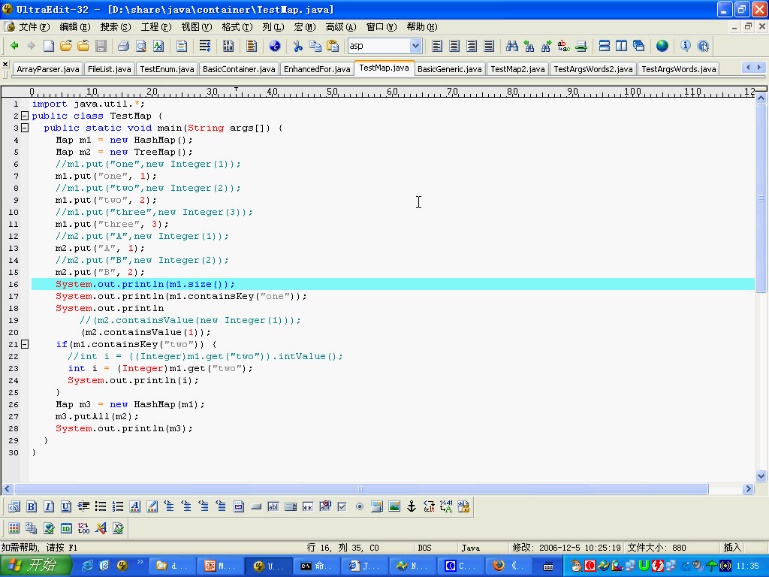
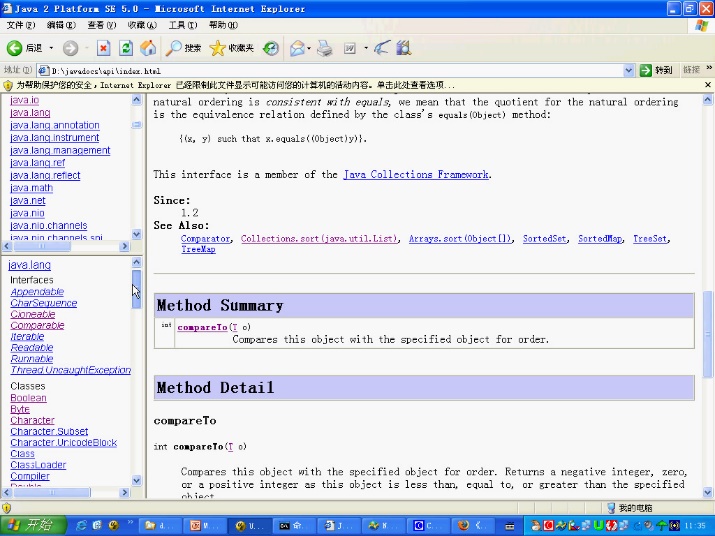
**14\_泛型\_2\_笔记**

**一、泛型编程 ﻿**

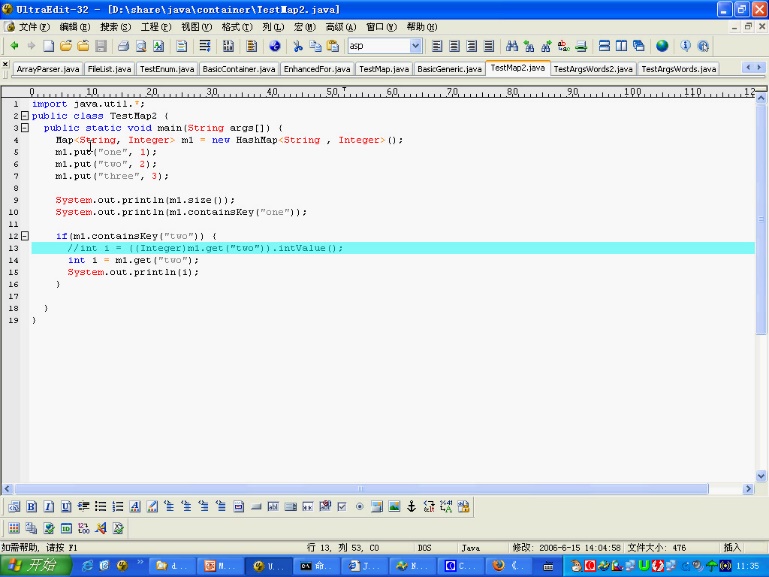


* 泛型Map声明：使用尖括号指定键值类型，如Map<String, Integer> m1 = new HashMap<String, Integer>()
* 自动打包解包：可以直接放入基础类型值（如m1.put("one", 1)），编译器会自动将1打包为Integer对象
* 类型安全获取：使用泛型后可直接获取值（如int i = m1.get("two")），无需强制类型转换
* Map常用方法：
  + size()：获取键值对数量
  + containsKey()：检查键是否存在
  + containsValue()：检查值是否存在
  + putAll()：合并两个Map



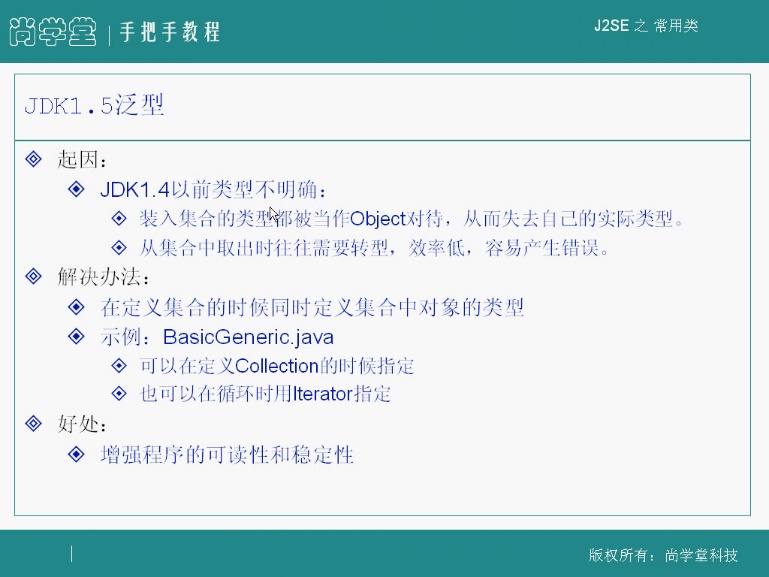
* API文档查询：通过查阅Map<K,V>接口文档可确认泛型参数数量，K代表key类型，V代表value类型
* 运行机制：泛型+自动打包解包组合使用时，编译器会先验证类型匹配，再自动完成对象与基础类型的转换

**二、命令行参数单词计数**



* 泛型声明：Map<String, Integer> m = new HashMap<String, Integer>()
* 自动打包应用：可直接put基础类型值（如m.put(x, 1)），无需手动创建Integer对象
* 计数优化：使用m.get(x)+1直接运算，自动完成解包→计算→打包过程
* 程序优势：
  + 可读性增强：明确知道Map中存储的键值类型
  + 稳定性提高：避免强制类型转换可能引发的ClassCastException

**三、泛型的总结**

**﻿**

* 使用建议：凡是用到集合时尽量使用泛型
* JDK1.4问题：
  + 类型不明确，所有元素被视为Object
  + 取出时需要强制转型，效率低且易出错
* 解决方法：
  + 定义集合时指定元素类型（如Collection<String>）
  + 使用迭代器时也可指定类型（如Iterator<Integer>）
* 双重指定：建议在集合声明和迭代时都指定泛型类型
* 三大优势：
  + 增强程序可读性
  + 提高类型安全性
  + 消除强制类型转换错误

**四、知识小结**

|  |  |  |  |
| --- | --- | --- | --- |
| 知识点 | 核心内容 | 考试重点/易混淆点 | 难度系数 |
| 泛型与自动打包解包的综合使用 | 通过Map<String, Integer>示例演示泛型定义、自动打包（如m1.put("one", 1)）及解包（如int i = m1.get("two")）的联合应用 | 泛型类型声明（<>内需写类型而非对象）、自动打包/解包与强制转换的区别 | ⭐⭐⭐ |
| 泛型的优势 | 提升程序可读性（明确集合元素类型）、稳定性（编译器类型检查避免强制转换错误） | 与非泛型代码对比（如原需(Integer)m1.get("two")强制转换） | ⭐⭐ |
| API文档的重要性 | 通过Map<K,V>接口说明泛型类型参数的来源（如K为key类型，V为value类型） | 如何查阅API文档确定泛型参数 | ⭐ |
| 命令行参数统计优化 | TestXWordsTwo示例中利用泛型Map<String,Integer>简化词频统计逻辑（自动打包put(x, 1)） | 泛型在集合遍历中的应用（如for循环中无需类型判断） | ⭐⭐⭐ |
| 泛型使用建议 | 定义集合时声明类型（如Collection<T>）、循环时用for-each进一步明确类型 | 双重类型声明的最佳实践 | ⭐⭐ |

该部分主要讲述了泛型和自动打包解包在Java编程中的应用。首先介绍了如何使用泛型来指定Map中key和value的类型，并演示了如何使用自动打包和解包来将基本类型自动转换为Integer对象，并从Map中获取值时自动解包为基本类型。

此外，还介绍了如何使用泛型和自动打包解包来简化强制类型转换的代码。最后，通过一个小程序示例演示了如何使用Map来统计命令行参数中单词的数量。

分段总结

泛型和自动打包解包的综合使用

1.泛型允许在定义集合时指定集合中对象的类型，增强程序的可读性和稳定性。

2.自动打包解包功能可以自动将对象打包为泛型类型，并自动解包为基本类型。

3.使用泛型和自动打包解包可以简化代码，减少强制类型转换的错误。

泛型在Map中的使用

1.Map接口可以指定key和value的类型，例如Map<String, Integer>。

2.使用泛型可以确保map中key和value的正确类型，避免强制类型转换的错误。

3.自动打包解包功能可以自动将对象打包为泛型类型，并自动解包为基本类型。

泛型在命令行参数处理中的应用

1.使用泛型可以方便地处理命令行参数，确保参数的正确类型。

2.自动打包解包功能可以自动将对象打包为泛型类型，并自动解包为基本类型。

3.使用泛型可以增强程序的可读性和稳定性。

泛型的总结

1.泛型定义了集合中对象的类型，增强程序的可读性和稳定性。

2.在定义集合和循环时指定泛型类型是最好的做法。

3.尽量使用泛型来定义集合，以提高程序的可读性和稳定性。