# 20241220周五实验课-Polymorphism-02班

## 1. 编程题

### 简易的银行账户管理

- 定义了一个父类`Account`：
  具有以下属性：账户ID（`accountId`），账户余额（`balance`）
  具有取款方法`withdraw()`，存款方法`deposit()`，账户信息显示方法`displayAccountInfo()`
- 定义一个子类 储蓄账户类`SavingsAccount`：
  这个类继承`Account`。
  `SavingsAccount`：增加一个属性，利率（`interestRate`），并实现利息计算方法`calculateInterest()`和账户信息显示方法`displayAccountInfo()`。
  利息计算可以自行设计，可以简单使用 利息=利率*余额。
- 定义一个子类 信用账户类`CreditAccount`：
  这个类继承`Account`。
  `CreditAccount`：增加一个属性，透支限额(`overdraftLimit`)， 即活期账户进行取款时，取款金额可以超过账户余额，但是超出账户余额的部分不能超过透支额度，可以简单理解为借款额度，所以该类账户`balance`可能小于0。
  并实现检查透支状态的方法`checkOverdraft()`和账户信息显示方法`displayAccountInfo()`。
- 定义一个测试类`AccountsTest`，在`main()`方法中，通过不同引用类型创建多个账户，并演示存款、取款和显示账户余额等操作。

**要求：**

参考群里发的`ex18.java`完成：

- 为`SavingsAccount`，补全`calculateInterest()`：计算利息额；重写`displayAccountInfo()`: 要求增加打印利息信息。
- 为`CreditAccount`，补全`displayAccountInfo()`：要求增加打印透支额度；重写`withdraw()`:取款注意不要超过透支额度。
- 使用两种子类账户进行存取款和账户信息打印等测试。

**目的：**

- 通过父类和子类的不同引用方式（父类引用指向子类对象与子类引用指向子类对象），理解`Polymorphism`。
- 重写父类方法，理解`Method overriding`。
- 通过`instanceof`检查对象的类型并进行类型转换，以便调用子类特有的方法（如`calculateInterest`和`checkOverdraft`）。`instanceof`运算符确保只有在对象确实是特定类型时才进行类型转换，避免了`ClassCastException`异常。

下面一共有四份源文件代码：`Account.java` `SavingsAccount.java` `CreditAccount.java` `AccountsTest.java`

`Account.java`

```
class Account {
        protected String accountId;
        protected double balance;

        public Account(String accountId, double balance) {
                // 账户 ID 不能为空
                if (accountId == null || accountId.trim().isEmpty()) {
                        throw new IllegalArgumentException("账户ID不能为空");
                }

                // 余额必须大于等于 0
                if (balance < 0) {
                        throw new IllegalArgumentException("余额不能为负数");
                }
                this.accountId = accountId;
                this.balance = balance;
        }

        public void displayAccountInfo() {
                System.out.println("账户ID: " + accountId + ", 当前余额: " + balance);
```

```
        }

        public void deposit(double amount) {
                if (amount <= 0) {
                        System.out.println("存款金额必须大于 0.");
                } else {
                        balance += amount;
                        System.out.println("账户ID: " + accountId + " 成功存款: " + amount);
                }
        }
        public void withdraw(double amount) {
                if (balance >= amount) {
                        balance -= amount;
                        System.out.println("账户ID: " + accountId + " 成功取款: " + amount);
                } else {
                        System.out.println("取款失败，超出余额!");
                }
        }
}
```

SavingsAccount.java

```
//子类：SavingsAccount

class SavingsAccount extends Account {
        private double interestRate; // 利率

        public SavingsAccount(String accountId, double balance, double interestRate) {
                super(accountId, balance);
                this.interestRate = interestRate;
        }

        public void calculateInterest() {

        }

        @Override
        public void displayAccountInfo() {
        }
}
```

CreditAccount.java

```
class CreditAccount extends Account {
        private double overdraftLimit; //透支限额

        public CreditAccount(String accountId, double balance, double overdraftLimit) {
                super(accountId, balance);
                this.overdraftLimit = overdraftLimit;
        }

        // TO DO，重写取款方法，注意补全参数，考虑透支限额
        @Override
        public void withdraw() {
        }

        // 检查透支状态的方法
        public void checkOverdraft() {
        }

        @Override
        public void displayAccountInfo() {

        }

}
```

AccountsTest.java

```

public class AccountsTest {
```

```
        public static void main(String[] args) {
        System.out.println("=====测试:父类引用指向子类对象====");

        Account account1 = new SavingsAccount("S001", 1000, 0.05);


        // 存款并显示余额（SavingsAccount）
        account1.deposit(200);
        account1.displayAccountInfo();
    //account1.calculateInterest(); // error

        // 检查储蓄账户（SavingsAccount）的利息
        if (account1 instanceof SavingsAccount) {
                ((SavingsAccount) account1).calculateInterest();
        }

        System.out.println("=====测试:子类引用指向子类对象====");
        SavingsAccount savingsAccount2 = new SavingsAccount("S002", 1000, 0.05);

        savingsAccount2.deposit(200); // 存款
        savingsAccount2.displayAccountInfo(); // 显示账户信息
        // 计算利息
        savingsAccount2.calculateInterest();

        }
}
```