**西安电子科技大学**

**Java程序设计 课程实验报告**

**实验名称 银行账户系统异常处理**

计算机科学与技术 学院 2103051 班

成 绩

姓名 张平 学号 21030540006

同作者

实验日期 2022 年 05 月 09 日

|  |
| --- |
| 指导教师评语：  指导教师：  年 月 日 |
| **实验报告内容基本要求及参考格式**  一、实验目的  二、实验内容  三、实验过程  四、实验结果分析  五、实验小结（实验过程感受和建议） |

# 实验目的

1. 掌握面向对象的类、继承、多态概念，能运用Java完成基于面向对象的指定功能的程序制作。
2. 在第四章实验的基础上，熟悉异常的定义、异常的类型层次，以及异常处理的动机。
3. 掌握异常处理的两种方法：捕获并处理异常；将异常抛出。
4. 熟悉如何定义并使用自己的异常类。了解断言的作用。

# 实验内容

为方便阅览，这里列出第四章实验3的实验要求如下：

Design **BankSystem** class. You need to design more than one class. For example:

1. You need to design a class BankAccount to model users’ bank accounts. Probably different bank accounts (CashAccount, CreditAccount, …).
2. The account should keep a user’s name and balance, accurate to the nearest cent…
3. The user should be able to make deposits and withdrawals on his/her account, as well as changing the account’s name at any time.
4. Also, the system needs to be able to find out how many BankAccounts have been created in total.
5. For each account, only the last 6 Transactions should be able to store in ascending order and be printed.

这一章将对第四章实验内容3添加异常处理、完善其运行机制。要求如下：

1. Add the CheckingAccount class of BankSystem , Designing and Using Classes to throw an IllegalArgumentException in any of the following circumstances:
2. when the account is constructed with a negative balance,
3. when a negative amount is deposited, or
4. when the account is overdrawn (when the amount withdrawn exceeds the current balance).

(2) An IllegalArgumentException is an unchecked exception that is thrown to indicate that a method has been passed an illegal or inappropriate argument.

Instructions:

Add (modify if you've already finished) the CheckingAccount class to handle errors and write a test program as indicated above.

# 实验过程

1. **实验环境**

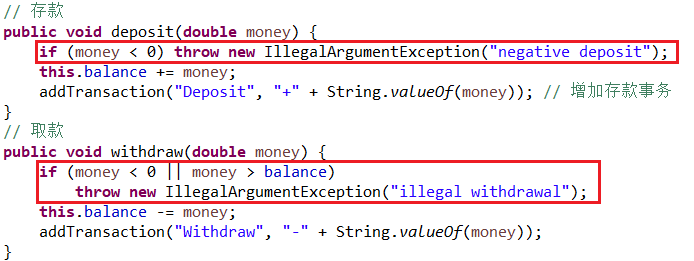
操作系统：Windows 11

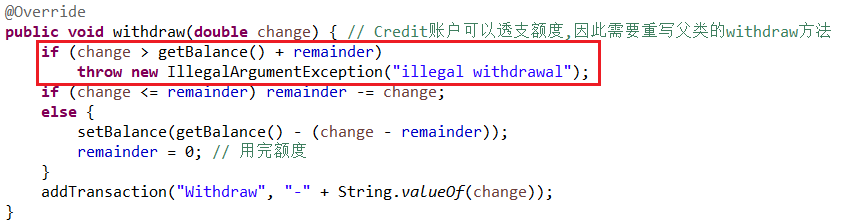
集成开发环境：Eclipse IDE for Enterprise Java and Web Developers (includes Incubating components) 2022-03 (4.23.0)

1. **题目分析**

这次实验比较简单，对其分析发现，本次实验要求我们给此前的银行账户系统添加异常处理，对这三种情况——账户对象以负数余额构造、存储负数金额进入账户、账户过度取款，要求抛出一个非法参数异常。为此，题目要求在银行系统中添加一个名为CheckingAccount的类。

而我们将在这个类中定义三个静态方法，用于分别检查这三种情况并抛出异常。之所以选择将其定义为静态方法，是因为我们不需要个性化的CheckingAccount类对象，我们只需要调用相关方法、完成相应检查操作即可。事实上，在第四章实验3中，我就已经完成了简单的异常处理，本次实验只用稍作修改：



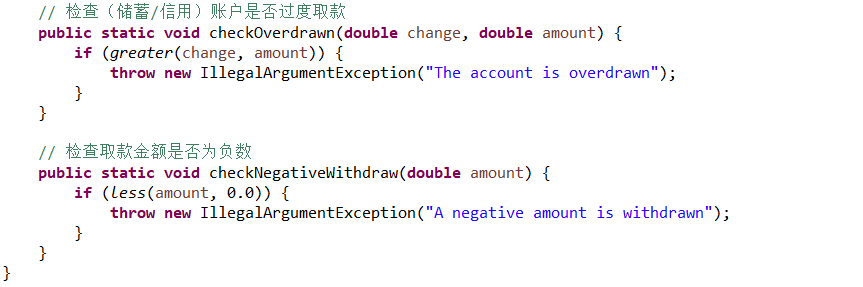


此外，这里不使用Erlang/Elixir的编程哲学Let it crash，而是在抛出异常后使用try/catch语句进行捕捉、打印错误提示、并继续执行程序。

1. **代码实现**

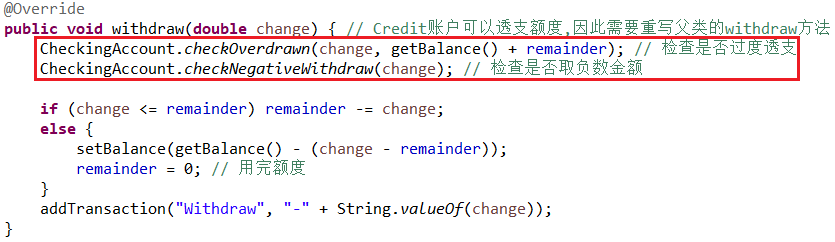
CheckingAccount类的实现如下所示，除了完成题目要求的情况外，还做了进一步完善。checkConstructor方法检查实例化账户对象时、是否以负数余额构造，checkNegativeDeposit方法检查是否存储负数金额进入账户，checkOverdrawn方法检查（储蓄/信用）账户是否过度取款，checkNegativeWithdraw方法检查取款金额是否为负数。此外，**代码中在判断浮点数是否为负数时，没有直接让浮点数与0比较**。

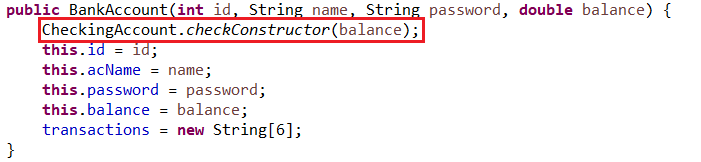


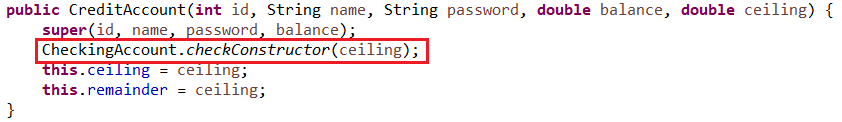


原先的检查语句，现在换成对CheckingAccount相应静态方法的调用，如果这些静态方法正常完成，则对应的存款、取款、初始化等操作继续执行；否则停止执行、并继续向上层调用抛出异常：

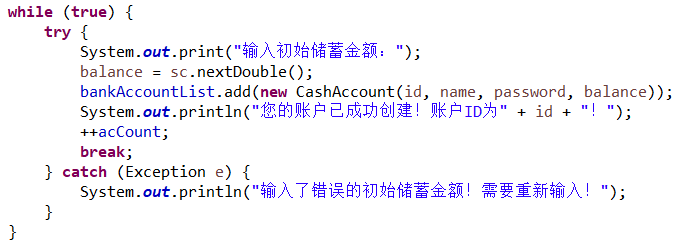


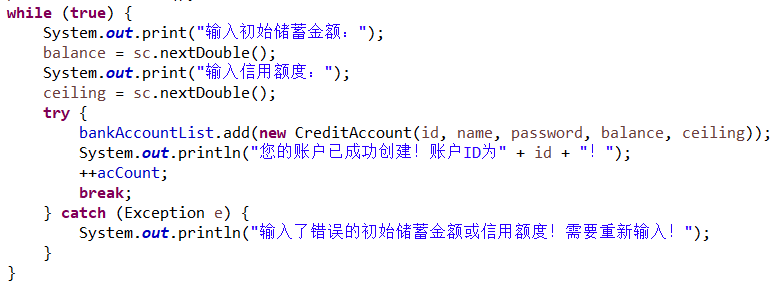


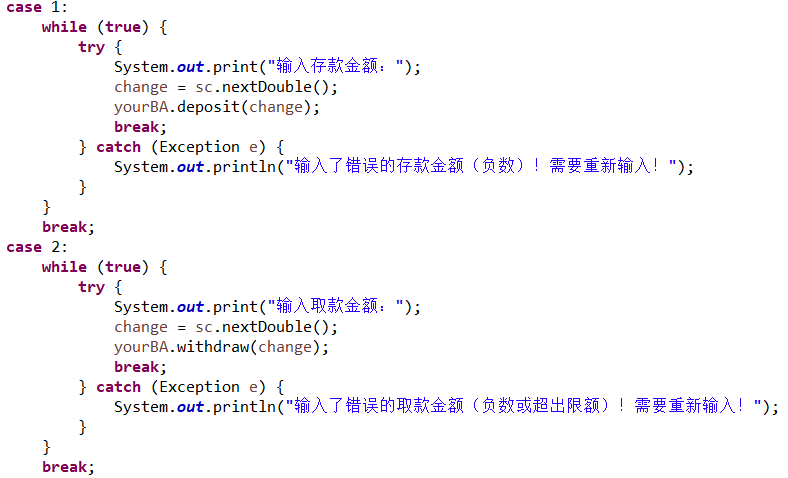




遵循题目要求，我们在主类BankSystem的openAccount方法（即创建储蓄账户和信用账户的方法）和login方法（即输入账户姓名和密码来登录账户，进行各种实际操作的方法）中，修改相关的代码。如果对账户对象的操作抛出异常，则用try/catch语句进行捕捉，并继续循环、直到输入正确数据（**不打印异常抛出的信息，将其与用户隔离开**）；若无异常，则break退出循环。部分代码如下所示：

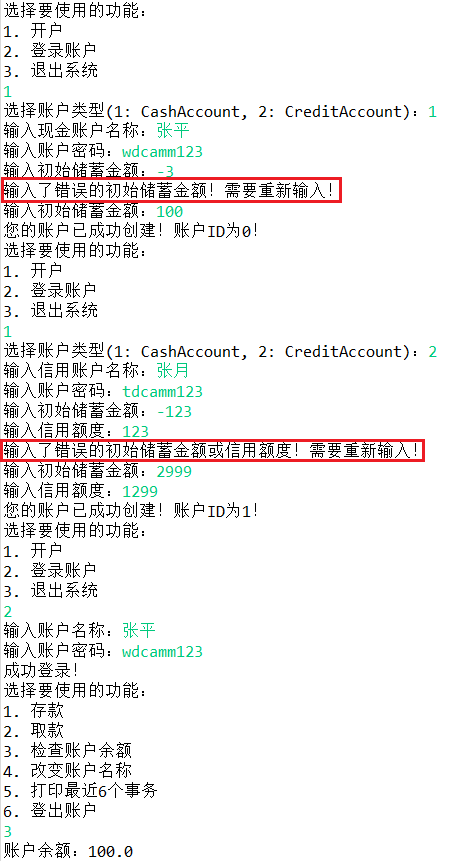


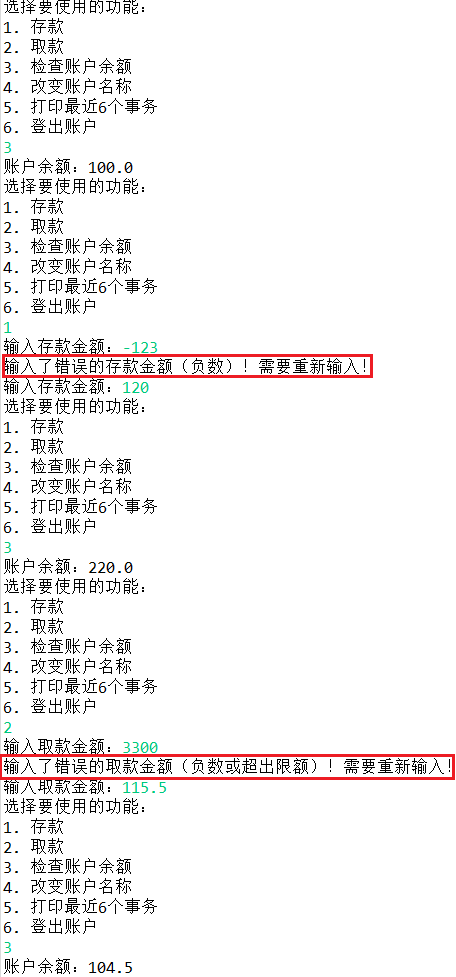


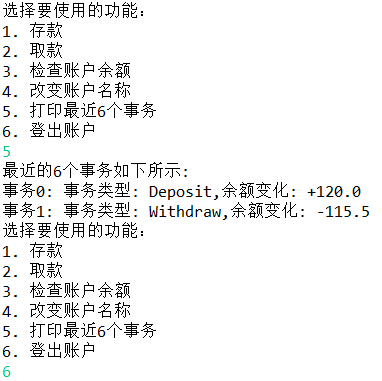


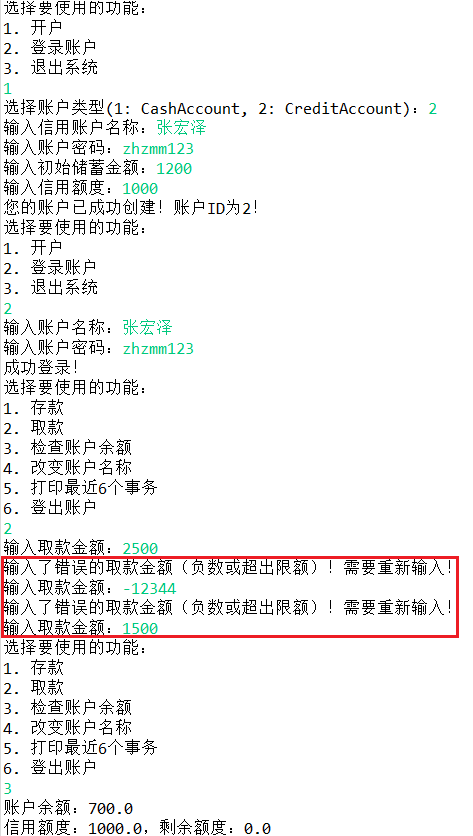
# 实验结果测试

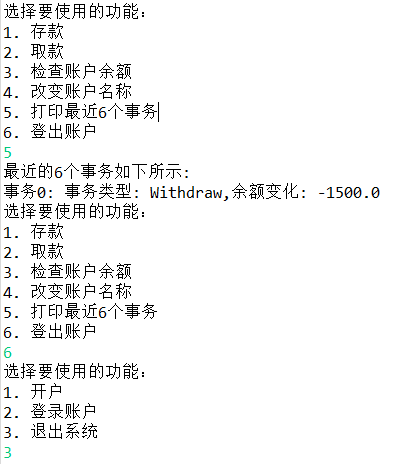
在Eclipse中对实现的功能进行测试，结果如下：











如上所示，实验结果正确。

# 实验小结

本次实验中，实现了银行账户系统的异常处理，包括抛出异常、捕获异常并处理。对异常的理解和使用，是了解现代编程思想十分重要的一环。最近我在看《深入理解计算机系统》一书，书中的第八章异常控制流，则站在更加宽泛、更加底层的角度剖析异常处理机制，使我收获颇多，只是在编程时还需多加磨炼。