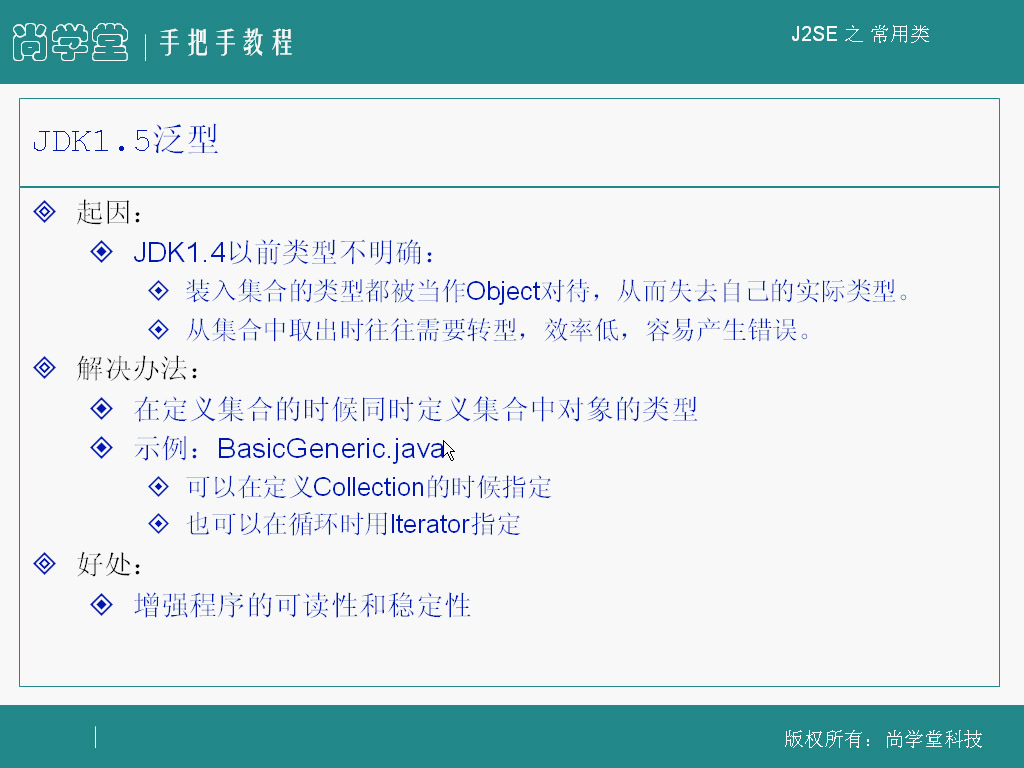
**13\_泛型\_1\_笔记**

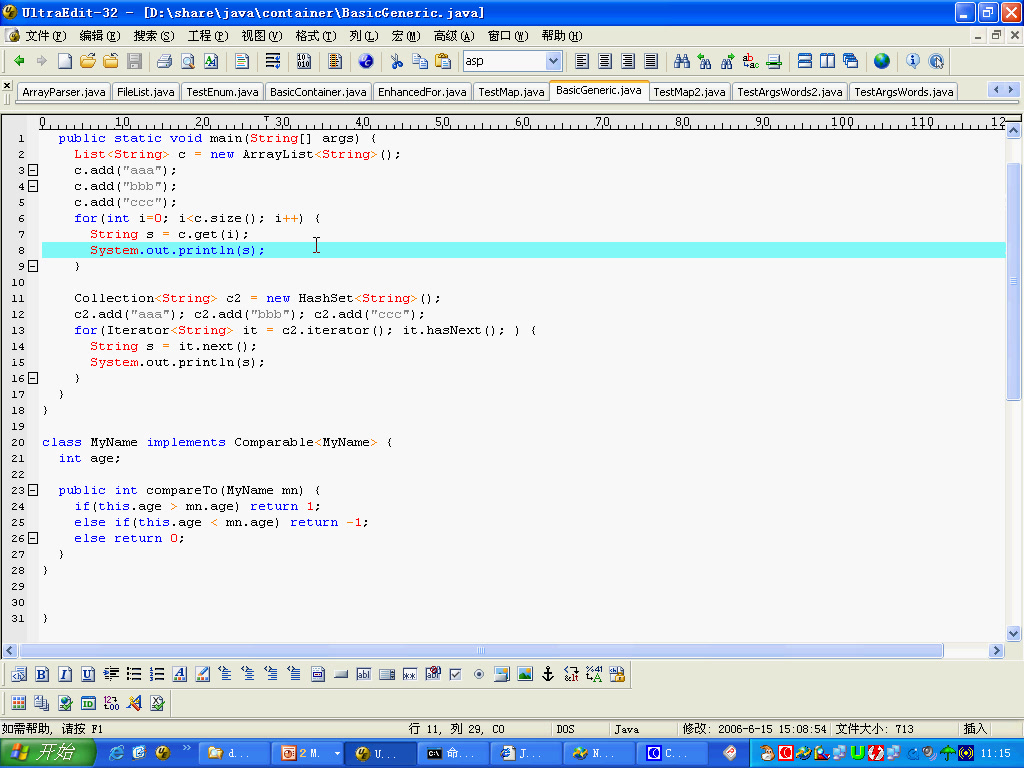
**一、泛型**

**1. 泛型的起因**



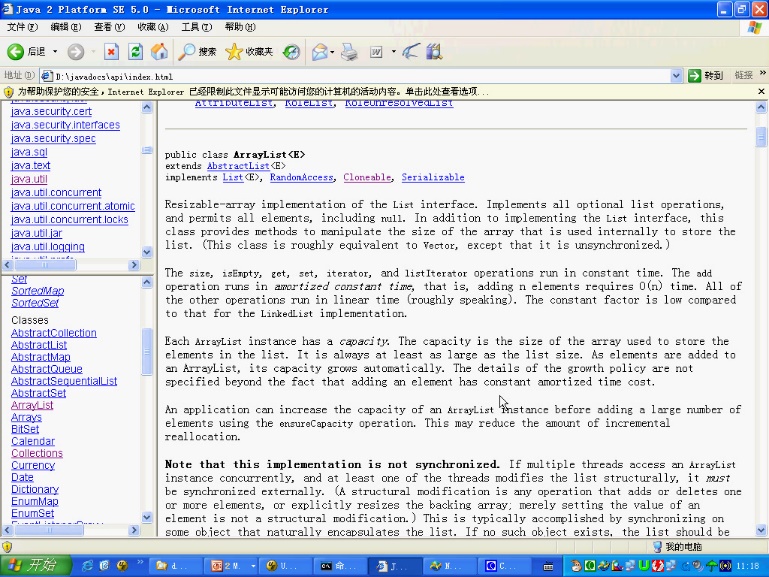
* 类型丢失问题：JDK1.4之前集合中的元素都被当作Object处理，导致实际类型信息丢失
* 强制转换缺陷：从集合取出元素时需要强制类型转换，效率低且容易在运行时出错
* 解决方案：在定义集合时同时指定元素类型，如List<String>
* 使用场景：
  + 可在定义Collection时指定类型
  + 也可在循环时用Iterator指定类型
* 优势：
  + 增强程序可读性（明确知道集合中存储的类型）
  + 提高稳定性（编译时即可发现类型错误）

**2. 例题：使用泛型 ﻿**



* 基本语法：
* 类型限制：指定String后只能添加String类型元素，尝试添加其他类型会编译报错
* 遍历优势：
  + 传统方式需要强制转换：String s = (String)c.get(i)
  + 泛型方式直接获取：String s = c.get(i)
* 集合实现：
* 迭代器使用：

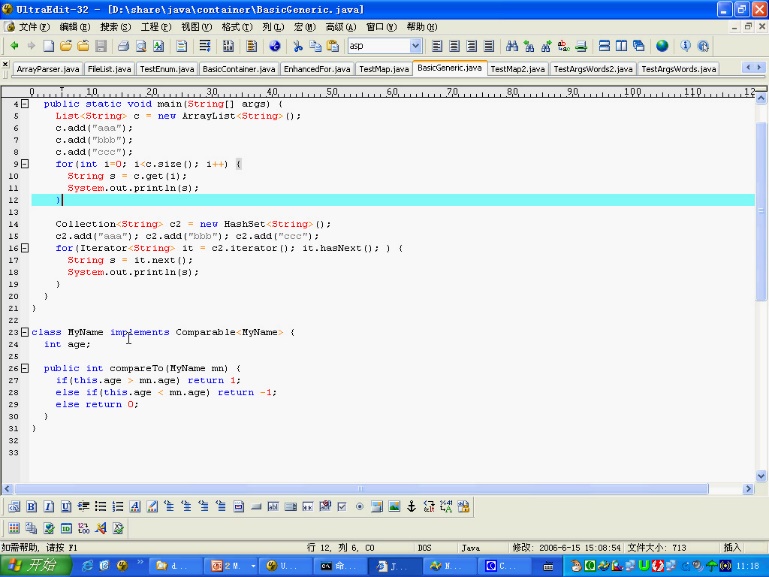
**3. ArrayList接口**



* 泛型类识别：类名后带尖括号<E>表示支持泛型
* 返回值类型：方法返回值中的E会自动替换为实际指定的类型
* API文档解读：
  + add(E o)：添加的元素类型与声明时一致
  + get(int index)：返回类型即为声明的泛型类型
* 类型一致性：整个类的泛型参数E会贯穿所有相关方法

**4. Comparable接口**

**﻿**



* 泛型接口实现：
* 类型安全比较：
  + 传统方式参数为Object，需要强制转换
  + 泛型方式直接限定为MyName类型
* 实现示例：
* 编译时检查：确保只能比较相同类型的对象，避免运行时错误

**二、知识小结**

|  |  |  |  |
| --- | --- | --- | --- |
| 知识点 | 核心内容 | 考试重点/易混淆点 | 难度系数 |
| 泛型概念 | 通过尖括号指定容器存储类型（如List<String>），避免强制类型转换和运行时错误 | 泛型擦除机制（JAVA与C++实现差异） | ⭐⭐ |
| 泛型优势 | 1. 类型安全（编译时检查）  2. 消除强制转换（如List<String>取出直接为String）  3. 提升代码可读性 | 运行时类型擦除导致无法直接获取泛型具体类型 | ⭐⭐ |
| 泛型应用场景 | 1. 集合框架（ArrayList<String>）  2. 接口实现（Comparable<MyClass>） | 自定义泛型类需在类声明后添加尖括号（如Class<T>） | ⭐⭐⭐ |
| 历史背景 | JDK1.5引入泛型，解决JDK1.4前容器存储Object导致的类型丢失问题 | 类型擦除与C++模板机制差异 | ⭐ |
| 泛型与集合 | 示例：Collection<String> c2 = new HashSet<>()，添加非String类型会编译报错 | 泛型通配符（<?>）未在文中展开 | ⭐⭐ |
| Comparable接口泛型化 | class MyName implements Comparable<MyName>，避免compareTo方法中强制转换 | 类型边界（如<T extends Comparable>）未提及 | ⭐⭐⭐ |
| 实践任务 | 改造testMap使用泛型，消除Integer强制转换 | 自动装箱/拆箱与泛型结合易忽略 | ⭐⭐ |

该部分主要讲述了Java中泛型的使用。泛型是一种编程技术，通过在类、接口和方法中使用类型参数，使得代码更加灵活和可重用。通过例子演示了泛型的使用方法和好处，包括避免类型转换错误和运行时异常，提高程序的稳定性和可维护性。同时，也解释了泛型底层的实现原理和为什么需要泛型这个概念。还介绍了泛型的一些限制和注意事项，例如类型参数的个数和类型参数的取值范围等。总之，该详细介绍了泛型的基本概念和在Java中的实现，对于初学者来说非常有帮助。

分段总结

泛型概述

1.泛型听起来很复杂，但实际上使用起来非常简单。

2.泛型的底层实现较难，尤其是JAVA的泛型实现与C++不同。

3.JDK 1.5引入泛型概念，旨在解决1.4版本类型不明确的问题。

泛型的好处

1.避免运行时错误，通过编译期间的类型检查提高程序稳定性。

2.解决类型转换的麻烦，增强代码的可读性和可维护性。

泛型的基本用法

1.使用尖括号指定泛型类型，如List<String>。

2.泛型类型参数可以用于类、接口和方法的定义中。

3.编译器会自动判断泛型类型的正确性，减少错误。

泛型类的定义和使用

1.泛型类定义时在类名后加上尖括号，如class MyName<T>。

2.泛型类型参数T可以在类的方法和属性中使用。

3.创建泛型类对象时，需要指定泛型类型参数的具体类型。

泛型接口的定义和使用

1.泛型接口定义时在接口名后加上尖括号，如interface Comparable<T>。

2.泛型类型参数T可以在接口的方法中使用。

3.实现泛型接口的类需要指定泛型类型参数的具体类型。

泛型在Map中的应用

1.将Map定义为泛型类型，如Map<String, Integer>。

2.避免运行时类型转换的错误，增强代码的安全性。

3.自动打包和解包功能简化了泛型的使用。