**实 验 报 告 (三)**

|  |  |
| --- | --- |
| **课程名称：** | **Java语言程序设计** |
| **学生姓名：** |  |
| **学生学号：** |  |
| **学生班级：** |  |
| **开课学期：** | **2018-2019学年第一学期** |

**软件学院**

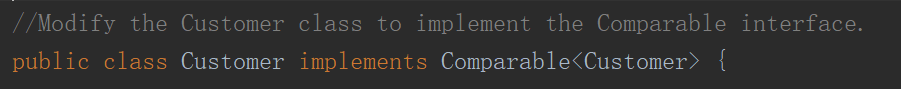
**2018年10月**

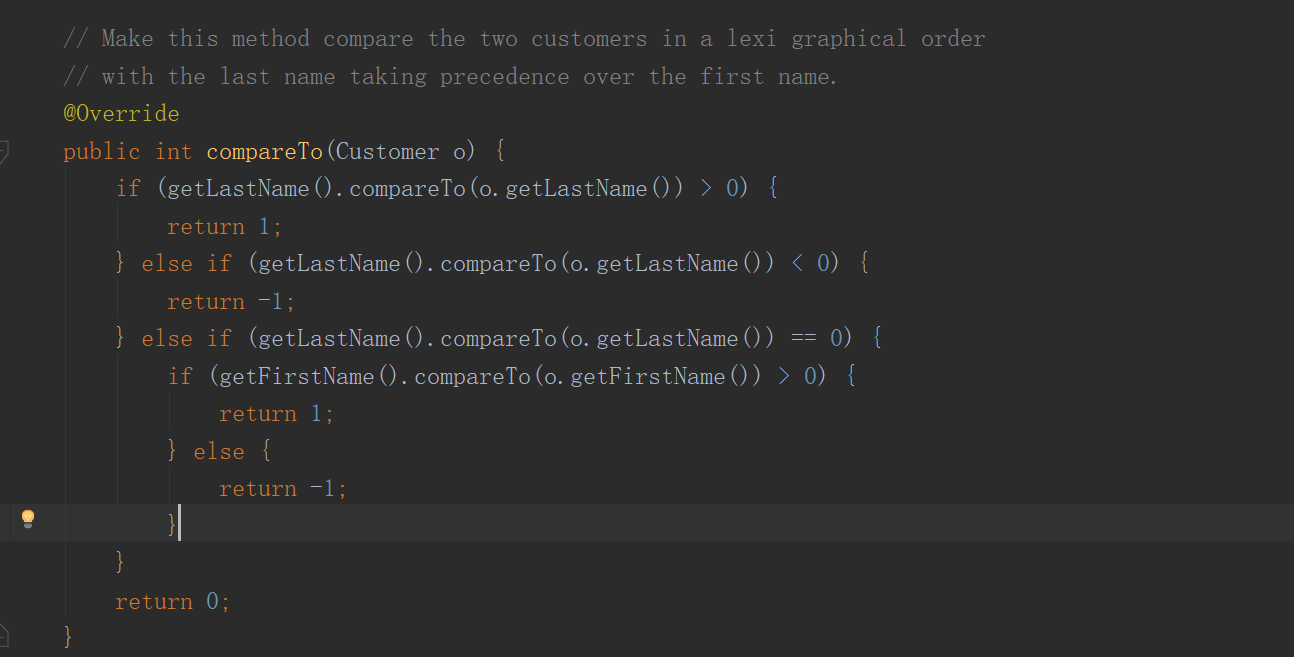
# 实验三 接口实现与异常处理

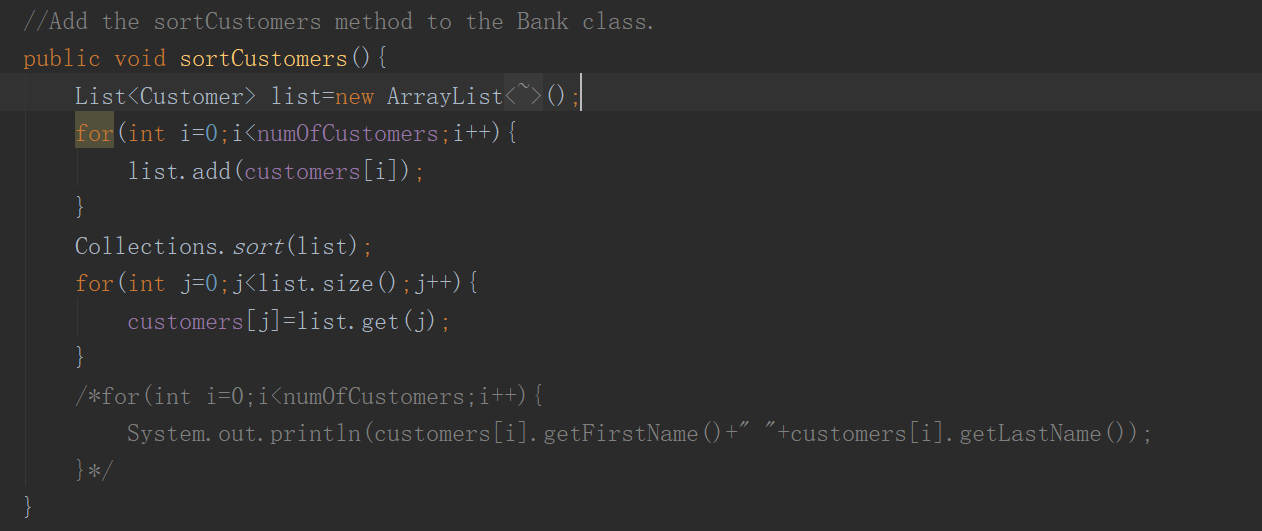
## 一、代码部分

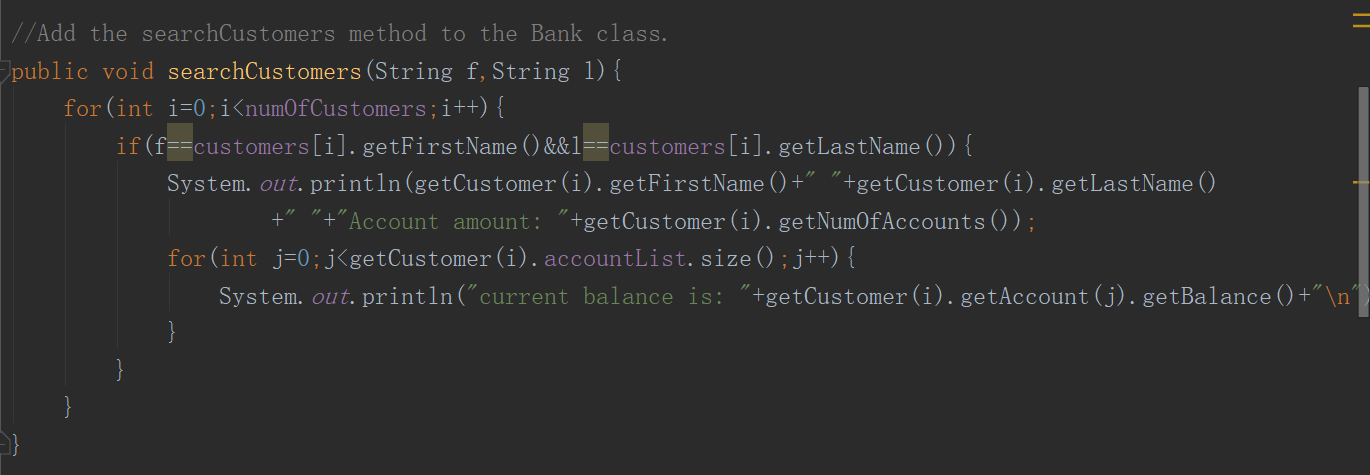
（注：每个步骤中新建或修改的类的源代码，其中修改的每个部分添加注释，如 /\*\* step6 ：功能描述 \*\*/ ）

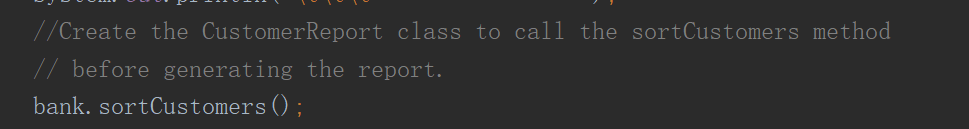
/\*\* step6 ：功能描述 \*\*/

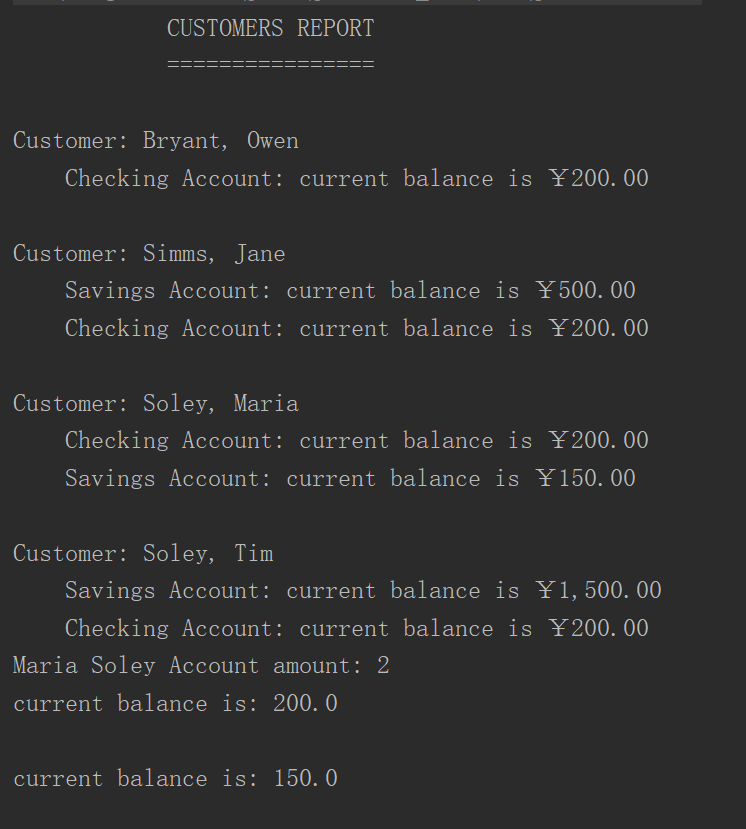




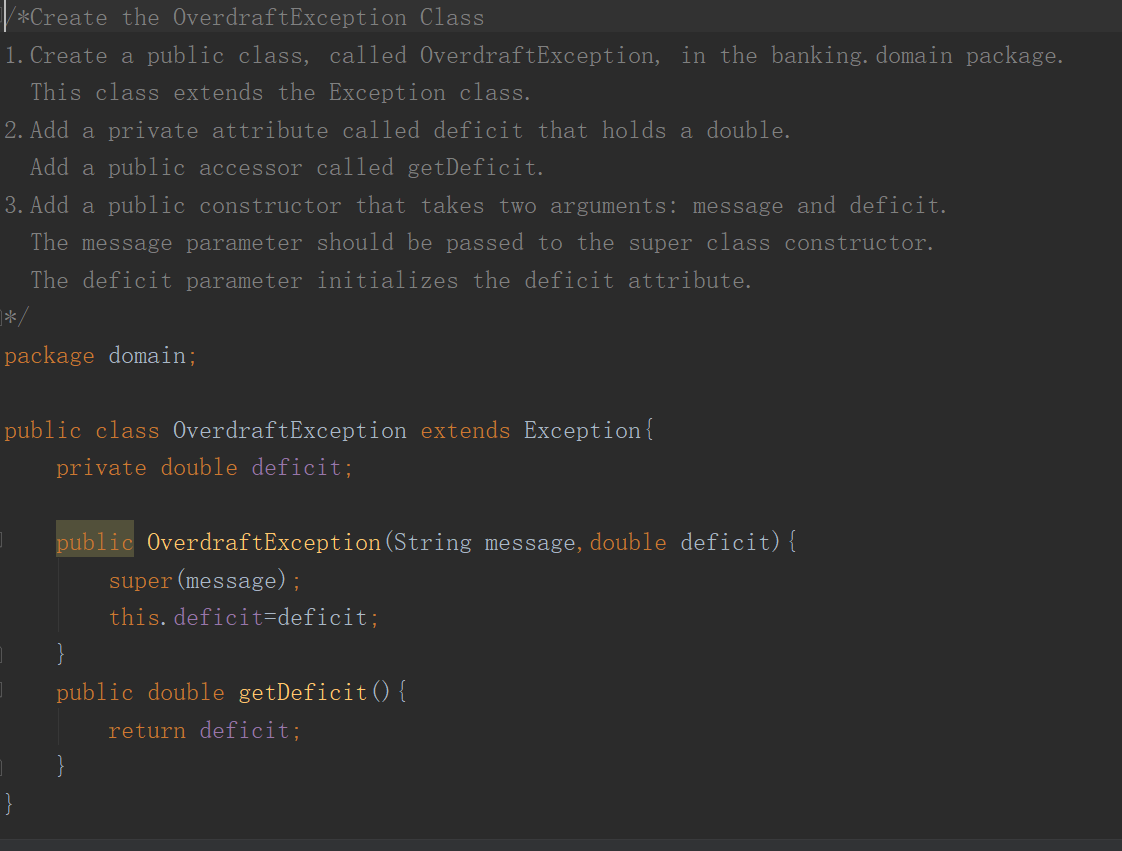


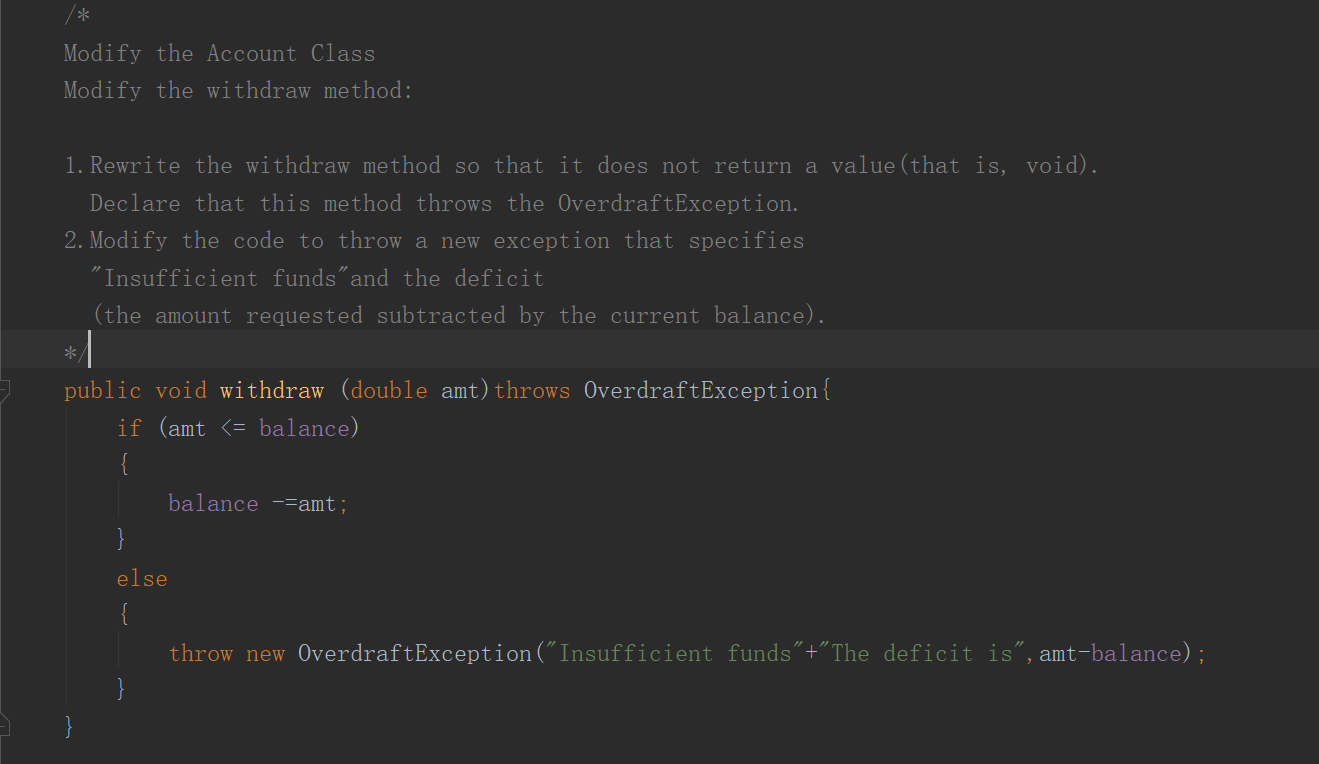


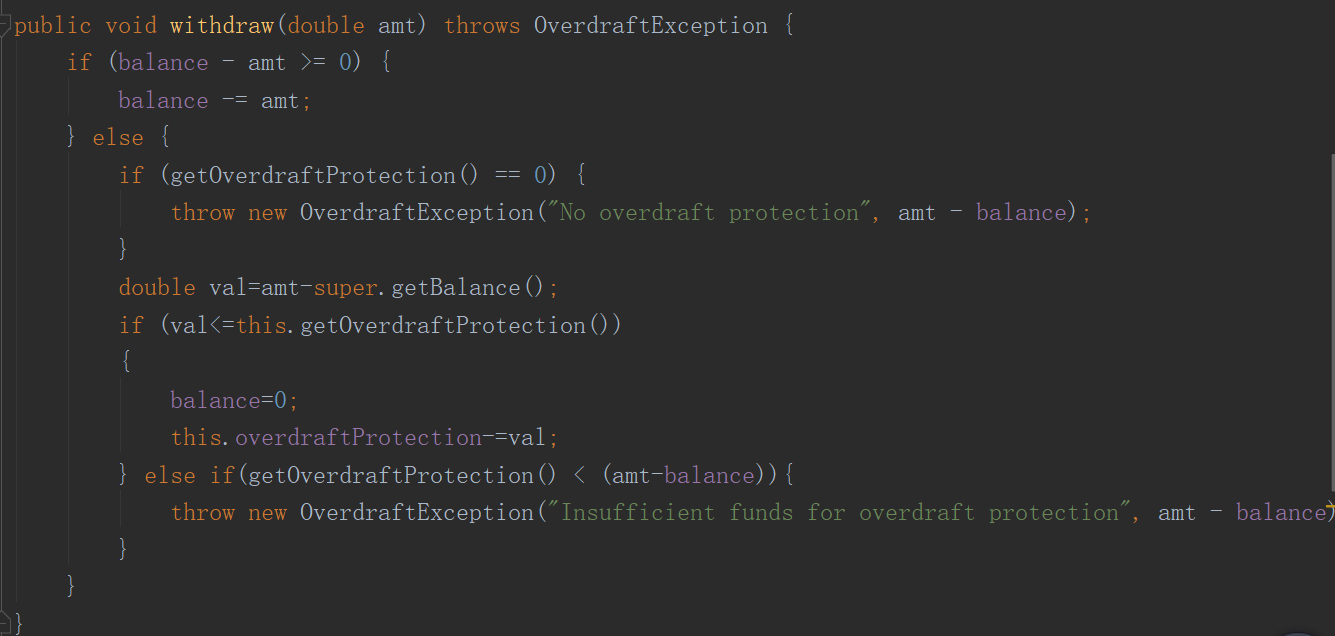
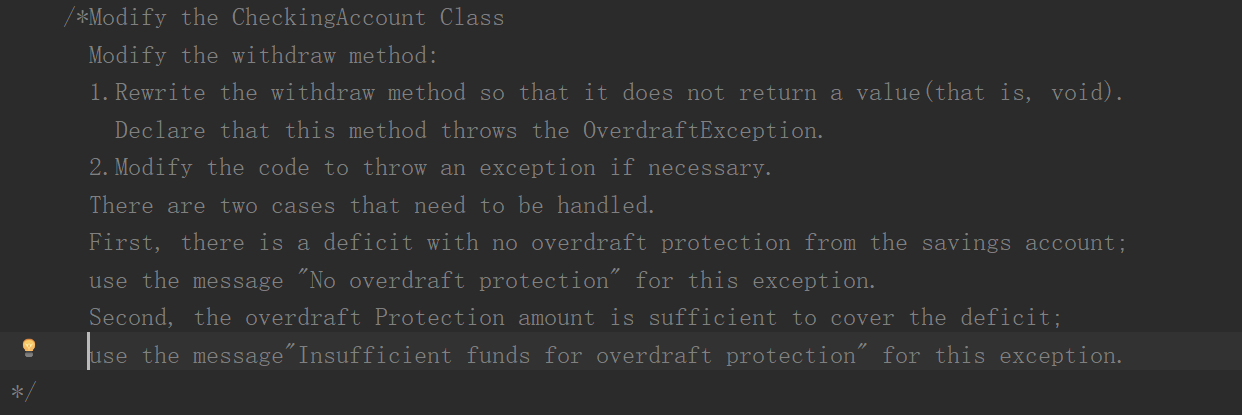




/\*\* step7 ：功能描述 \*\*/



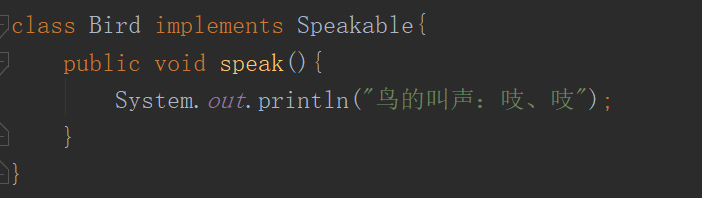


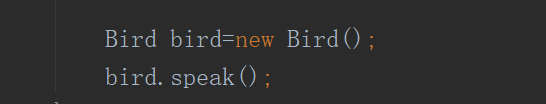


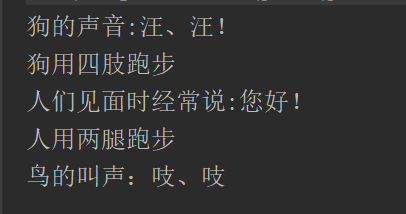


## 二、思考问题陈述

1. 4个.class文件

2. 





3.

抽象类实现：

public abstract class ClassName {

     abstract void fun();

}

接口与抽象类的区别：

1.语法层面上的区别

1） 抽象类可以提供成员方法的实现细节，而接口中只能存在public abstract 方法；

2） 抽象类中的成员变量可以是各种类型的，而接口中的成员变量只能是public static final类型的；

3）接口中不能含有静态代码块以及静态方法，而抽象类可以有静态代码块和静态方法；

4）一个类只能继承一个抽象类，而一个类却可以实现多个接口。

2. 设计层面上的区别

1）抽象类是对一种事物的抽象，即对类抽象，而接口是对行为的抽象。抽象类是对整个类整体进行抽象，包括属性、行为，但是接口却是对类局部（行为）进行抽象。举个简单的例子，飞机和鸟是不同类的事物，但是它们都有一个共性，就是都会飞。那么在设计的时候，可以将飞机设计为一个类Airplane，将鸟设计为一个类Bird，但是不能将“飞行” 这个特性也设计为类，因此它只是一个行为特性，并不是对一类事物的抽象描述。此时可以将 飞行 设计为一个接口Fly，包含方法fly( )，然后Airplane和Bird分别根据自己的需要实现Fly这个接口。然后至于有不同种类的飞机，比如战斗机、民用飞机等直接继承Airplane即可，对于鸟也是类似的，不同种类的鸟直接继承Bird类即可。从这里可以看出，继承是一个 “是不是”的关系，而 接口 实现则是 “有没有”的关系。如果一个类继承了某个抽象类，则子类必定是抽象类的种类，而接口实现则是有没有、具备不具备的关系，比如鸟是否能飞（或者是否具备飞行这个特点），能飞行则可以实现这个接口，不能飞行就不实现这个接口。

2）设计层面不同，抽象类作为很多子类的父类，它是一种模板式设计。而接口是一种行为规范，它是一种辐射式设计。什么是模板式设计？最简单例子，大家都用过ppt里面的模板，如果用模板A设计了ppt B和ppt C，ppt B和ppt C公共的部分就是模板A了，如果它们的公共部分需要改动，则只需要改动模板A就可以了，不需要重新对ppt B和ppt C进行改动。而辐射式设计，比如某个电梯都装了某种报警器，一旦要更新报警器，就必须全部更新。也就是说对于抽象类，如果需要添加新的方法，可以直接在抽象类中添加具体的实现，子类可以不进行变更；而对于接口则不行，如果接口进行了变更，则所有实现这个接口的类都必须进行相应的改动。