****

Java程序设计

课程设计报告

姓名： 姚文琪

学号： 58123327

东南大学计算机科学与工程学院

二0 五 年 五 月

# **课程设计报告**

## **一、代码架构**

本次课程设计采用 ****MVC（Model-View-Controller）架构****，将程序分为三层：

****1.模型层（Model）****：由 Account 和 AccountManager 类组成，负责存储和管理账户数据。

****2.控制层（Controller）****：由 ATMController 类组成，负责处理业务逻辑，如存取款、登录等操作。

****3.视图层（View）****：由 UserInterface 类组成，负责与用户交互，显示菜单和处理用户输入。

项目结构说明：

项目根目录

├── src

│ └── atm1

│ ├── Account.java // 用户账户模型

│ ├── AccountManager.java // 账户管理模型

│ ├── ATMController.java // ATM系统控制器

│ ├── UserInterface.java // 用户界面视图

│ ├── FileManager.java // 文件操作辅助类

│ └── ATMMain.java // 程序入口

└── accounts.txt // 用户账户数据文件（运行时自动生成）

## **二、利息计算方法**

* 利息计算逻辑
* 利息计算采用 简单利息公式：
* 利息=本金×利率×时间
* 具体参数和步骤如下：
* 利率：固定为 5%（interestRate = 0.05）。
* 时间单位：以 10 秒为单位。
* 利息计算周期：每 10 秒计算一次利息。
* 计算逻辑：
* 获取当前时间 currentTime 和账户的 lastInterestUpdate（记录最后一次利息计算时间）。
* 计算时间差 elapsedTime = currentTime - account.getLastInterestUpdate()。
* 如果 elapsedTime 大于等于 10 秒，则进行利息计算。
* 利息计算公式：interest = principal \* interestRate \* time，其中：
* principal 是账户的本金（account.getBalance()）。
* time = elapsedTime / 10000.0，将时间差转换为 10 秒单位。
* 将计算得到的利息累加到 accumulatedInterest 属性中。
* 更新 lastInterestUpdate 为当前时间。
* 利息与本金分离
* 利息和本金是分开存储的。为了确保利息和本金的独立性，我们在 Account 类中增加了 getTotalBalance() 方法，用于在查询余额时返回本金和累计利息的总和：
* public double getTotalBalance() {
* return balance + accumulatedInterest;
* }

## **三、遇到的问题及解决方法**

### **1. 利息计算问题**

****问题描述****：  
最初没有注意到“利滚利”的问题，导致利息计算逻辑有误。错误地将利息直接加到本金中，从而影响后续利息的计算。

****解决方法****：  
将利息和本金分开存储。增加一个新的方法 getTotalBalance()，用于在查询余额时返回本金和累计利息的总和：

public double getTotalBalance() {

return balance + accumulatedInterest;

}

### **2. 文件存储与读取问题**

****问题描述****：  
最初没有考虑到文件存储的安全性和完整性，可能导致数据丢失或损坏。程序需要在启动时加载之前保存的账户信息，并在退出时保存当前的账户信息。如果文件操作不当，可能会导致数据无法正确保存或读取。

****解决方法****：  
使用 ObjectOutputStream 和 ObjectInputStream 进行序列化和反序列化操作，确保数据能够完整地保存和读取。具体实现如下：

****文件保存****：

在 FileManager 类中，定义了一个 saveAccounts() 方法，使用 ObjectOutputStream 将账户信息写入文件。

将 AccountManager 中的所有账户信息序列化后保存到文件 accounts.txt 中。

****文件读取****：

在 FileManager 类中，定义了一个 loadAccounts() 方法，使用 ObjectInputStream 从文件中读取账户信息。

从文件 accounts.txt 中读取序列化的账户信息，并将其还原为 AccountManager 中的账户数据。

****异常处理****：

在文件操作中加入了异常处理机制，确保程序在遇到文件读写错误时能够正常运行并给出错误提示。

通过以上方法，确保了程序在启动和退出时能够正确地保存和读取账户信息，避免了数据丢失或损坏的问题。

## **四、总结**

通过本次课程设计，我深入理解了 ****MVC 架构**** 的设计思想，并掌握了如何将其实现到实际项目中。MVC 架构的使用使得程序的结构更加清晰，各个模块之间的职责更加明确。模型层负责数据管理，控制层负责业务逻辑，视图层负责用户交互，这种分离设计使得代码的可维护性和可扩展性得到了显著提升。

通过这次课程设计，我对 Java 语言有了更深刻的理解。Java 的面向对象特性使得代码的组织和管理变得更加高效，而其强大的标准库则为文件操作、数据序列化等任务提供了极大的便利。同时，我也认识到了编程中细节的重要性，一个小小的逻辑错误可能会导致整个程序的失败，因此在今后的编程实践中，我会更加注重代码的逻辑严谨性和细节处理。

这次课程设计不仅让我在技术上有所收获，也让我在解决问题的能力上得到了锻炼。我学会了如何分析问题、查找原因并设计解决方案。我期待在今后的学习和工作中能够运用所学知识，解决更多复杂的问题，开发出更加优秀和可靠的软件系统。

对于课程的一点小建议：

我觉得课上其实不用给太多的时间给大家照抄一些代码自己跑一遍，老师演示一遍我们在底下看一下理解一下就够啦。

另外还想多听一些关于常用容器或数据结构在Java中如何实现的，和C++有什么不同。