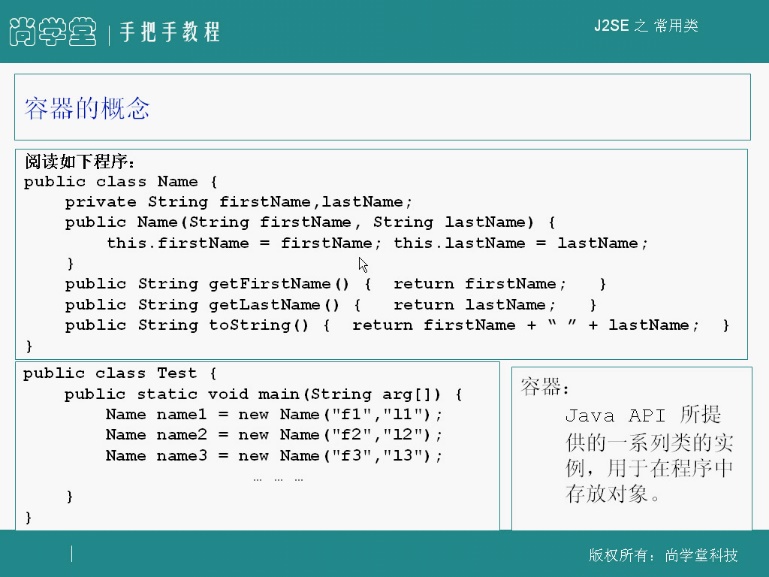
05\_容器\_Iterator\_笔记

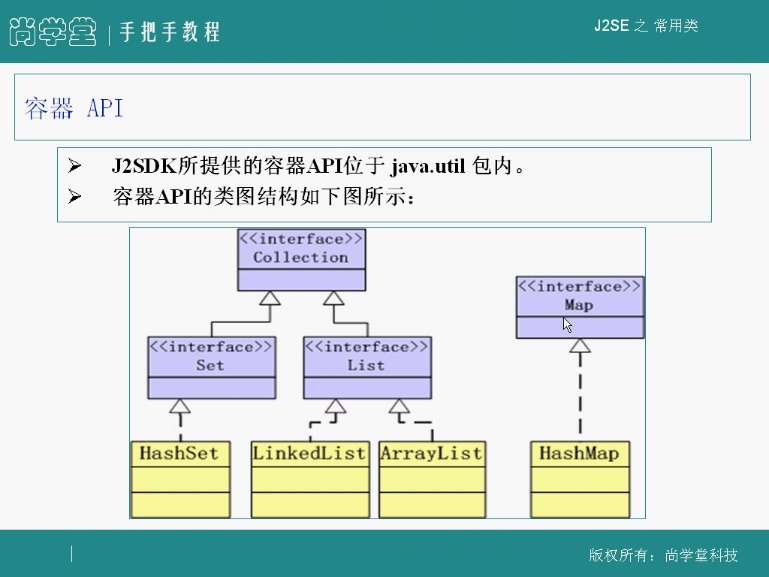
**一、容器**

**1. 容器的概念**



* 定义：容器是可以存放其他对象的对象，就像桶或篮子能装东西一样
* 示例：数组就是一种容器，如Name name1 = new Name("f1","11")创建的对象存放在容器中
* 特点：Java API提供了一系列容器类，专门用于在程序中存放对象

**1）容器API的类图结构**



* 分类体系：
  + Collection接口：包含Set和List两大子接口
  + Map接口：独立于Collection的键值对容器
* 核心区别：
  + Set：元素不可重复且无序，如HashSet
  + List：元素可重复且有序，如ArrayList、LinkedList
  + Map：存储键值对，如HashMap、TreeMap
* 实现关系：具体容器类（如HashSet）实现了这些接口，使用时需创建这些类的对象

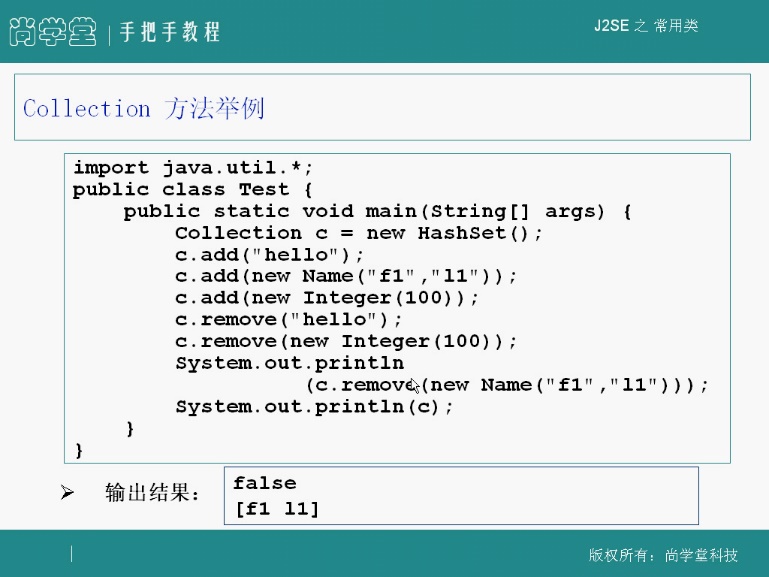
**2. Collection接口 ﻿**

**1）Collection接口方法的介绍**

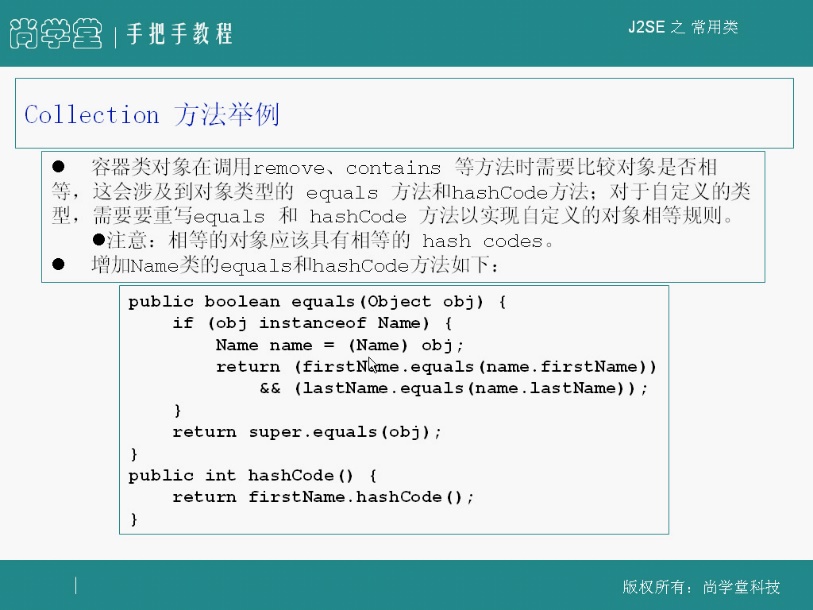


* 基本操作：
  + 增删：add(Object)添加元素，remove(Object)删除元素
  + 查询：size()获取元素数量，isEmpty()判断是否为空
  + 批量操作：addAll(Collection)批量添加，removeAll(Collection)批量删除
* 特殊方法：
  + 迭代器：iterator()获取遍历容器的迭代器
  + 转换：toArray()将容器转为对象数组
  + 集合运算：retainAll(Collection)保留交集

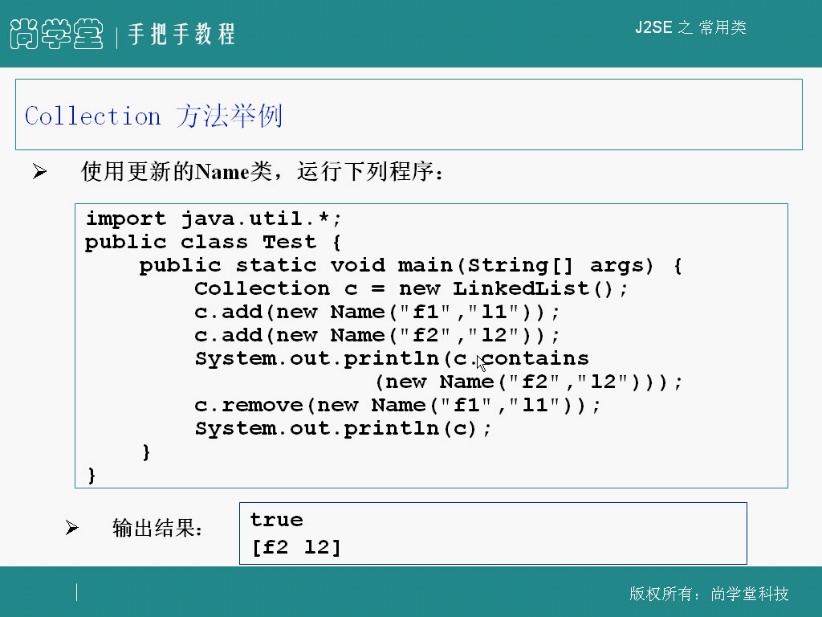
**2）Collection方法举例**



* 实际应用：
  + 创建HashSet容器：Collection c = new HashSet()
  + 添加多种对象：字符串、自定义类实例、整型对象
  + 删除操作：remove("hello")会调用元素的equals方法比较
* 输出说明：示例输出false [f1 11]展示删除操作返回值及剩余内容



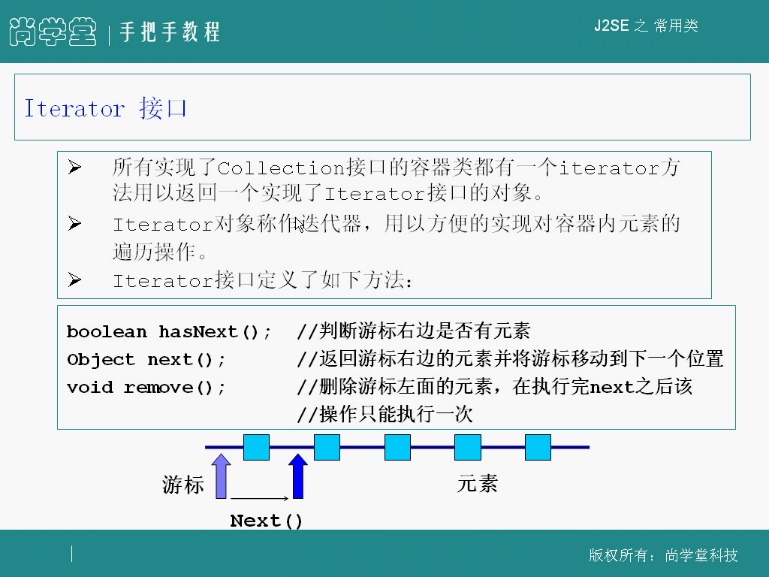
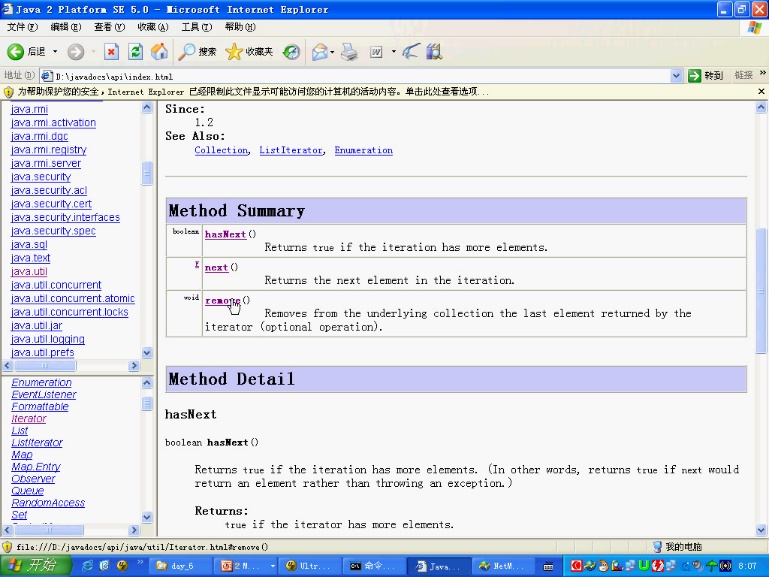
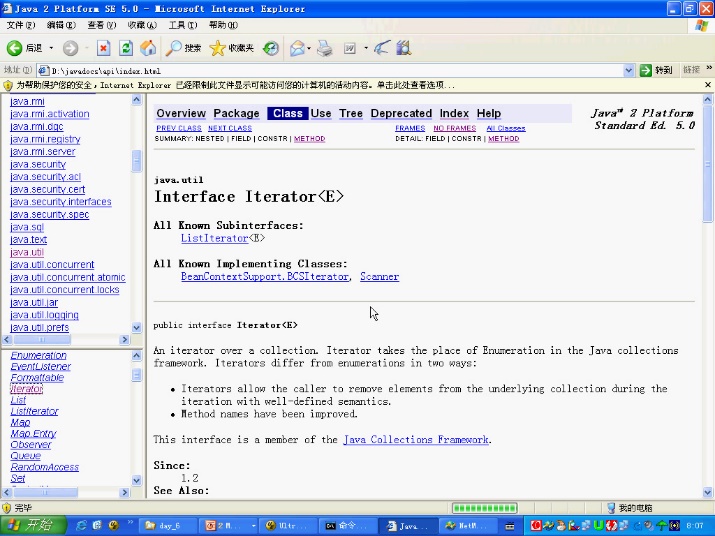
* 对象比较机制：
  + equals重写：自定义类需重写equals方法定义相等规则，如Name类比较firstName和lastName
  + hashCode契约：相等的对象必须具有相等的hashCode，如return firstName.hashCode()
  + 使用场景：当对象作为Map的键时，hashCode用于快速查找



* 完整示例：
  + 使用LinkedList容器存储Name对象
  + contains()和remove()方法验证：输出true表示成功找到对象，[f2 12]展示最终容器内容
* 实现要点：正确实现equals和hashCode后，容器操作才能按预期工作

**3. Iterator接口 ﻿**

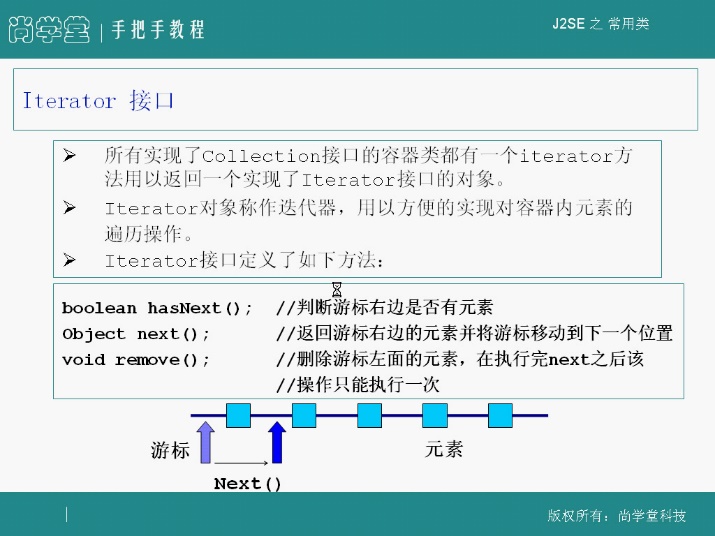
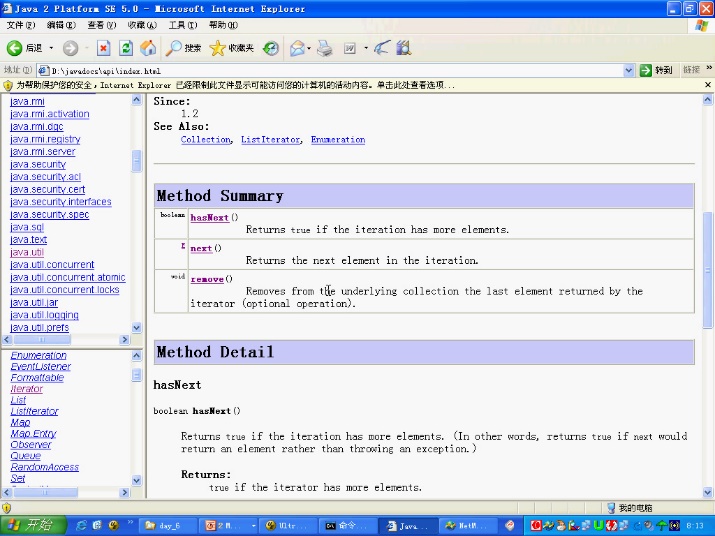
**1）Iterator接口方法的介绍**

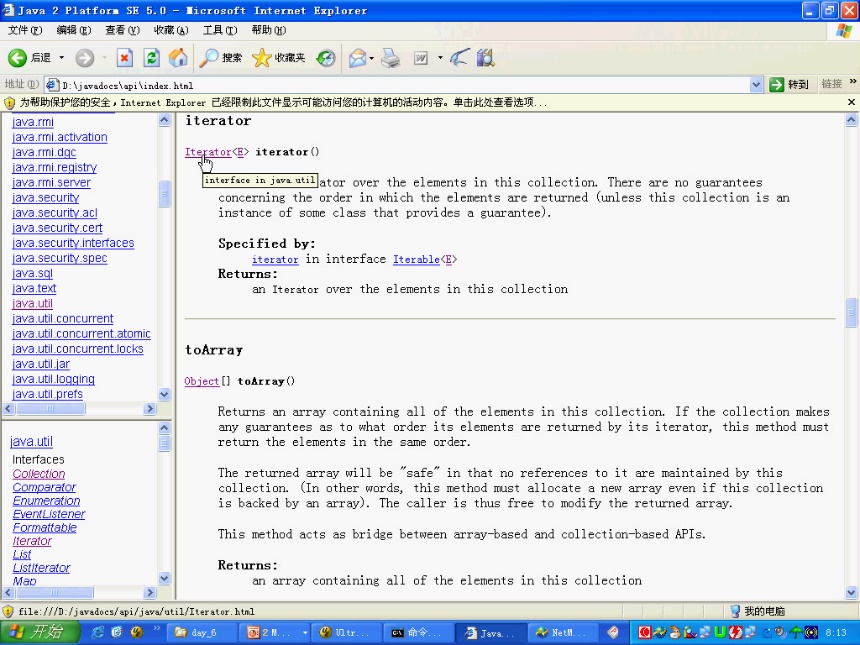
* 
* 接口定义：Iterator是Java集合框架中的一个接口，用于遍历集合中的元素。
* 发音说明：专业读法为"iterator"，而非"itert"。
* 获取方式：所有实现了Collection接口的容器类都有一个iterator()方法，用于返回实现了Iterator接口的对象。
* 
* 核心方法：
  + hasNext()：判断游标右边是否有元素，返回布尔值。当next()方法能返回元素而不抛出异常时返回true。
  + next()：返回游标右边的元素并将游标移动到下一个位置。
  + remove()：删除游标左边的元素，必须在执行next()之后才能调用，且每次next()后只能执行一次。
* 设计特点：
  + 接口只定义方法声明，具体实现由子类完成
  + 主要用途是提供统一的集合遍历方式
  + 相比Collection接口的直接操作方法，Iterator更专注于元素的遍历访问
* 注意事项：
  + remove()是可选操作，某些实现可能不支持
  + 使用remove()前必须先调用next()
  + 每次next()后只能调用一次remove()
  + 
* 框架关系：Iterator接口是Java集合框架的六个核心接口之一，其他五个是Collection、Set、List、Map和ListIterator。
* 历史演变：Iterator取代了早期的Enumeration接口，主要改进包括：
  + 增加了安全的元素删除功能
  + 使用了更直观的方法命名

**2）Collection接口方法总结 ﻿**

* Collection接口概述



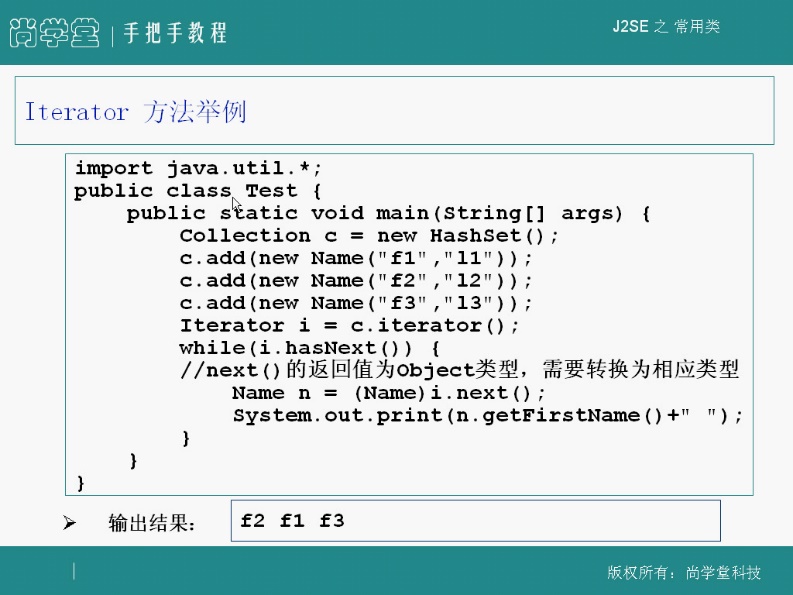
* + 核心功能：Collection是Java集合框架的根接口，代表一组对象（元素），允许重复元素和空值
  + 底层实现差异：不同实现类（如ArrayList和LinkedList）底层数据结构不同（数组vs链表），导致遍历方式不同
  + 与数组对比：相比数组，集合没有长度限制，可以动态添加元素
* List接口及其实现类
  + ArrayList特点：
    - 底层使用数组实现
    - 随机访问快（通过索引直接定位）
    - 遍历简单（通过数组长度循环）
  + LinkedList特点：
    - 底层使用链表实现
    - 元素在内存中不连续
    - 每个元素保存下一个元素的引用
    - 遍历需要从头节点开始逐个访问
* Iterator接口的作用
  + 
  + 设计目的：为解决不同集合实现类遍历方式不统一的问题
  + 实现原理：采用迭代器模式，让集合自身返回遍历器对象
  + 多语言通用性：类似C++ STL中的iterator概念
* Iterator接口的方法
  + 
  + hasNext()：判断游标右侧是否有元素
  + next()：返回右侧元素并移动游标
  + remove()：删除游标左侧元素（需在next()后调用）
  + 游标机制：初始位置在第一个元素左侧，每次next()右移一位
* Collection接口中的iterator方法



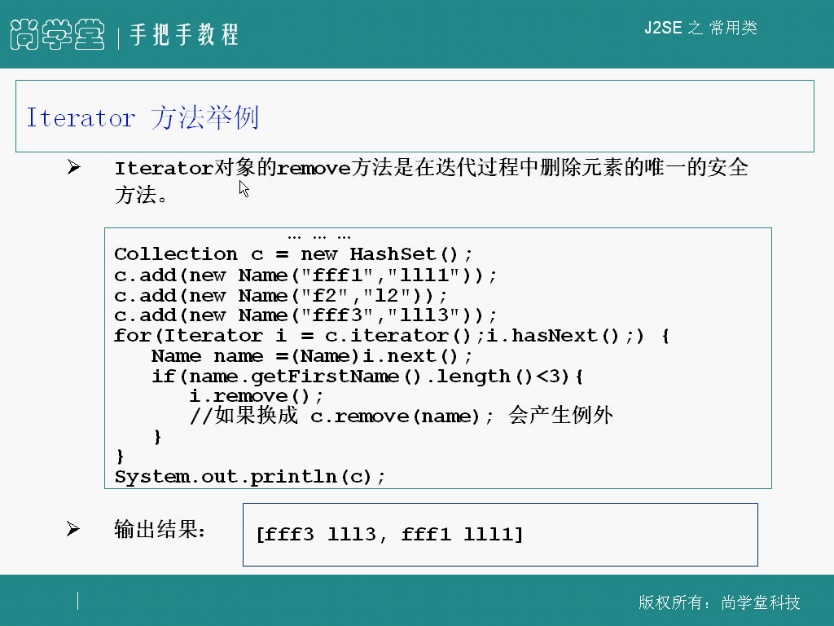
* + 多态实现：每个Collection实现类返回自己的Iterator实现
    - ArrayList返回基于数组的迭代器
    - LinkedList返回基于链表的迭代器
  + 接口最小化原则：使用者只需知道Iterator接口方法，无需关心具体实现
* Iterator的图形化理解
  + 游标模型：可以视为指向集合元素的指针
  + 遍历过程：
    - 初始位置：第一个元素左侧
    - hasNext()检查右侧是否有元素
    - next()返回元素并右移游标
    - 重复2-3直到遍历完成
* Iterator接口中的多态 ﻿
  + 多态三要素：
    - 继承/实现关系（Iterator接口与实现类）
    - 方法重写（实现类重写接口方法）
    - 父类引用指向子类对象（Iterator引用指向具体迭代器）
  + 临时空间：iterator()方法在栈空间创建临时变量存放迭代器
* Iterator的next方法返回值
  + 返回类型：JDK1.4前固定返回Object类型
  + 类型转换：使用时需要强制转换为实际类型
  + 设计原因：迭代器设计时无法预知集合元素类型

**3）Iterator方法举例 ﻿**

* Iterator的基本使用 ﻿



* + 标准遍历模式：
  + HashSet遍历：元素顺序与插入顺序无关（示例输出f2 f1 f3）
* Iterator的remove方法 ﻿



* + 安全删除：遍历时只能使用Iterator的remove()方法
  + 锁定机制：迭代过程中集合被锁定，防止并发修改
  + 错误示范：直接调用集合的remove()会导致ConcurrentModificationException
  + 使用限制：每次next()后只能调用一次remove()

**二、知识小结**

|  |  |  |  |
| --- | --- | --- | --- |
| 知识点 | 核心内容 | 考试重点/易混淆点 | 难度系数 |
| 容器概念 | 可添加其他对象的存储结构（如数组/篮子/桶的比喻） | 容器与数组的区别  • 容器动态扩容 vs 数组固定长度 | ⭐⭐ |
| 容器分类体系 | JDK容器分类：  • Set（无序不重复）  • List（有序可重复）  • Map（键值对存储） | Set与List核心差异  • 元素重复性  • 存储顺序性 | ⭐⭐⭐ |
| Collection接口 | 定义容器基础操作：  • add()/remove()  • 所有非Map容器都实现该接口 | 接口继承关系  • 子接口自动继承父接口方法 | ⭐⭐⭐⭐ |
| equals与hashCode | 对象比较规则：  • equals返回true则hashCode必须相同  • Map键值比对时优先用hashCode | 重写规范  • 必须同时重写两个方法 | ⭐⭐⭐⭐⭐ |
| Iterator迭代器 | 统一遍历容器的设计模式：  • hasNext()  • next()  • remove() | 多态实现  • 父接口引用指向子类对象  线程安全  • 遍历时只能调用迭代器的remove() | ⭐⭐⭐⭐ |
| 具体实现类 | • ArrayList（数组实现）  • LinkedList（链表实现）  • TreeMap（红黑树实现） | 底层结构差异  • 随机访问效率对比  • 内存占用差异 | ⭐⭐⭐ |
| 泛型历史 | JDK1.4返回Object类型  JDK1.5+引入泛型 | 类型强制转换风险  • ClassCastException处理 | ⭐⭐ |