

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

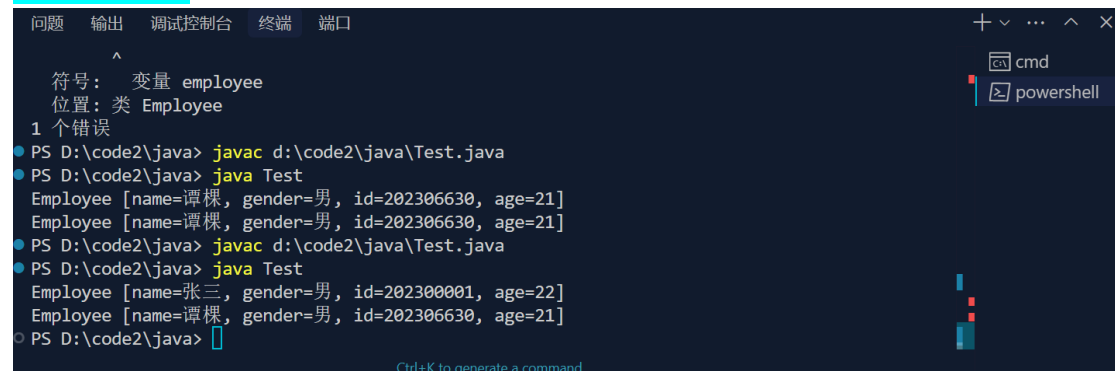
```

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);
    }
}

```

2. 粘贴程序的输出信息。

2.3 运行结果



The screenshot shows an IDE terminal window with the following content:

```
问题 输出 调试控制台 终端 端口
^
符号: 变量 employee
位置: 类 Employee
1 个错误
• PS D:\code2\java> javac d:\code2\java\Test.java
• PS D:\code2\java> java Test
Employee [name=谭棵, gender=男, id=202306630, age=21]
Employee [name=谭棵, gender=男, id=202306630, age=21]
• PS D:\code2\java> javac d:\code2\java\Test.java
• PS D:\code2\java> java Test
Employee [name=张三, gender=男, id=202300001, age=22]
Employee [name=谭棵, gender=男, id=202306630, age=21]
• PS D:\code2\java> 
```

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";

    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());
        // 父类定义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.setName("李四").setAge(20);
        employee.eat();
        employee.speak();
        employee.work();
        System.out.println();

        // 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

```
package animal;

public class Tiger implements Animal {
    private String name;
    private int age;
    private String gender;

    @Override
    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");
    }
}
```

2. 粘贴程序的输出信息。

➤ 实验一运行结果

```
(pt2) PS D:\code\Experimental_Report\JAVA\E2_继承、接口与多态\实验1_员工> java Main
这是一名员工：
姓名：谭棵
性别：男
年龄：20
工号：202306630
eat
speak
work
```

➤ 实验二采用继承的方式

```
(pt2) PS D:\code\Experimental_Report\JAVA\E2_继承、接口与多态\实验2_动物世界_继承> java Main
我是兔子，我吃草！
我是老虎，我吃肉！
sleep
sleep
```

➤ 实验二采用接口的方式

```
(pt2) PS D:\code\Experimental_Report\JAVA\E2_继承、接口与多态\实验2_动物世界> java Main
我是兔子，我吃草！
sleep
我是老虎，我吃肉！
sleep
```

实验三 异常处理

1. 粘贴程序代码（可截图）。

● 实验一

- Week.java(未采用枚举)

```
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 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

```

➤ Week.java(采用枚举)

```

package com.sicau;
// 这段代码外层做防御, 内层不做防御

public class Week {
    // 采用枚举
    private enum Days {
        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 Days.values()[index].toString();
    }

    private int index;

    public String toString() {
        return Days.values()[index].toString();
    }
}

```

➤ 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");
    }
}
}

```

2. 粘贴程序的输出信息。

● 实验一运行结果

➤ 异常捕获

```

(base) PS D:\code\Experimental_Report\JAVA\E3\demo> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\code\Experimental_Report\JAVA\E3\demo\target\classes' 'com.sicau.Main'
开始执行:请输入1-7
3
解析成功:输入的数字不在1-7之间
abc
解析失败:只能输入数字1-7
acv8
解析失败:只能输入数字1-7

```

➤ 正常输入

```

(base) PS D:\code\Experimental_Report\JAVA\E3\demo> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\code\Experimental_Report\JAVA\E3\demo\target\classes' 'com.sicau.Main'
开始执行:请输入1-7
1
星期一

```

● 实验二运行结果

```

(pt2) PS D:\code\Experimental_Report> d.; cd 'd:\code\Experimental_Report'; & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\code\Experimental_Report\JAVA\E3\sj03\target\classes' 'com.tk.Test03'
Try:这是一个异常处理的例子:
Catch:java.lang.ArithmeticException: / by zero;
reason:/ by zero
Finally:must go inside finally

```

实验四 输入输出

1. 粘贴程序代码（可截图）。

● 实验一代码

➤ Main.java

```

package com.tk;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
public class Main {
    public static void main(String[] args) {
        String content_text = "./content.txt"; // 注意路径为项目的相对路径
        try(
            FileWriter fw = new FileWriter(content_text);
            BufferedWriter bw = new BufferedWriter(fw);
        ) {
            // 这里使用的硬编码的方法
            bw.write("This is test");
            bw.newLine();
        }
    }
}

```

```

        bw.write("这是一个写入测试!");
    } catch (IOException e) {
        System.out.println("文件写入失败");
        e.printStackTrace();
    }
}
}

```

➤ 改进硬编码问题接收输入并写入的代码

```

package com.tk;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        String content_text = "./content.txt"; // 注意路径为项目的相对路径
        try(
            FileWriter fw = new FileWriter(content_text);
            BufferedWriter bw = new BufferedWriter(fw);
        ) {
            // 这里使用的是 Scanner, 并根据 Scanner 的 hasNext() 方法判断是否还有
            输入
            Scanner sc = new Scanner(System.in);
            System.out.println("请输入内容: ");
            while(sc.hasNext()){
                String input_str = sc.nextLine();
                bw.write(input_str);
                bw.newLine();
            }
            bw.close();
        } catch (IOException e) {
            System.out.println("文件写入失败");
            e.printStackTrace();
        }
    }
}

```

● 实验二代码

➤ Main.java

```

package com.tk;

import java.io.*;

public class Main {

```

```

public static void main(String[] args) {
    String test_text = "./test1.txt";
    String test_copy_text = "./test2.txt";

    // 1. 写入 test1.txt
    try (
        FileWriter fw1 = new FileWriter(test_text);
        BufferedWriter out = new BufferedWriter(fw1)
    ) {
        out.write("Hello!");
        out.newLine();
        out.write("Hello, nice to meet you!");
        out.newLine();
    } catch (IOException e) {
        System.out.println("文件写入失败");
        e.printStackTrace();
    }

    // 2. 读取 test1.txt 并复制到 test2.txt
    try (
        FileReader fr = new FileReader(test_text);
        BufferedReader in = new BufferedReader(fr);
        FileWriter fw2 = new FileWriter(test_copy_text);
        BufferedWriter out2 = new BufferedWriter(fw2)
    ) {
        String line;
        while ((line = in.readLine()) != null) {
            out2.write(line);
            out2.newLine();
        }
    } catch (IOException e) {
        System.out.println("文件复制失败");
        e.printStackTrace();
    }
}
}

```

● 跟着老师敲的示例代码

```

package test4;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.Buffer;

```

```

public class Main {
    public static void main(String[] args) {
        String content_text = "./content.txt";
        try(
            FileWriter fw = new FileWriter(content_text); //注意文件的相对
            //路径的问题
            BufferedWriter bw = new BufferedWriter(fw); //执行完以后
            //resource 就关闭
        ) {
            for (int i = 0; i < 3; i++) {
                bw.write("这是第" + (i + 1) + "行");
                bw.newLine();
            }

        } catch (IOException e) {
            System.out.println("文件写入失败");
            e.printStackTrace();
        }

        try(
            BufferedReader br = new BufferedReader(new
            FileReader(content_text));
            BufferedWriter bw_copy = new BufferedWriter(new
            FileWriter("./content_copy.txt"));
        ) {
            String input_str = "";
            while ((input_str = br.readLine()) != null) {
                bw_copy.write(input_str);
                bw_copy.newLine();
            }
        } catch (IOException e) {
            System.out.println("文件读取失败");
            e.printStackTrace();
        }

    }
}

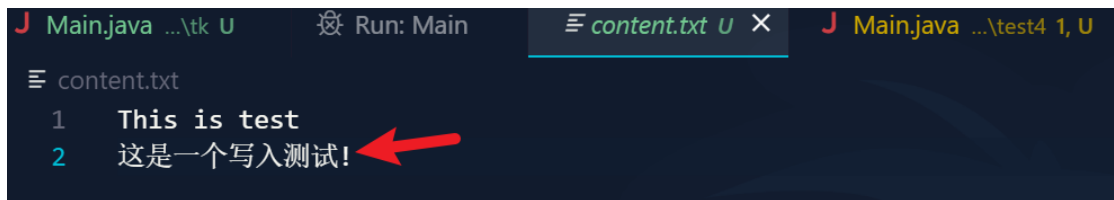
```

2. 粘贴程序的输出信息。

● 实验一运行结果

➤ 硬编码代码运行结果

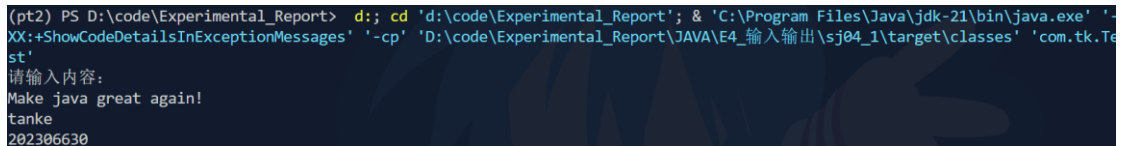
■ Content.txt 文件中的内容



```
content.txt
1 This is test
2 这是一个写入测试!
```

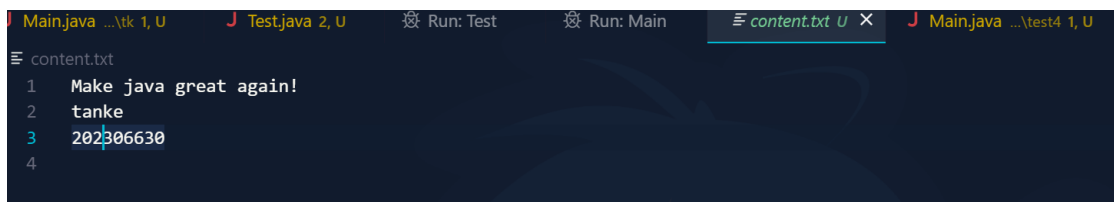
➤ 改进硬编码的运行结果

■ 终端输入



```
(pt2) PS D:\code\Experimental_Report> d:; cd 'd:\code\Experimental_Report'; & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\code\Experimental_Report\JAVA\E4_输入输出\sj04_1\target\classes' 'com.tk.Test'
请输入内容:
Make java great again!
tanke
202306630
```

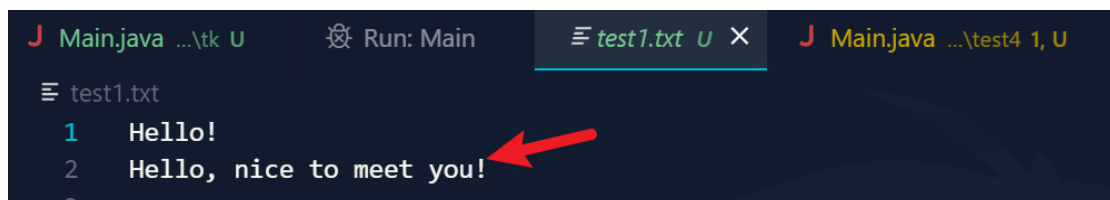
■ Content.txt 文件中的内容



```
content.txt
1 Make java great again!
2 tanke
3 202306630
4
```

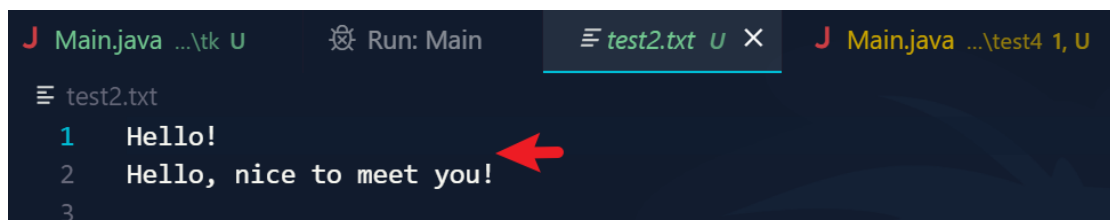
● 实验二运行结果

➤ Test1.txt 文件中的信息



```
test1.txt
1 Hello!
2 Hello, nice to meet you!
```

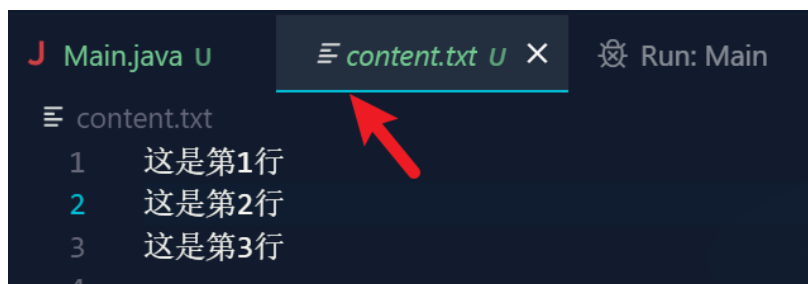
➤ Test2.txt 文件中的信息



```
test2.txt
1 Hello!
2 Hello, nice to meet you!
3
```

● 示例结果

➤ Content.txt 文件中的内容



```
content.txt
1 这是第1行
2 这是第2行
3 这是第3行
```

➤ Content_copy 文件中的内容

