实验一 java类与对象

1. 粘贴程序代码（可截图）。

1.1Employee.java

public class Employee {

    private String name,gender;

    private int id,age;

*// 尽量保持函数参数与成员变量顺序一致*

    Employee(String name,String gender,int id,int age){

        this.name = name;

        this.gender = gender;

        this.id = id;

        this.age = age;

    }

*// 这里的this可以省略*

    String getName(){

        return this.name;

    }

    String getGender(){

        return this.gender;

    }

    int getId(){

        return this.id;

    }

    int getAge(){

        return this.age;

    }

*// 如果返回值是Employee类型,则可以进行链式调用*

    Employee setName(String name){

        this.name = name;

*//调用远程服务,可能是对数据库的操作,可以使用boolean类型判断操作成功还是失败*

*// employee.setname(name);*

        return this;

    }

*//   Employee setName(String name){*

*//     // 链式调用*

*//     this.name = name;*

*//     //调用远程服务,可能是对数据库的操作,可以使用boolean类型判断操作成功还是失败*

*//     employee.setname("张三").setId();*

*//     return this;*

*// }*

    Employee setGender(String gender){

        this.gender = gender;

        return this;

    }

    Employee setId(int id){

        this.id = id;

        return this;

    }

    Employee setAge(int age){

        this.age = age;

        return this;

    }

    @Override

    public String toString(){

        String str = "Employee [name=" + name + ", gender=" + gender + ", id=" + id + ", age=" + age + "]";

        return str;

    }

}

2.2Test.java

public class Test {

    public static void main(String[] args) {

        String name = "张三";

        String gender = "男";

        int id = 202300001;

        int age = 22;

        Employee employee = new Employee(name,gender,id,age);

        System.out.println(employee);

        employee.setName("谭棵").setGender("男").setId(202306630).setAge(21);

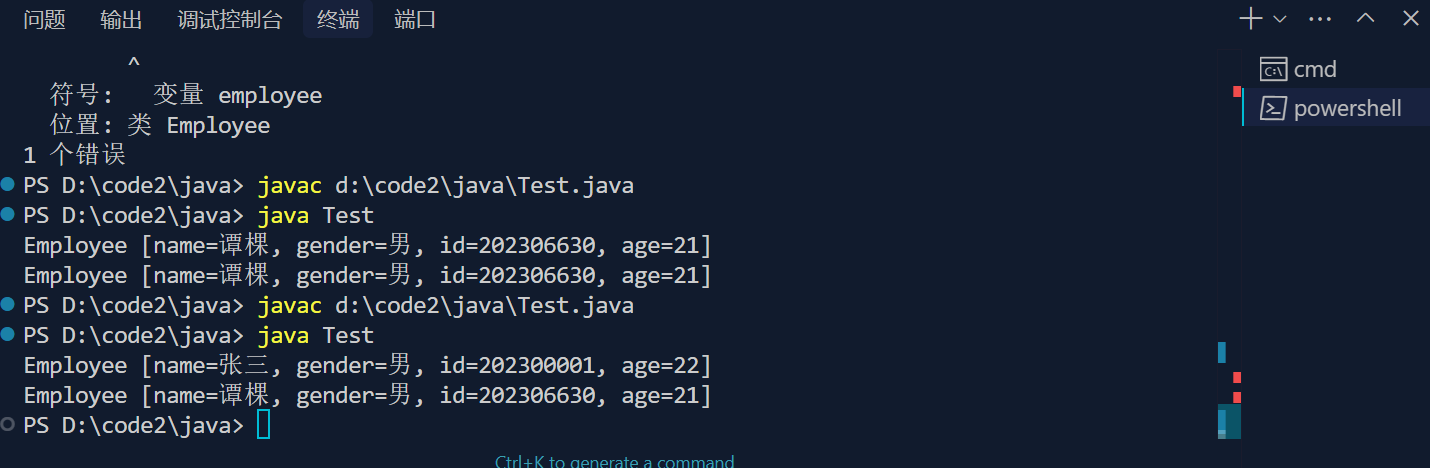
        System.out.println(employee);

    }

}

1. 粘贴程序的输出信息。

2.3运行结果



## 实验二 继承、接口与多态

1. 粘贴程序代码（可截图）。

* 实验一
  + Employee.java

package people;

import people.People;

public class Employee extends People{

    int **id**;

    protected String **test** = "子类";

*// @Override*

*//  public String getTest(){*

*//     return this.test;*

*//  }*

    public Employee(String name,String gender,int age,int id){

        super(name,gender,age);

        this.**id** = id;

    }

    public void speak(){

        System.**out**.println("speak");

    }

    public void eat(){

        System.**out**.println("eat");

    }

    public void work(){

        System.**out**.println("work");

    }

    public int getId(){

        return this.**id**;

    }

    public Employee setId(int id){

        this.**id** = id;

        return this;

    }

}

* + People.java

package people;

public abstract class People {

    protected String **name**,**gender**;

    protected int **age**;

    protected String **test** = "父类";

    public String getTest(){

        return this.**test**;

    }

    public People(String name,String gender,int age){

        this.**name** = name;

        this.**gender** = gender;

        this.**age** = age;

    }

*// 定义抽象方法*

    public abstract void speak();

    public abstract void eat();

    public String getName(){

        return this.**name**;

    }

    public String getGender(){

        return this.**gender**;

    }

    public int getAge(){

        return this.**age**;

    }

*// 链式调用,返回this当前对象指针*

    public People setName(String name){

        this.**name** = name;

        return this;

    }

    public People setGender(String gender){

        this.**gender** = gender;

        return this;

    }

    public People setAge(int age){

        this.**age** = age;

        return this;

    }

}

* + Main.java

import java.util.Scanner;

import people.Employee;

public class Main {

    public static void main(String[] args){

        Employee employee = new Employee("谭棵","男",20,202306630);

        System.**out**.println("这是一名员工：");

        System.**out**.println("姓名："+employee.getName());

        System.**out**.println("性别："+employee.getGender());

        System.**out**.println("年龄："+employee.getAge());

        System.**out**.println("工号："+employee.getId());

*// 父类定影的返回的是父类的指针,setName返回的是People的指针,setName返回的是Employee的指针,父类没有work方法*

*// ((Employee)employee).setName("李四").setAge(20);*

*// ((Employee)employee.setName("李四").setAge(20)).work();*

*//*

        employee.setName("李四").setAge(20);

        employee.eat();

        employee.speak();

        employee.work();

        System.**out**.println();

*// People people = new People("丽丝","女",16);*

*// System.out.println("这是一个普通的人：");*

*// System.out.println("姓名："+people.getName());*

*// System.out.println("性别："+people.getGender());*

*// System.out.println("年龄："+people.getAge());*

*// people.eat();*

*// people.speak();*

*// System.out.println(employee.getTest()); 返回的是父类的属性,因为是引用关系*

    }

}

* 实验二采用继承的方式
  + Animal.java

package animal;

public class Animal {

*// 动物的吃法都是不一样的*

    public void eat(){

        System.**out**.println("eat");

    }

*// 动物的睡眠方法*

    public void sleep() {

        System.**out**.println("sleep");

    }

}

* + Rabbit.java

package animal;

public class Rabbit extends Animal {

    private String **name**;

    private int **age**;

    private String **gender**;

    @Override

    public void eat() {

        System.**out**.println("我是兔子，我吃草！");

    }

}

* + Tiger.java

*// Source code is decompiled from a .class file using FernFlower decompiler.*

package animal;

public class Tiger implements Animal {

   private String name;

   private int age;

   private String gender;

   public Tiger() {

   }

   public void eat() {

      System.out.println("我是老虎，我吃肉！");

   }

}

* + Main.java

import animal.Animal;

import animal.Rabbit;

import animal.Tiger;

public class Main{

    public static void main(String[] args) {

            Rabbit r = new Rabbit();

            Tiger t = new Tiger();

            r.eat();

            t.eat();

            r.sleep();

            t.sleep();

        }

}

* 实验二采用接口的方式
  + 与继承类似,只是将Animal改为了interface,Rabbit和Tigger使用implements调用接口
  + Animal.java

package animal;

public interface *Animal* {

*// 动物的吃法都是不一样的*

    void eat();

*// 动物的睡眠方法*

    default void sleep() {

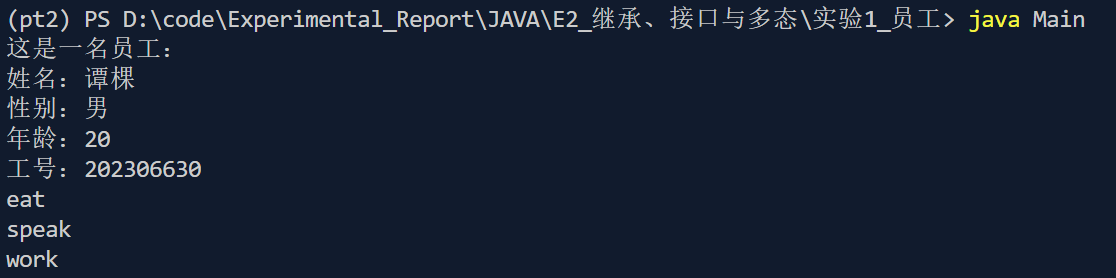
        System.**out**.println("sleep");

    }

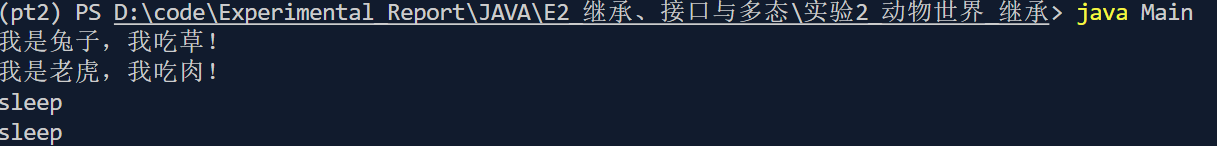
}

1. 粘贴程序的输出信息。

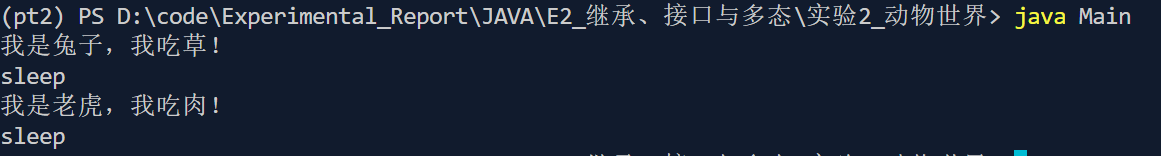
* 实验一运行结果



* 实验二采用继承的方式



* 实验二采用接口的方式



实验三 异常处理

1. 粘贴程序代码（可截图）。

* 实验一
* Week.java

package com.sicau;

*// 这段代码外层做防御,内层不做防御*

public class Week {

    private final String **data**[] = { "星期一", "星期二", "星期三", "星期四", "星期五", "星期六", "星期日" };

    public String getDays(int index) {

        return **data**[index];

    }

    private int **index**;

*// public Week(int index) {*

*//     this.index = index;*

*// }*

    public String toString() {

        return **data**[**index**];

    }

}

*// 外层做防御,内层不做*

*// 一般web开发常用*

*// 外层和内层都做*

*// 高可靠性,资源开销大,例如金融系统*

*// 内层做防御,外层不做*

*// 系统级底层开发常用,但是内层极其复杂*

* Main.java

package com.sicau;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.**in**);

        Week week = new Week();

        String input\_str = null;

        String output\_str = null;

        boolean invalid = true; *// Changed to true to enter the loop*

        int index = -1;

        System.**out**.println("开始执行:请输入1-7");

        while(invalid){

            input\_str = sc.nextLine();

            try{

                index = Integer.parseInt(input\_str) - 1;

                if(index < 0 || index > 6){

                    System.**out**.println("解析成功:输入的数字不在1-7之间");

                } else {

                    invalid = false;

                }

            } catch (NumberFormatException e) {

                System.**out**.println("解析失败:只能输入数字1-7");

            }

        }

        sc.close(); *// Added scanner close*

        output\_str = week.getDays(index);

        System.**out**.println(output\_str);

    }

}

* 实验二
* Main.java

package com.tk;

import java.io.IOException;

public class Test03 {

    public static void main(String[] args) throws IOException {

        int i = 1, j;

        try{

            System.**out**.println("Try:这是一个异常处理的例子：");

            j = i/0;

            throw new ArithmeticException();

        }catch(ArithmeticException e){

            System.**out**.println("Catch:"+e+";"+"\n"+"reason:"+e.getMessage());

        }finally{

            System.**out**.println("Finally:must go inside finally");

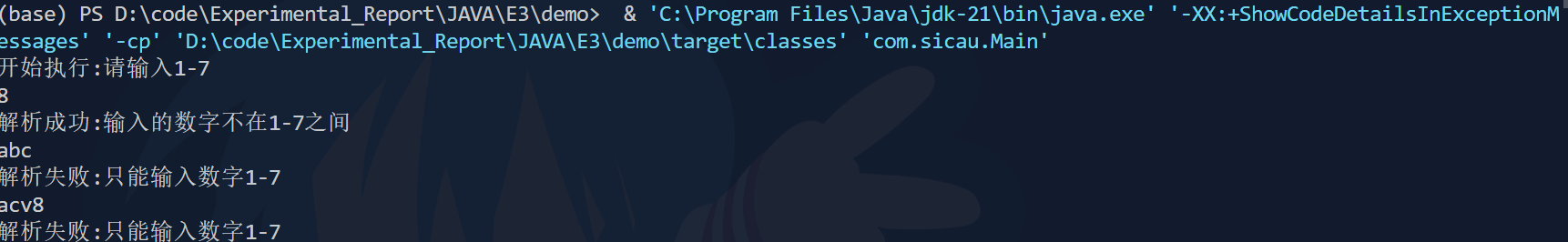
        }

    }

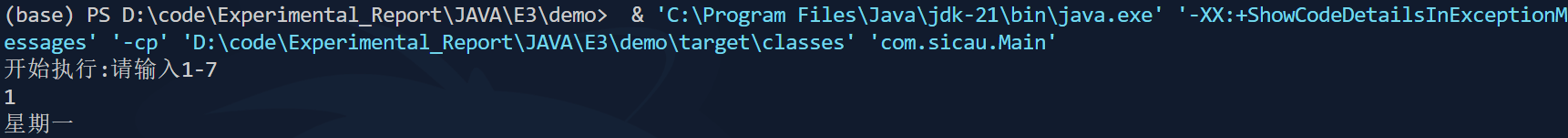
}

1. 粘贴程序的输出信息。

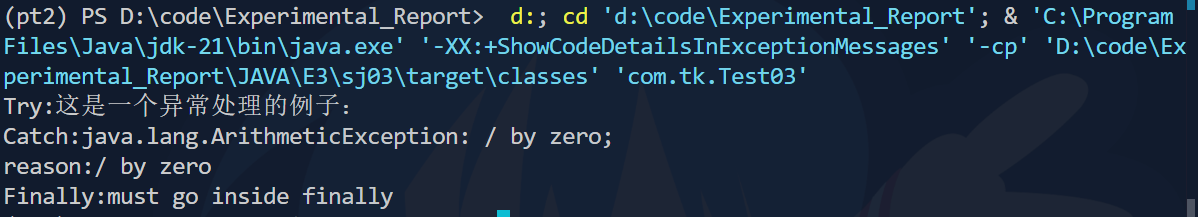
* 实验一运行结果
* 异常捕获



* 正常输入



* 实验二运行结果



实验四 输入输出

1. 粘贴程序代码（可截图）。
2. 粘贴程序的输出信息。