("c");
   s2.add("d");s2.add("a");s2.add("b");
   //Set和List容器类都具有Constructor(Collection c)
   //构造方法用以初始化容器类
   Set sn = new HashSet(s1);
   sn.retainAll(s2);      //求交集
   Set su = new HashSet(s1);
   su.addAll(s2);
   System.out.println(sn);
   System.out.println(su);
}
```

输出结果：

```
[a, b]
[d, a, c, b]
```

List 接口

- List接口是Collection的子接口，实现List接口的容器类中的元素是有顺序的，而且可以重复。
- List 容器中的元素都对应一个整数型的序号记载其在容器中的位置，可以根据序号存取容器中的元素。
- J2SDK 所提供的 List 容器类有 ArrayList，LinkedList 等

```
Object get(int index);
Object set(int index, Object element); //返回旧的元素
void add(int index, Object element);
Object remove(int index);
int indexOf(Object o);
int lastIndexOf(Object o);
```

List 方法举例


```
List l1 = new LinkedList();
for(int i=0; i<=5; i++) {
    l1.add("a"+i);
}
System.out.println(l1);
l1.add(3,"a100");
System.out.println(l1);
l1.set(6,"a200");
System.out.println(l1);
System.out.print((String)l1.get(2)+ " ");
System.out.println(l1.indexOf("a3"));
l1.remove(1);
System.out.println(l1);
```

输出结果：

```
[a0, a1, a2, a3, a4, a5]
[a0, a1, a2, a100, a3, a4, a5]
[a0, a1, a2, a100, a3, a4, a200]
a2 4
[a0, a2, a100, a3, a4, a200]
```

hashCode

果核科技 CORE TECH.

- 一个对象被当作Map里面的key的时候,hashCode用来比较两个对象是不是相等
- hashCode非常适合用来做索引
- hashCode 的常规协定
- 重写equals方法,通常需要重写hashCode方法.因为你的本意是想让他相等的.但hashCode如果不重写,不同的对象就不会相等
- Hashcode被谁调用了?什么时候调用的?

Collection 方法举例


```
import java.util.*;
public class Test {
    public static void main(String[] args) {
        Collection c = new ArrayList();
        //可以放入不同类型的对象
        c.add("hello");
        c.add(new Name("f1","l1"));
        c.add(new Integer(100));
        System.out.println(c.size());
        System.out.println(c);
    }
}
```

输出结果：

```
3
[hello, f1 l1, 100]
```

Collection 方法举例


```
import java.util.*;
public class Test {
    public static void main(String[] args) {
        Collection c = new HashSet();
        c.add("hello");
        c.add(new Name("f1","l1"));
        c.add(new Integer(100));
        c.remove("hello");
        c.remove(new Integer(100));
        System.out.println
                    (c.remove(new Name("f1","l1")));
        System.out.println(c);
    }
}
```

输出结果：

```
false
[f1 l1]
```

Collection 方法举例

- 容器类对象在调用remove、contains 等方法时需要比较对象是否相等，这会涉及到对象类型的 equals 方法和hashCode(hash容器)方法；对于自定义的类型，需要要重写equals 和 hashCode 方法以实现自定义的对象相等规则。
- 注意：相等的对象应该具有相等的 hash codes。
- 增加Name类的equals和hashCode方法如下：

```
public boolean equals(Object obj) {
    if (obj instanceof Name) {
        Name name = (Name) obj;
        return (firstName.equals(name.firstName))
            && (lastName.equals(name.lastName));
    }
    return super.equals(obj);
}
public int hashCode() {
    return firstName.hashCode();
}
```

Collection 方法举例

使用更新的Name类，运行下列程序：

```
import java.util.*;
public class Test {
    public static void main(String[] args) {
        Collection c = new LinkedList();
        c.add(new Name("f1","l1"));
        c.add(new Name("f2","l2"));
        System.out.println(c.contains
                        (new Name("f2","l2")));
        c.remove(new Name("f1","l1"));
        System.out.println(c);
    }
}
```

输出结果：

```
true
[f2 l2]
```

ArrayList为什么有两个remove方法？

- ArrayList为什么有两个remove方法？
- remove(int index) 是子类新添加的！
- 如果是父类引用指向子类对象，则找不到子类中新添加的方法。
- 要想调用子类新添加的方法，需要先转型
- 是优先自动打包呢？还是先去匹配方法？

```
list.add(12);
list.remove(12);
```

Auto-boxing/unboxing

果核科技 CORE TECH.

- 在合适的时机自动打包、解包
  - 自动将基础类型转换为对象
  - 自动将对象转换为基础类型

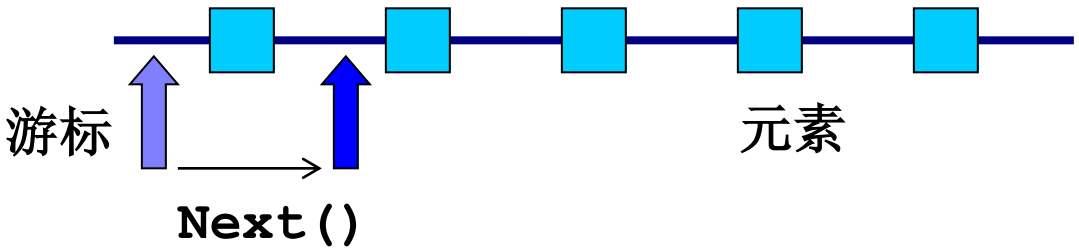
```
int a = 12;
Integer b = Integer.valueOf(a);
int c = b.intValue();
```

```
int a = 12;
Integer b = a;
int c = b;
```

Iterator 接口

- 所有实现了Collection接口的容器类都有一个iterator方法用以返回一个实现了Iterator接口的对象。
- Iterator对象称作迭代器，用以方便的实现对容器内元素的遍历操作。
- Iterator接口定义了如下方法：
- Enumeration ArrayList - Vector

```
boolean hasNext();   //判断游标右边是否有元素
Object next();       //返回游标右边的元素并将游标移动到下一个位置
void remove();       //删除游标左面的元素，在执行完next之后该
                     //操作只能执行一次
```



Iterator 方法举例


```
import java.util.*;
public class Test {
    public static void main(String[] args) {
        Collection c = new HashSet();
        c.add(new Name("f1","l1"));
        c.add(new Name("f2","l2"));
        c.add(new Name("f3","l3"));
        Iterator i = c.iterator();
        while(i.hasNext()) {
        //next()的返回值为Object类型，需要转换为相应类型
            Name n = (Name)i.next();
            System.out.print(n.getFirstName()+" ");
        }
    }
}
```

输出结果：

```
f2 f1 f3
```

Iterator 方法举例

Iterator对象的remove方法是在迭代过程中删除元素的唯一的安全方法。

```
                    … … …
Collection c = new HashSet();
c.add(new Name("fff1","llll"));
c.add(new Name("f2","l2"));
c.add(new Name("fff3","llll3"));
for(Iterator i = c.iterator();i.hasNext();) {
   Name name =(Name)i.next();
   if(name.getFirstName().length()<3){
      i.remove();
      //如果换成 c.remove(name); 会产生例外
   }
}
System.out.println(c);
```

输出结果:

```
[fff3 llll3, fff1 llll]
```

JDK1.5增强的for循环

- 增强的for循环对于遍历array 或 Collection的时候相当简便
  - 示例 EnhancedFor.java
- 缺陷：
  - 数组：
    - 不能方便的访问下标值
  - 集合：
    - 与使用Iterator相比，不能方便的删除集合中的内容
- 总结：
  - 除了简单遍历并读出其中的内容外，不建议使用增强for

JDK1.5泛型

- 起因：
  - JDK1.4以前类型不明确：
    - 装入集合的类型都被当作Object对待，从而失去自己的实际类型。
    - 从集合中取出时往往需要转型，效率低，容易产生错误。
- 解决办法：
  - 在定义集合的时候同时定义集合中对象的类型
  - 示例：BasicGeneric.java
    - 可以在定义Collection的时候指定
    - 也可以在循环时用Iterator指定
- 好处：
  - 增强程序的可读性和稳定性

Map 接口

- 实现Map接口的类用来存储键－值对。
- Map 接口的实现类有HashMap和TreeMap等。
- Map类中存储的键－值对通过键来标识，所以键值不能重复。

```
Object put(Object key, Object value);
Object get(Object key);
Object remove(Object key);
boolean containsKey(Object key);
boolean containsValue(Object value);
int size();
boolean isEmpty();
void putAll(Map t);
void clear();
```

Collection 常用算法

类 java.util.Collections
提供了一些静态方法实现了基于List容器的一些常用算法。

void sort(List)　对List容器内的元素排序

void shuffle(List)　对List容器内的对象进行随机排列

void reverse(List)　对List容器内的对象进行逆续排列

void fill(List, Object)　用特定的对象初始化整个List

将src List容器内容拷贝到dest List容器
void copy(List dest,List src)
注意：目标容器长度不能小于源容器

对于顺序的List容器，采用折半查找的方法查找特定对象
int binarySearch(List, Object)

Collection 常用算法


```
List l1 = new LinkedList();
List l2 = new LinkedList();
for(int i=0; i<=9; i++) { l1.add("a"+i); }
System.out.println(l1);
Collections.shuffle(l1); //随机排列
System.out.println(l1);
Collections.reverse(l1); //逆续
System.out.println(l1);
Collections.sort(l1); //排序
System.out.println(l1);
System.out.println
   (Collections.binarySearch(l1,"a5")); //折半查找
```

输出结果：

```
[a0, a1, a2, a3, a4, a5, a6, a7, a8, a9]
[a1, a3, a8, a9, a4, a6, a5, a2, a0, a7]
[a7, a0, a2, a5, a6, a4, a9, a8, a3, a1]
[a0, a1, a2, a3, a4, a5, a6, a7, a8, a9]
5
```

数组的常用算法

Java.util.Arrays

- sort
- binarySearch
- toString
- copyOf
- fill
- equals
- hashCode 数组对象的hashcode方法

Comparable 接口

- 问题：上面的算法根据什么确定容器中对象的“大小”顺序？
- 所有可以“排序”的类都实现了java.lang.Comparable 接口，Comparable接口中只有一个方法

  public int compareTo(Object obj)；该方法：

- 返回 0 表示 this == obj
- 返回正数表示 this > obj
- 返回负数表示 this < obj

- 实现了Comparable 接口的类通过实现 comparaTo 方法从而确定该类对象的排序方式。

Comparable 接口

- 改写 Name 类 让其实现 Comparable 接口，其 compareTo 方法定义为：

```
class Name implements Comparable {
                    … … …
    public int compareTo(Object o) {
        Name n = (Name)o;
        int lastCmp =
            lastName.compareTo(n.lastName);
        return
             (lastCmp!=0 ? lastCmp :
              firstName.compareTo(n.firstName));
    }
}
```

Comparable 接口

使用新的 Name 类 运行下列程序：

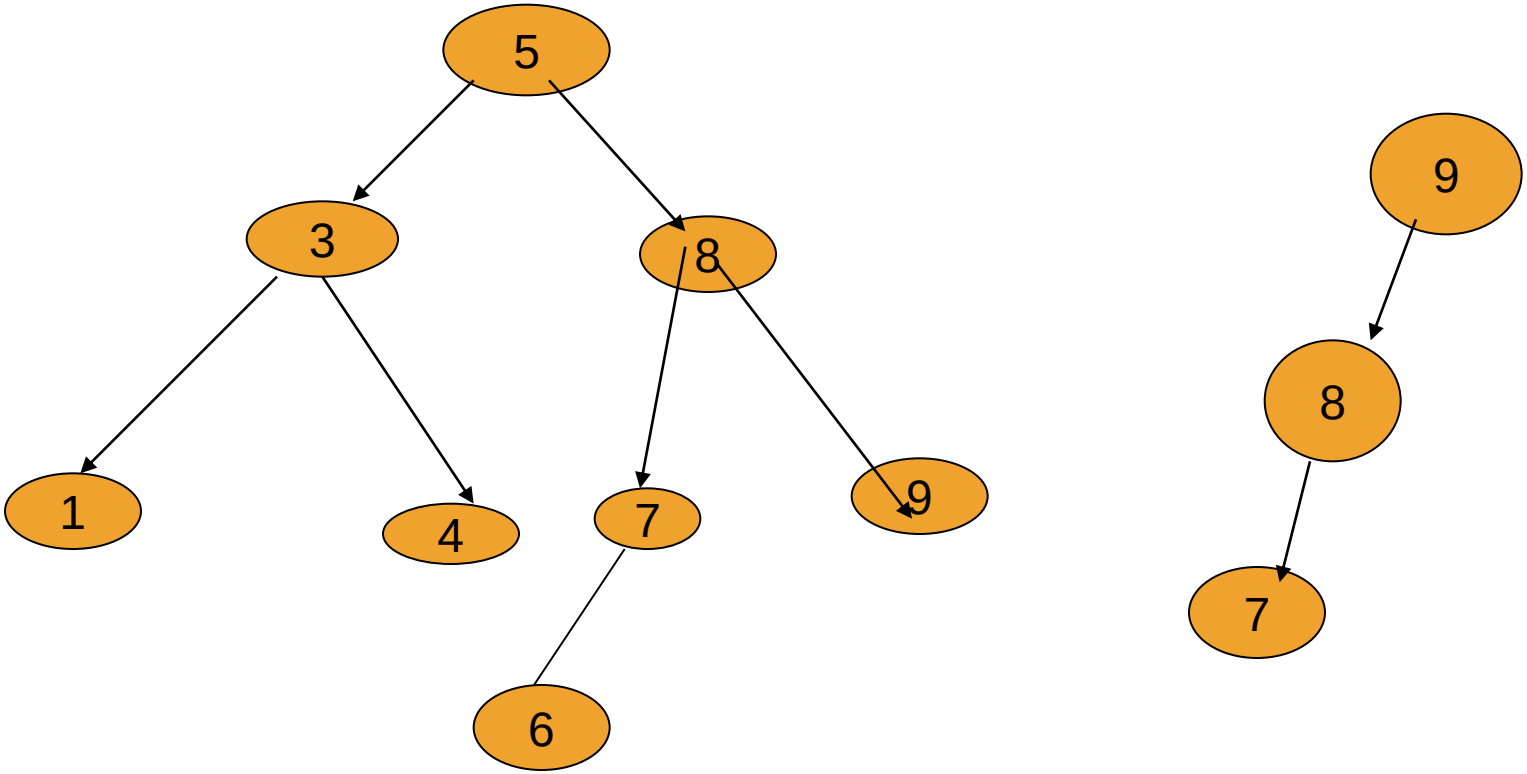
```
List l1 = new LinkedList();
l1.add(new Name("Karl", "M"));
l1.add(new Name("Steven", "Lee"));
l1.add(new Name("John", "O"));
l1.add(new Name("Tom", "M"));
System.out.println(l1);
Collections.sort(l1);
System.out.println(l1);
```

输出结果：

```
[Karl M, Steven Lee, John O, Tom M]
[Steven Lee, Karl M, Tom M, John O]
```

如何选择数据结构*

- Array读快改慢
- Linked改快读慢
- Hash搜索极快，遍历极慢
- Tree插入/搜索都比较快，适合做索引



总结

果核科技 CORE TECH.

- **Java Collections Framework**
  - 一个图
    - Set 无序，不可**重复** / List有序，可以**重复** / Map
  - 两个类：JCF框架中还有两个很实用的公用类：Collections和Arrays
    - Collections提供了对一个Collection容器进行诸如排序、复制、查找和填充等方法
    - Arrays 提供了数组的填充，查找，比较，排序等一系列的对数组的操作
  - 三个知识点
    - Enhanced For →不重要
    - Generic 范型 →增强程序的可读性和稳定性
    - Auto-boxing / unboxing →谨慎使用
  - 六个接口
    - Collection
    - Set
    - List
    - Map
    - Comparable
    - Iterator

作业

作业1: 有一个字符串，有很多重复出现的字符，例如”abcdaaabcddddfsfdsfddfgghjgkhj”
找出出现次数做多的字符，打印该字符并打印出现次数

作业2: 一个都是字母的字符串，把大小转成小写，小写转换成大写
例如: “sdfFsfdFGsdLFsdfeoLDK”

作业3: 对一个字串中的所有数字求和
例如："123abcd22xyz45" 求123+22+45

作业4: 控制台贪吃蛇 提示：休眠 Thread.sleep(毫秒)

```
------------------------------
|                            |
|                @           |
|     #                      |
|     #                      |
|     ###                    |
|       #                    |
|       #                    |
|                            |
------------------------------
```

1. 至少吃掉一个@，并产生新的@ ； 2. 死掉提示；3. 支持方向控制