成绩	

重庆邮电大学

实验报告

2020-2021 学年第 2 学期

计算机科学导论

（第 7 次试验）

班级：34082003

姓名：黄凯升

学号：2020215138

指导老师：许汀汀

课程名称：计算机科学导论

实验时间：2021 年 5 月 20 日

实验地点：综合实验大楼 A511/A512

1 实验名称

Inheritance

2 实验目的

- Be able to derive a class from an existing class
- Be able to define a class hierarchy in which methods are overridden and fields are hidden
- Be able to use derived-class objects
- Implement a copy constructor

3 实验内容

Task#1 Extending BankAccount

Create a subclass called CheckingAccount which extends the super class BankAccount, representing an account for making checks. It should contain a static constant `FEE` to be set to $0.15, which is the extra fee for each transaction. And it should also have an overridden method `withdraw`. The overriden method `withdraw` should add the fee to the amount of number to be withdrawal. In particular, the account number of CheckingAccount should be the original account number concentrated with "-10".

Task #2 Creating a Second Subclass

Create another subclass called SavingsAccount, which also extends the super class BankAccount, representing an account for saving money. It should contain a static constant `rate` to be set to 2.5%, which is the annual interest rate. And it should also have a method called `postInterest` which add monthly interest to the account. And it should have a field called `savingsNumber` that tells different savings account under one account and a new `accountNumber` that hides the field with same name in super class. A copy method is needed to create new savings account from one savings account. And also the method `getAccountNumber` should be overridden.

4 实验方法(原理、流程图)

The development environment is:

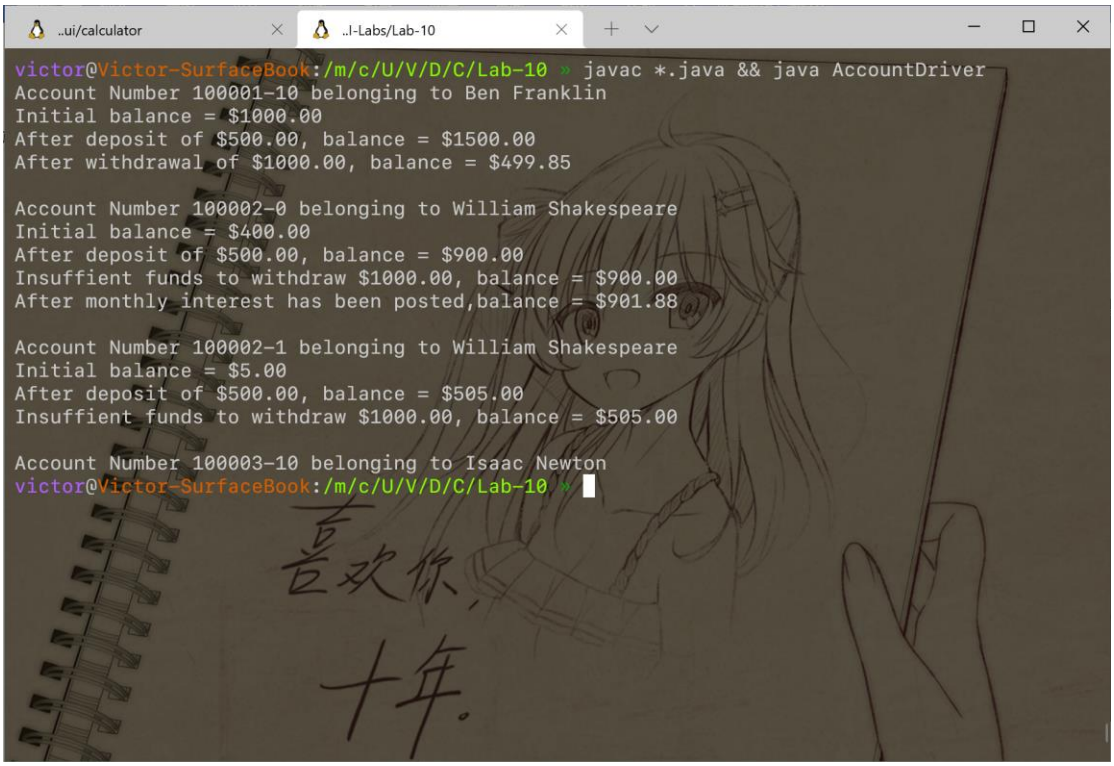
- OS: Ubuntu 20.04.2 LTS on Windows 10 (WSL1, Kernel build 19041)
- IDE/Editor: Visual Studio Code
- Java Runtime: OpenJDK 14.0.2 (build 14.0.2+12-Ubuntu-120.04)

For Task #1, we should declare the class CheckingAccount with keyword `extends`. And just add a constant field `FEE` and set it to 0.15. In the constructor, we should call a special callable pointer `super` first, which points to the super class, and when we call it we actually call the constructor of super class. Similarly, `this` points to this class. Note that `super` or `this` should be called in the beginning of constructor. Then we concentrate the account number of superclass and "-10" and set it via mutator methods. In the end we make an overridden method `withdraw`. Just call the `super` in the superclass with the amount added by fee.

For Task #2, we should also declare an extended class SavingsAccount. We should add a constant field `rate`, set it to 2.5/100, and add a integer `savingsNumber` initialized to 0. To hide the field `accountNumber` in the super class, we should add a field in the same name and the same type. The standard constructor is similar to Task #1. In the copy constructor, we should first call the `super` constructor, then increase `savingsNumber` by 1 and set `accountNumber`. Note that we should also override the accessor method `getAccountNumber`. To implement the method `postInterest`, we just call `deposit` method to deposit the interest. Because the rate is annual rate, we can calculate the monthly interest by this formula: $\text{MonthlyInterest} = \text{Balance} * \text{rate} \div 12$.

5 实验结论

The lab has finished successfully. The programs can completely achieve all goals. Here is the screenshot.



6 实验体会和收获

In this lab we practiced more of object-oriented programming. With inheritance, we can reuse most of our code. It shows the idea of abstraction – to find out the common ground between things and use united ways to handle them, elegantly.

7 程序代码

CheckingAccounts.java:

```
public class CheckingAccount extends BankAccount {
    private final double FEE = 0.15;
    /**
     * standard constructor
     *
     * @param name   the owner of the account
     * @param amount the beginning balance
     */
    public CheckingAccount(String name, double amount) {
        super(name, amount);
        this.setAccountNumber(super.getAccountNumber() + "-10");
    }

    /**
     * allows you to remove money from the account if enough money is
     * available,returns true if the transaction was completed, returns false if the
     * there was not enough money.
     *
     * @param amount the amount to withdraw from the account
     * @return true if there was sufficient funds to complete the transaction, false
     *         otherwise
     */
    @Override
    public boolean withdraw(double amount) {
        return super.withdraw(amount + this.FEE);
    }
}
```

SavingsAccount.java:

```
public class SavingsAccount extends BankAccount {
    private final double rate = 2.5 / 100;   // ANNUALY RATE
    private int savingsNumber = 0;
    private String accountNumber;

    /**
     * standard constructor
     *
     * @param name   the owner of the account
     * @param amount the beginning balance
     */
    public SavingsAccount(String name, double amount) {
        super(name, amount);
        this.accountNumber = String.format("%s-%d", super.getAccountNumber(), this.savingsNumber
);
    }
    /**
     * copy constructor
     *
     * @param oldAccount   the old account
     * @param amount       the beginning balance
     */
    public SavingsAccount(SavingsAccount oldAccount, double amount) {
        super(oldAccount, amount);
        this.savingsNumber = oldAccount.savingsNumber + 1;
        this.accountNumber = String.format("%s-%d", super.getAccountNumber(), this.savingsNumber
);
    }

    /**
     * post MONTHLY interest into account
     */
    public void postInterest() {
        this.deposit(this.getBalance() * this.rate / 12);
    }

    /**
     * accessor method to account number
     *
     * @return the account number
     */
    @Override
    public String getAccountNumber() {
        return this.accountNumber;
    }
}
```