

廈門大學



信息学院软件工程系

《JAVA 程序设计》实验报告

实验七

姓名：陈澄

学号：32420212202930

学院：信息学院

专业：软件工程专业

完成时间：2023.04.12

一、实验目的及要求

- 熟悉异常处理
- 熟悉泛型方法和泛型类

二、实验题目及实现过程

题目 2：（构造方法失败）编写一个程序，给出一个构造方法，它将关于构造方法失败的信息传递给一个异常处理器。定义一个 `SomeClass` 类，它在构造方法中抛出异常。程序应创建一个 `SomeClass` 型的对象，并捕获由这个构造方法抛出的异常。

（一）实验环境（集成开发环境、jdk 版本、字符编码等）

集成开发环境：IntelliJ IDEA

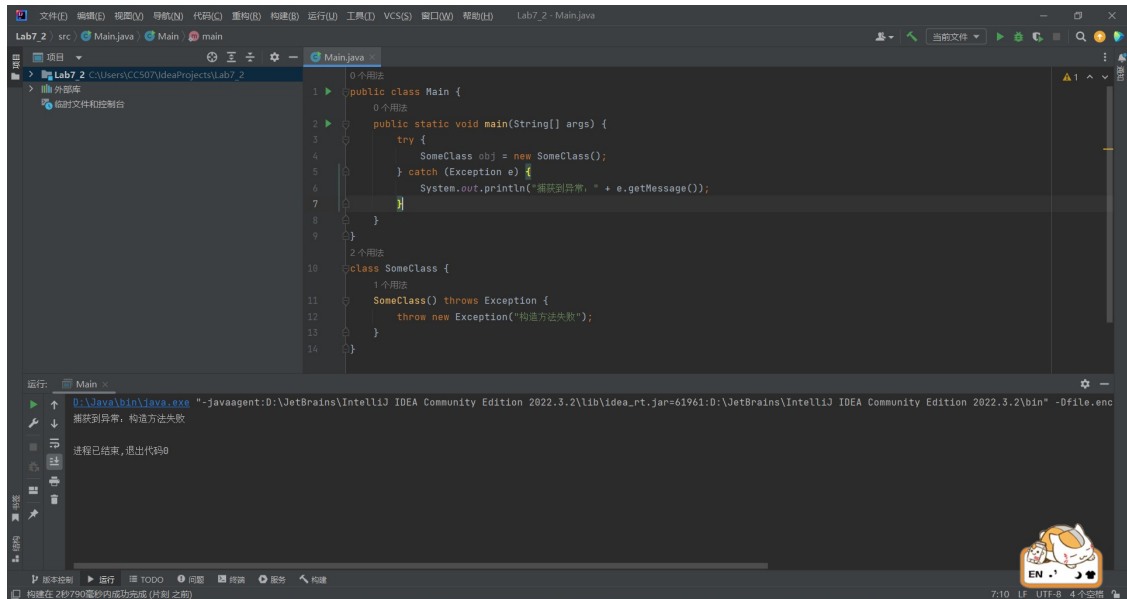
jdk 版本：17.0.5

字符编码：ASCII

（二）实现过程（**本部分为主要评分依据**，请描述解题思路，比如总共设计几个类，各个类的用途、成员、主要方法等及其之间调用关系等）

1. 新建一个类 `SomeClass`，在构建该类的方法中抛出一个 `Exception`，报错内容是“构造方法失败”。
2. 在主函数中利用 `try...catch` 语句：try 中创建一个 `SomeClass` 对象 catch 中捕获异常并输出该异常的信息。

（三）过程截图（**本部分为主要评分依据**，一张全屏截图（必须）、若干运行结果展示图（可选），主要代码（可选））



题目 3: (重抛异常)编写一个演示重抛异常的程序。定义两个方法 `someMethod` 和 `someMethod2`, `someMethod2` 方法的功能就是抛出一个异常。`someMethod` 方法调用 `someMethod2`, 捕获一个异常并重抛它。用 `main` 方法调用 `someMethod` 方法, 并捕获被重抛的异常。输出这个异常的栈踪迹。

(一) 实验环境 (集成开发环境、jdk 版本、字符编码等)

集成开发环境: IntelliJ IDEA

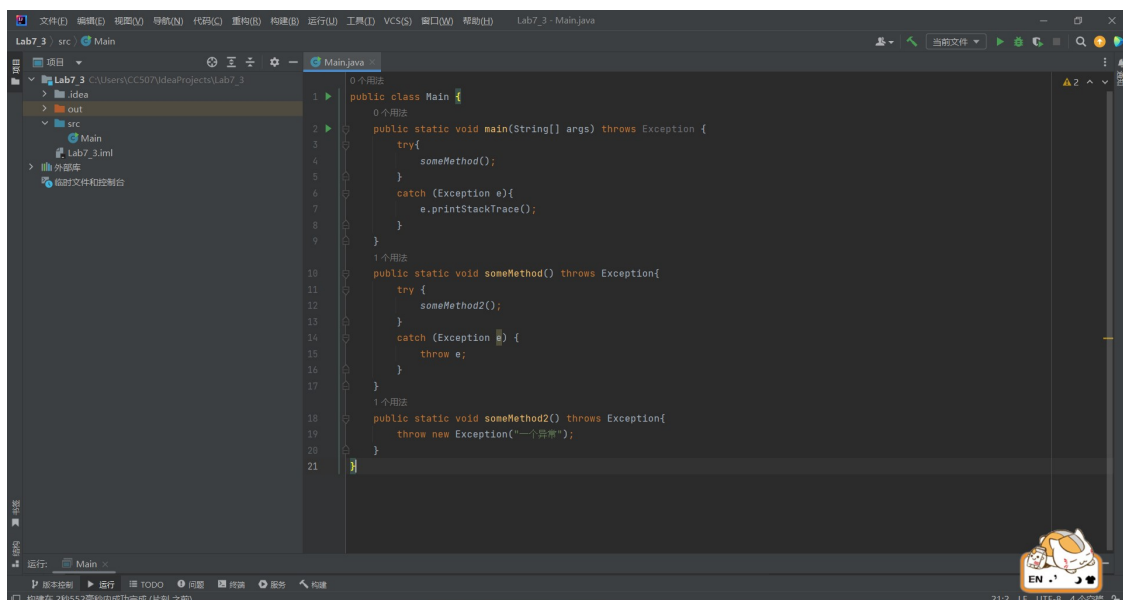
jdk 版本: 17.0.5

字符编码: ASCII

(二) 实现过程 (本部分为主要评分依据, 请描述解题思路, 比如总共设计几个类, 各个类的用途、成员、主要方法等及其之间调用关系等)

- 1.新建一个方法 `someMethod2` 在其中抛出一个异常, 其报错信息为“一个异常”。
- 2.新建一个方法 `someMethod` 在其中利用 `try...catch` 语句, 调用 `someMethod2` 如果出现异常则在 `catch` 中捕获并重新抛出这个异常。
- 3.在主函数中利用 `try...catch` 语句, 调用 `someMethod` 方法, 若出现异常则在 `catch` 中捕获并调用 `printStackTrace` 方法输出该异常的栈踪迹。

(三) 过程截图（**本部分为主要评分依据**，一张全屏截图（必须）、若干运行结果展示图（可选），主要代码（可选））



```
1 public class Main {
2     0个用法
3     public static void main(String[] args) throws Exception {
4         try{
5             someMethod();
6         }
7         catch (Exception e){
8             e.printStackTrace();
9         }
10    }
11    1个用法
12    public static void someMethod() throws Exception{
13        try {
14            someMethod2();
15        }
16        catch (Exception e) {
17            throw e;
18        }
19    }
20    1个用法
21    public static void someMethod2() throws Exception{
22        throw new Exception("一个异常");
23    }
24 }
```



```
运行: Main x
D:\Java\bin\java.exe "-javaagent:D:\JetBrains\IntelliJ IDEA Comm
java.lang.Exception Create breakpoint : 一个异常
    at Main.someMethod2(Main.java:19)
    at Main.someMethod(Main.java:12)
    at Main.main(Main.java:4)
进程已结束,退出代码0
```

题目 4：自定义异常的定义、抛出和捕获：

- (1) 自定义两个异常类：非法姓名异常 `IllegalNameException` 和非法地址异常 `IllegalAddressException`。
- (2) 定义 `Student` 类包含 `name` 和 `address` 属性，和 `setName`、`setAddress` 方法，当姓名长度小于 1 或者大于 5 时抛出 `IllegalNameException`，当地址中不含有“省”或者“市”关键字时抛出 `IllegalAddressException`。
- (3) 编写程序抛出这两种异常，在 `main` 方法中进行捕获并合理地处理。

（一）实验环境（集成开发环境、jdk 版本、字符编码等）

集成开发环境：IntelliJ IDEA

jdk 版本：17.0.5

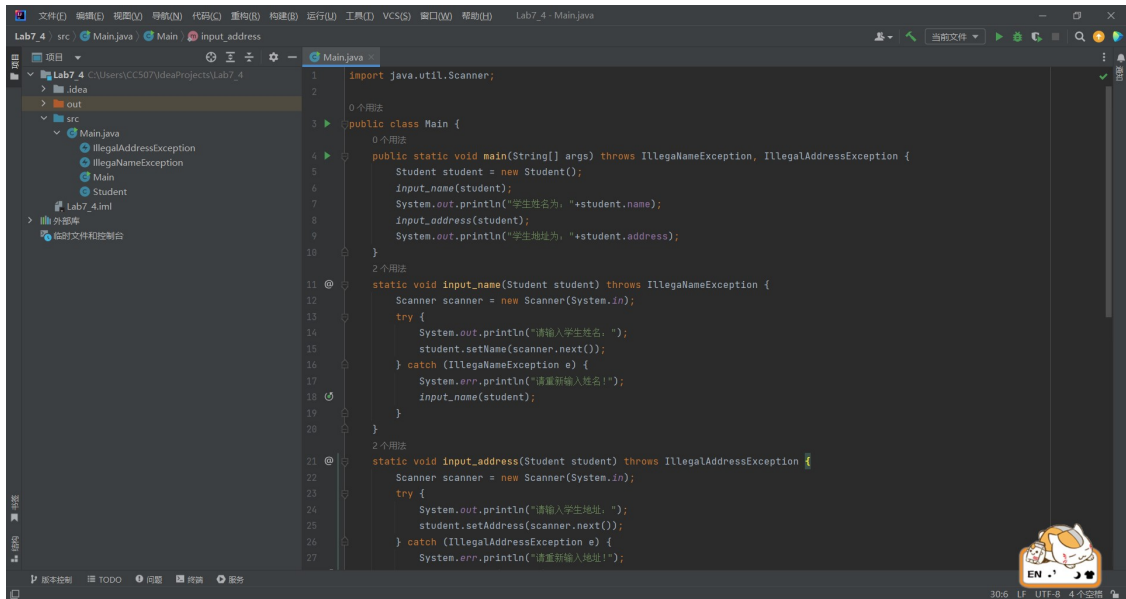
字符编码：ASCII

（二）实现过程（**本部分为主要评分依据**，请描述解题思路，比如总共设计几个类，各个类的用途、成员、主要方法等及其之间调用关系等）

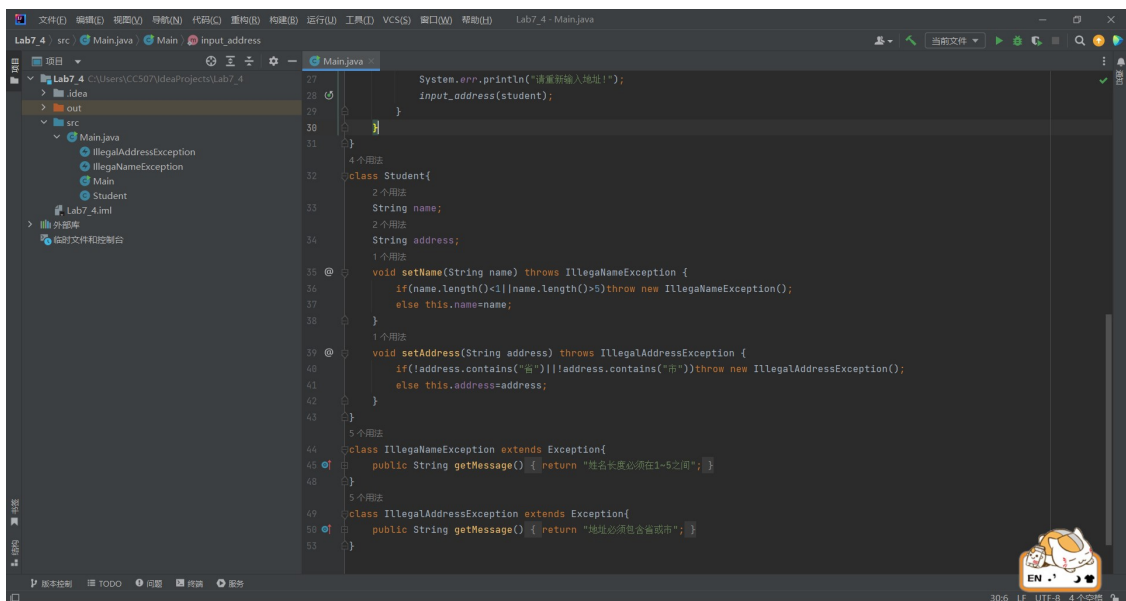
- 1.新建 `IllegalNameException` 类继承自 `Exception` 并在其中重写 `getMessage` 函数，将报错信息改为“姓名长度必须在 1~5 之间”。
- 2.新建 `IllegalAddressException` 类继承自 `Exception` 并在其中重写 `getMessage` 函数，将报错信息改为“地址必须包含省或市”。
- 3.新建 `Student` 类，包含两个 `String` 型变量 `name` 和 `address`，在其中建立 `setName` 方法，输入参数 `name`，当 `name` 的长度小于 1 或者大于 5 的时候抛 `IllegalNameException` 异常。在其中建立 `setAddress` 方法，输入参数 `address`，用 `contains` 方法检测 `address` 中是否包含“省”或者“市”关键词，若都没有则抛出 `IllegalAddressException` 异常。
- 4.在主函数所在的类中创建两个方法 `input_name` 和 `input_address`，其中用 `try...catch` 语句，`try` 中输入学生的姓名（地址），若不合法则在 `catch` 中捕获相应的错误，并让用户重新输入并再次调用该方法。
- 5.主函数中创建一个 `Student` 变量，一次调用上述两个方法，其后输出获得的地址和姓名。

（三）过程截图（**本部分为主要评分依据**，一张全屏截图（必须）、若干运行结果展示图（可选），主要代码（可选））

《Java 程序设计—王美红》实验报告



```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) throws IllegalArgumentException, IllegalAddressException {
5         Student student = new Student();
6         input_name(student);
7         System.out.println("学生姓名为: "+student.name);
8         input_address(student);
9         System.out.println("学生地址为: "+student.address);
10    }
11
12    static void input_name(Student student) throws IllegalArgumentException {
13        Scanner scanner = new Scanner(System.in);
14        try {
15            System.out.println("请输入学生姓名: ");
16            student.setName(scanner.next());
17        } catch (IllegalArgumentException e) {
18            System.err.println("请重新输入姓名!");
19            input_name(student);
20        }
21    }
22
23    static void input_address(Student student) throws IllegalAddressException {
24        Scanner scanner = new Scanner(System.in);
25        try {
26            System.out.println("请输入学生地址: ");
27            student.setAddress(scanner.next());
28        } catch (IllegalAddressException e) {
29            System.err.println("请重新输入地址!");
30        }
31    }
32 }
```



```
27 System.err.println("请重新输入地址!");
28 input_address(student);
29 }
30 }
31
32 class Student {
33     String name;
34     String address;
35
36     void setName(String name) throws IllegalArgumentException {
37         if(name.length()<1||name.length()>5)throw new IllegalArgumentException();
38         else this.name=name;
39     }
40
41     void setAddress(String address) throws IllegalAddressException {
42         if(!address.contains("省")||!address.contains("市"))throw new IllegalAddressException();
43         else this.address=address;
44     }
45 }
46
47 class IllegalNameException extends Exception{
48     public String getMessage(){ return "姓名长度必须在1-5之间"; }
49 }
50
51 class IllegalAddressException extends Exception{
52     public String getMessage(){ return "地址必须包含省或市"; }
53 }
```



题目 5: (泛型方法 `isEqualTo`) 编写 `isEqualTo` 方法的一个简单泛型版本。它用 `equals` 方法比较两个实参相等时返回 `true`, 否则返回 `false`。利用这个泛型方法, 在程序中调用 `isEqualTo` 处理各种内置的类型, 例如 `Object` 或 `Integer`。运行程序时, 传递给 `isEqualTo` 方法的对象会根据它们的内容或者所引用的对象进行比较吗?

(一) 实验环境 (集成开发环境、jdk 版本、字符编码等)

集成开发环境: IntelliJ IDEA

jdk 版本: 17.0.5

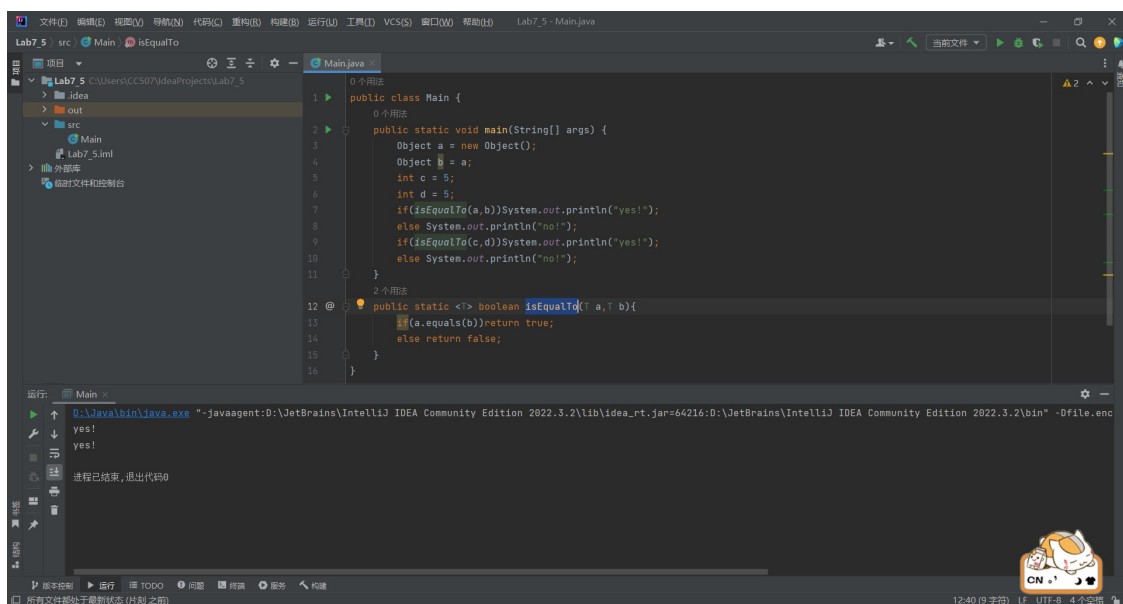
字符编码: ASCII

(二) 实现过程 (本部分为主要评分依据, 请描述解题思路, 比如总共设计几个类, 各个类的用途、成员、主要方法等及其之间调用关系等)

1. 新建一个方法 `isEqualTo`, 选用泛型 `T`, 输入参数为 `T` 类型的 `a, b`, 在其中调用 `equals` 方法, 比较参数 `a, b` 是否相同, 若相同则返回 `true`, 否则返回 `false`。

2.主函数中新建一个 Object 型变量 a,b, 并令 b=a, 调用 isEqualTo 方法比较, 若相同输出 yes, 否则输出 no。新建两个 int 型变量 c,d, 并令他们都为 5, 一样调用 isEqualTo 方法比较。结果都为 yes。

(三) 过程截图 (本部分为主要评分依据, 一张全屏截图 (必须)、若干运行结果展示图 (可选), 主要代码 (可选))



题目 6: (泛型类 Pair) 编写一个泛型类 Pair, 它有两个类型参数 F 和 S, 分别代表一对值中第一个元素第二个元素的类型。为第一个元素和第二个元素添加 get 方法和 set 方法。(提示: 类首部应当是 public class Pair <F, S>.)

(一) 实验环境 (集成开发环境、jdk 版本、字符编码等)

集成开发环境: IntelliJ IDEA

jdk 版本: 17.0.5

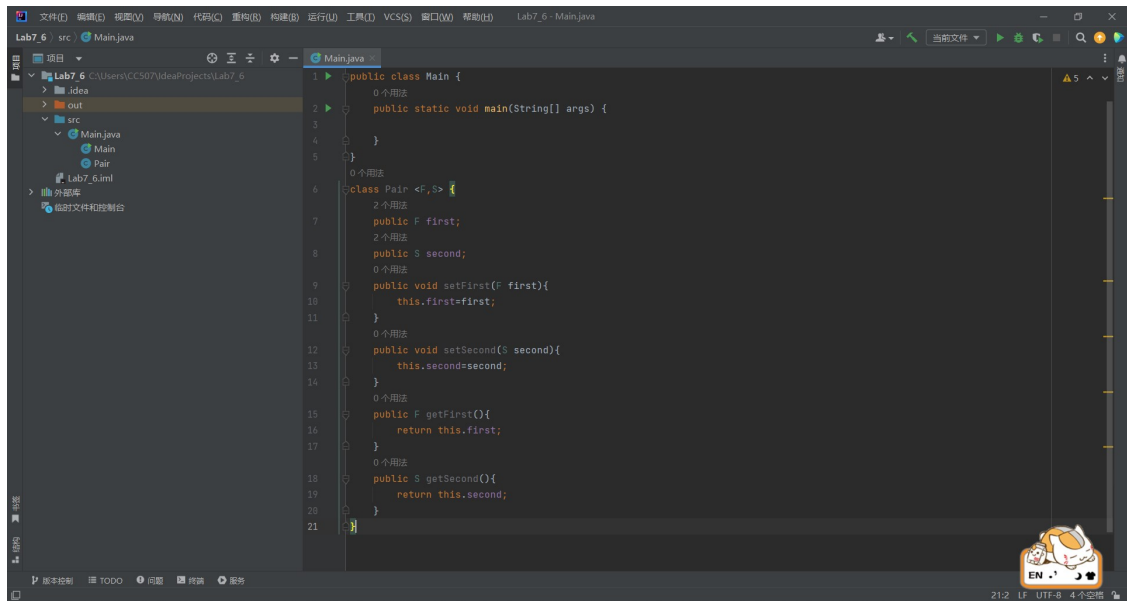
字符编码: ASCII

(二) 实现过程 (本部分为主要评分依据, 请描述解题思路, 比如总共设计几个类, 各个类的用途、成员、主要方法等及其之间调用关系等)

1.创建一个含两个泛型 F,S 的类 Pair, 包含一个 F 型变量 first, 一个 S 型变量 second

2. 建立函数 `setFirst`, `setSecond` 分别用于输入两个变量, 建立函数 `getFirst`, `getSecond` 用于返回两个变量。

(三) 过程截图 (本部分为主要评分依据, 一张全屏截图 (必须)、若干运行结果展示图 (可选), 主要代码 (可选))



题目 7: (混用组合和继承) 将 `CommissionEmployee`—`BaseCommissionEmployee` 继承层次重新建模成一个 `Employee` 层次, 使得每一种员工都具有不同 `CompensationModel` 对象。本练习中, 要求重新实现 `CompensationModel` 类成为一个接口, 提供的公共抽象方法 `earnings` 不包含参数, 且返回一个 `double` 值。然后, 实现了 `CompensationModel` 接口的如下几个类:

- a) `SalariedCompensationModel` 类——对于周薪固定的员工, 这个类需包含一个 `weeklySalary` 实例变量, 且需要实现 `earnings` 方法, 返回 `weeklySalary` 值。
- b) `HourlyCompensationModel` 类——对于按时薪计酬 (包括每周工作超过 40 小时的加班工资) 的员工, 这个类需包含 `wage` 和 `hours` 实例变量, 且需根据工作的小时数实现 `earnings` 方法。

c) `CommissionCompensationModel` 类——对于按佣金付酬的员工，这个类需包含 `grossSales` 和 `commissionRate` 实例变量，还需要实现 `earnings` 方法，它返回 `grossSales x commissionRate` 的结果。

d) `BasePlusCommissionCompensationMode` 类——对按加金付酬的员工，这个类需包含 `grossSales` `commissionRate` 和 `baseSalary` 实例变量，还需要实现 `earnings` 方法，它返回 `baseSalary + grossSales x commissionRate`。

在测试程序中，需为上面描述的每一种 `CompensationModel` 创建一个 `Employee` 对象，然后显示每一类员工的收入。接着，动态地改变员工的 `CompensationModel`，重新显示他的收入。

（一）实验环境（集成开发环境、jdk 版本、字符编码等）

集成开发环境：IntelliJ IDEA

jdk 版本：17.0.5

字符编码：ASCII

（二）实现过程（**本部分为主要评分依据**，请描述解题思路，比如总共设计几个类，各个类的用途、成员、主要方法等及其之间调用关系等）

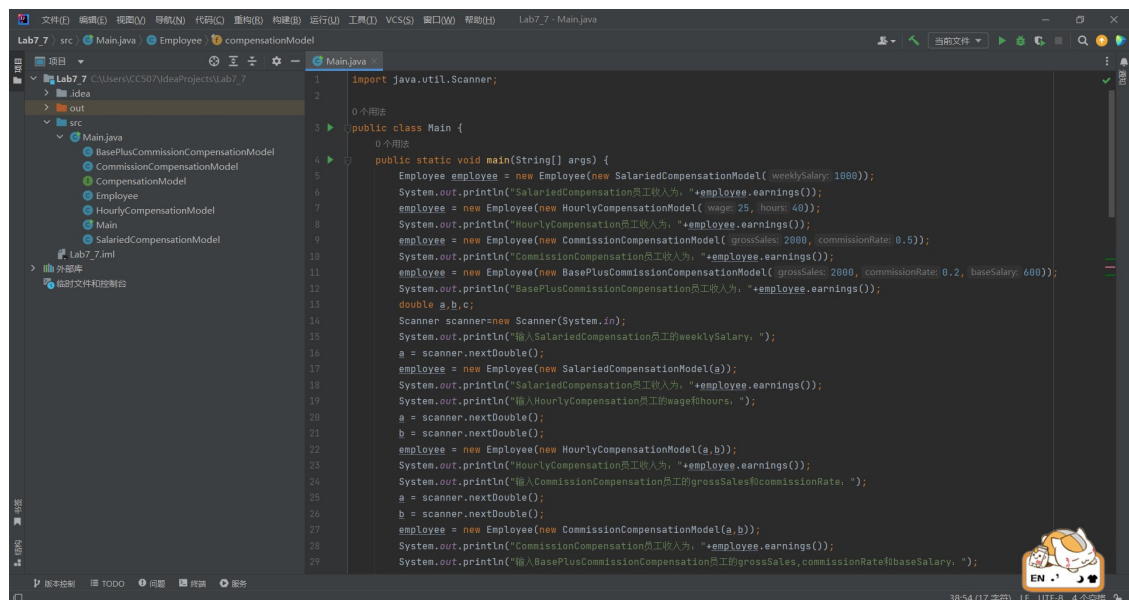
1.新建一个接口 `CompensationModel`，内含一个方法 `earnings` 直接返回 `double` 型的 0

2.建立类 `SalariedCompensationModel` 实现 `CompensationModel` 接口，包含一个 `double` 型变量 `weeklySalary`，在其中实现构建方法，并在其中重写 `earnings` 方法，直接返回 `weeklySalary`。

3.建立类 `HourlyCompensationModel` 实现 `CompensationModel` 接口，包含两个 `double` 型变量 `wage` 和 `hours`，在其中实现构建方法，并在其中重写 `earnings` 方法，若 `hours>40`，返回 `wage*40+wage*3*(hours-40)`，否则返回 `wage*hours`。

4. 建立类 `CommissionCompensationModel` 实现 `CompensationModel` 接口，包含两个 `double` 型变量 `grossSales` 和 `commissionRate`，在其中实现构建方法，并在其中重写 `earnings` 方法，直接返回 `grossSales*commissionRate`。
5. 建立类 `BasePlusCommissionCompensationModel` 实现 `CompensationModel` 接口，包含三个 `double` 型变量 `grossSales`，`commissionRate` 和 `baseSalary`，在其中实现构建方法，并在其中重写 `earnings` 方法，直接返回 `baseSalary + grossSales*commissionRate`。
6. 建立类 `Employee`，内含一个 `CompensationModel` 变量 `compensationModel`，使得员工的类型唯一。包含一个构建方法，还有一个 `earnings` 方法调用 `compensationModel` 下的 `earnings` 方法返回一个 `double` 值。
7. 在主函数中创建一个 `Employee` 型 `employee`，分别将其初始化为四个不同类型员工并给定对应变量的值，调用 `earnings` 方法输出其薪资。其后创建三个 `double` 型变量 `a,b,c` 用于获得输入，根据输入初始化四种不同类型员工，输出其薪资。

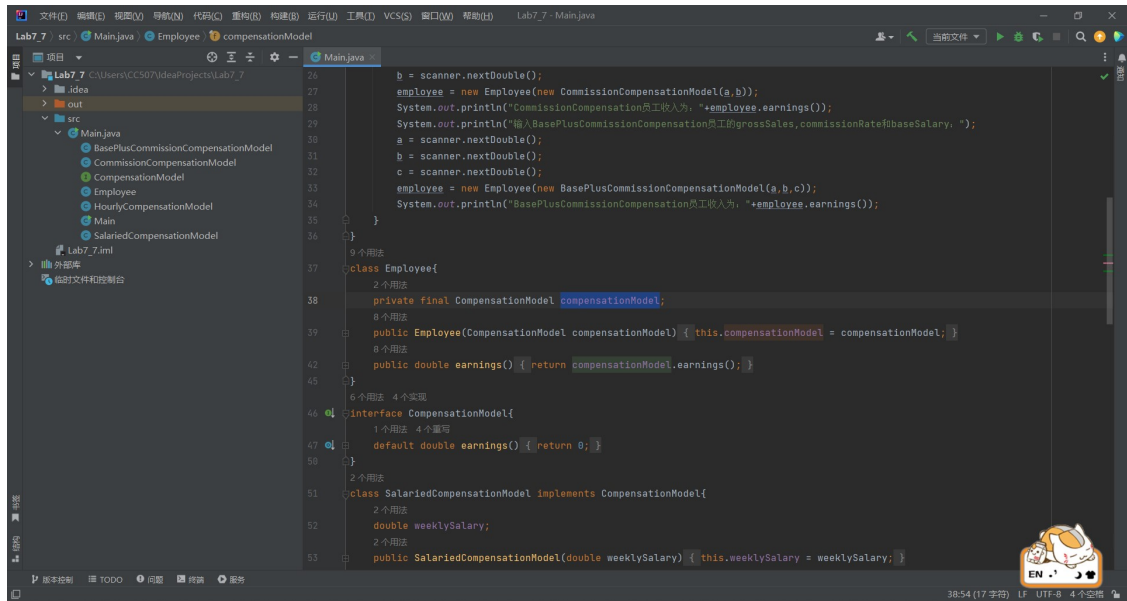
（三）过程截图（本部分为主要评分依据，一张全屏截图（必须）、若干运行结果展示图（可选），主要代码（可选））



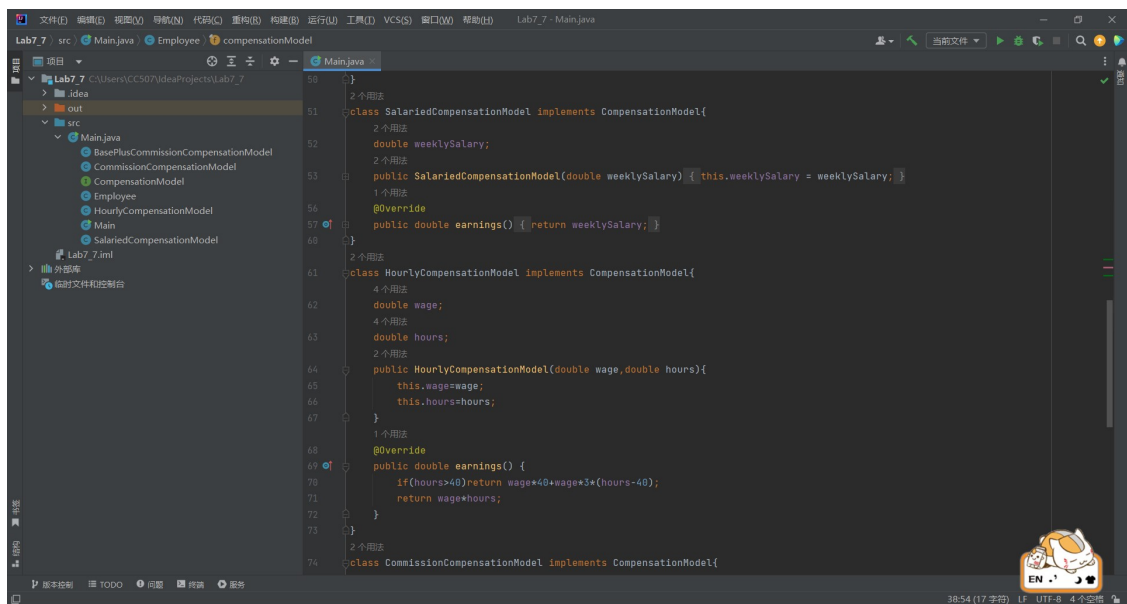
```
import java.util.Scanner;

0 个用法
public class Main {
    0 个用法
    public static void main(String[] args) {
        Employee employee = new Employee(new SalariedCompensationModel(weeklySalary: 1000));
        System.out.println("SalariedCompensation员工收入为: "+employee.earnings());
        employee = new Employee(new HourlyCompensationModel(wage: 25, hours: 40));
        System.out.println("HourlyCompensation员工收入为: "+employee.earnings());
        employee = new Employee(new CommissionCompensationModel(grossSales: 2000, commissionRate: 0.5));
        System.out.println("CommissionCompensation员工收入为: "+employee.earnings());
        employee = new Employee(new BasePlusCommissionCompensationModel(grossSales: 2000, commissionRate: 0.2, baseSalary: 600));
        System.out.println("BasePlusCommissionCompensation员工收入为: "+employee.earnings());
        double a,b,c;
        Scanner scanner=new Scanner(System.in);
        System.out.println("输入SalariedCompensation员工的weeklySalary. ");
        a = scanner.nextDouble();
        employee = new Employee(new SalariedCompensationModel(a));
        System.out.println("SalariedCompensation员工收入为: "+employee.earnings());
        System.out.println("输入HourlyCompensation员工的wage和hours. ");
        a = scanner.nextDouble();
        b = scanner.nextDouble();
        employee = new Employee(new HourlyCompensationModel(a,b));
        System.out.println("HourlyCompensation员工收入为: "+employee.earnings());
        System.out.println("输入CommissionCompensation员工的grossSales和commissionRate. ");
        a = scanner.nextDouble();
        b = scanner.nextDouble();
        employee = new Employee(new CommissionCompensationModel(a,b));
        System.out.println("CommissionCompensation员工收入为: "+employee.earnings());
        System.out.println("输入BasePlusCommissionCompensation员工的grossSales,commissionRate和baseSalary. ");
```

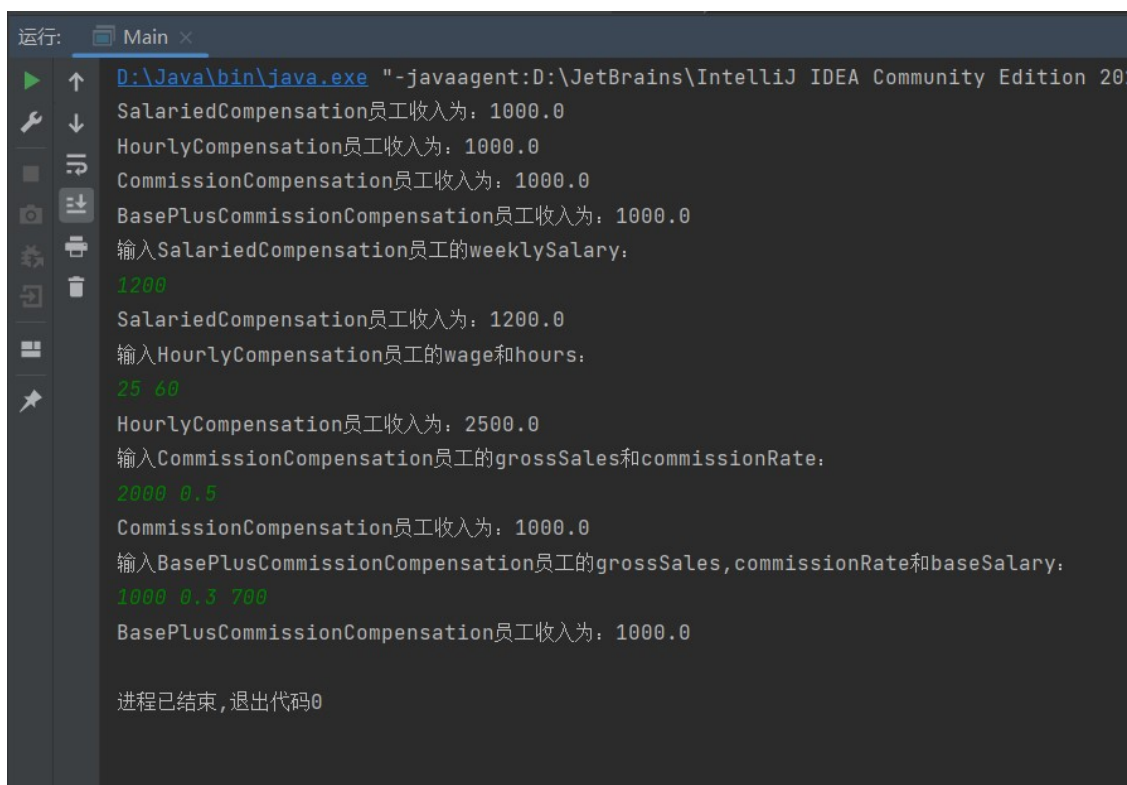
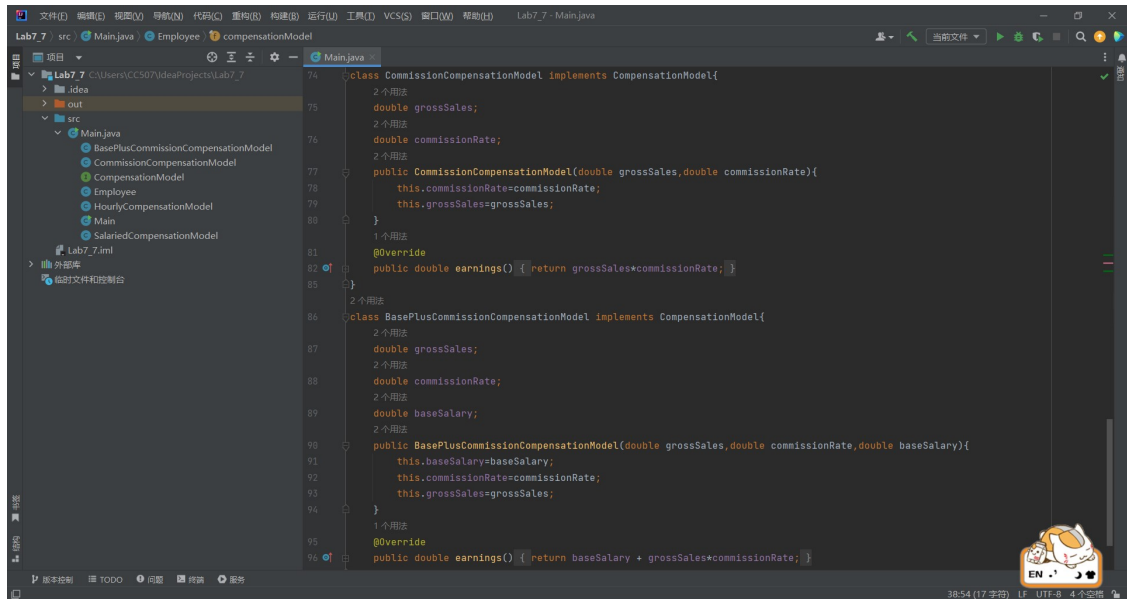
《Java 程序设计—王美红》实验报告



```
26 b = scanner.nextDouble();
27 employee = new Employee(new CommissionCompensationModel(a, b));
28 System.out.println("CommissionCompensation员工收入为: "+employee.earnings());
29 System.out.println("输入BasePlusCommissionCompensation员工的grossSales,commissionRate和baseSalary: ");
30 a = scanner.nextDouble();
31 b = scanner.nextDouble();
32 c = scanner.nextDouble();
33 employee = new Employee(new BasePlusCommissionCompensationModel(a, b, c));
34 System.out.println("BasePlusCommissionCompensation员工收入为: "+employee.earnings());
35 }
36 }
37
38 class Employee{
39     private final CompensationModel compensationModel;
40
41     public Employee(CompensationModel compensationModel) { this.compensationModel = compensationModel; }
42
43     public double earnings() { return compensationModel.earnings(); }
44 }
45
46 interface CompensationModel{
47     default double earnings() { return 0; }
48 }
49
50 class SalariedCompensationModel implements CompensationModel{
51     double weeklySalary;
52
53     public SalariedCompensationModel(double weeklySalary) { this.weeklySalary = weeklySalary; }
```



```
58 }
59
60 class SalariedCompensationModel implements CompensationModel{
61     double weeklySalary;
62
63     public SalariedCompensationModel(double weeklySalary) { this.weeklySalary = weeklySalary; }
64
65     @Override
66     public double earnings() { return weeklySalary; }
67 }
68
69 class HourlyCompensationModel implements CompensationModel{
70     double wage;
71     double hours;
72
73     public HourlyCompensationModel(double wage, double hours){
74         this.wage=wage;
75         this.hours=hours;
76     }
77
78     @Override
79     public double earnings() {
80         if(hours>40) return wage*40+wage*3*(hours-40);
81         return wage*hours;
82     }
83 }
84
85 class CommissionCompensationModel implements CompensationModel{
```



三、实验总结与心得记录

本部分根据实验过程的所得所想描述，记录可供以后复习回看 {可以记录调试过程遇到的问题，自己哪些知识点掌握不够，设计是否有缺陷（比如耗时？耗

内存?) 是否有亮点, 是否有精妙的算法, 或者设计模式的应用, 可吐槽, 也可与其他语言作适当对比。} (本部分不作为平时评分依据)

备注:

建议附带代码提交的方式: 导出工程压缩包。

平时实验成绩以考查参与度为主, 所有实验要求自己完成, 一旦发现抄袭或者其他投机取巧, 取消所有平时成绩