实验 06：继承与多态

一、实验目的

1. 掌握继承的基本概念，理解父类与子类的关系，学会在 Java 中实现继承。
2. 理解多态性的实现方式，包括方法的重载和覆盖。
3. 熟悉 `Object` 类和 `Class` 类的使用，掌握对象引用转换和访问继承成员的方法。
4. 理解访问控制符（`public`、`protected`、`private`）和 `final` 修饰符的作用。
5. 综合运用继承与多态的知识，设计并实现一个较为复杂的医院患者管理系统。

二、实验学时

2 学时

三、实验类型

验证性

四、实验需求

1、硬件

每人配备计算机 1 台，建议优先使用个人计算机开展实验。

2、软件

Windows 10/11 操作系统

IntelliJ IDEA 版本号：2024.03

安装 IntelliJ IDEA，以及 Java 运行所需要的相关基础环境。

3、网络

本地主机能够访问互联网和实验中心网络。

4、工具

无。

五、实验任务

5.1 **继承的实现**：设计一个 `Person` 类作为父类， `Patient` 和 `Doctor` 类作为子类。

5.2 **方法的重载与覆盖**：在 `Doctor` 类中实现方法的重载和覆盖。

5.3 **对象引用转换**：实现父类与子类之间的对象引用转换，并使用 `instanceof` 检查对象类型。

5.4 **访问控制与 `final` 修饰符**：使用访问控制符和 `final` 修饰符限制类的成员访问。

5.5 **综合任务**：设计一个 `Hospital` 类，管理患者和医生，并实现多态性的诊疗过程。

六、实验内容及步骤

6.1、任务 1：继承的实现

步骤：

(1) 设计一个 `Person` 类作为父类，包含姓名和年龄属性。
(2) 设计 `Patient` 类和 `Doctor` 类作为子类，分别扩展患者和医生的特有属性。

示例代码：

Java

```java
// Person.java
public class Person {
    // 属性
    private String name;
    private int age;

    // 构造函数
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Getter 方法
    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    // 打印信息
    public void printInfo() {
        System.out.println("姓名: " + name);
        System.out.println("年龄: " + age);
    }
}

// Patient.java
public class Patient extends Person {
    // 特有属性
    private String symptom;

    // 构造函数
    public Patient(String name, int age, String symptom) {
        super(name, age);
        this.symptom = symptom;
    }

    // Getter 方法
    public String getSymptom() {
        return symptom;
```

```
    }

    // 覆盖父类方法
    @Override
    public void printInfo() {
        super.printInfo();
        System.out.println("症状: " + symptom);
    }
}

// Doctor.java
public class Doctor extends Person {
    // 特有属性
    private String department;

    // 构造函数
    public Doctor(String name, int age, String department) {
        super(name, age);
        this.department = department;
    }

    // Getter 方法
    public String getDepartment() {
        return department;
    }

    // 覆盖父类方法
    @Override
    public void printInfo() {
        super.printInfo();
        System.out.println("科室: " + department);
    }
}
```

6.2、任务 2：方法的重载与覆盖

步骤：

(1) 在 `Doctor` 类中实现方法的重载（`diagnose` 方法）。

(2) 在 `Doctor` 类中覆盖父类的 `printInfo` 方法。

示例代码：

Java

```java
// 在 Doctor 类中添加以下方法
// 方法重载
public void diagnose(Patient patient) {
    System.out.println("医生 " + getName() + " 正在为患者 " + patient.getName() + " 进行诊疗。");
    System.out.println("患者症状: " + patient.getSymptom());
    System.out.println("建议前往 " + department + " 进一步检查。");
}

public void diagnose(Patient patient, String prescription) {
    System.out.println("医生 " + getName() + " 正在为患者 " + patient.getName() + " 进行诊疗。");
    System.out.println("患者症状: " + patient.getSymptom());
    System.out.println("开具处方: " + prescription);
}

// 测试类
public class DoctorTest {
    public static void main(String[] args) {
        Patient patient = new Patient("张三", 35, "发热");
        Doctor doctor = new Doctor("王医生", 40, "发热门诊");

        doctor.diagnose(patient);
        doctor.diagnose(patient, "退烧药");
    }
}
```

6.3、任务 3：对象引用转换

步骤：

(1) 实现父类与子类之间的对象引用转换。

(2) 使用 `instanceof` 关键字检查对象类型。

示例代码：

Java

```java
public class ObjectConversion {
    public static void main(String[] args) {
        // 父类引用指向子类对象
        Person person = new Patient("李四", 28, "咳嗽");

        // 向下转型
        if (person instanceof Patient) {
            Patient patient = (Patient) person;
            patient.printInfo();
        }

        // 父类引用指向另一个子类对象
        person = new Doctor("张医生", 45, "呼吸科");

        // 向下转型
        if (person instanceof Doctor) {
            Doctor doctor = (Doctor) person;
            doctor.printInfo();
        }
    }
}
```

6.4、任务 4：访问控制与 `final` 修饰符

步骤：

(1) 使用访问控制符（`private`、`protected`）限制类的成员访问。

(2) 使用 `final` 修饰符定义不可继承的类和方法。

Java

```java
// 使用 final 修饰符定义不可继承的类
public final class MedicalRecord {
    private String recordId;
    protected String diagnosis;

    public MedicalRecord(String recordId, String diagnosis) {
        this.recordId = recordId;
        this.diagnosis = diagnosis;
    }

    // final 方法，不可覆盖
    public final void printRecord() {
        System.out.println("病历号: " + recordId);
        System.out.println("诊断结果: " + diagnosis);
    }
}

// 测试类
public class AccessControlTest {
    public static void main(String[] args) {
        MedicalRecord record = new MedicalRecord("12345", "感冒");
        record.printRecord();
    }
}
```

6.5、任务 5：综合任务 - 医院管理系统

步骤：

(1) 设计一个 `Hospital` 类，管理患者和医生。

(2) 实现多态性的诊疗过程，通过父类引用调用子类方法。

示例代码：

Java

```java
import java.util.ArrayList;
import java.util.List;

// Hospital.java
public class Hospital {
    private List<Person> people;

    // 构造函数
    public Hospital() {
        people = new ArrayList<>();
    }

    // 添加人员
    public void addPerson(Person person) {
        people.add(person);
        System.out.println("添加人员: " + person.getName());
    }

    // 多态性诊疗过程
    public void processDiagnosis() {
        for (Person person : people) {
            if (person instanceof Doctor) {
                Doctor doctor = (Doctor) person;
                for (Person p : people) {
                    if (p instanceof Patient) {
                        Patient patient = (Patient) p;
                        doctor.diagnose(patient);
                    }
                }
            }
        }
    }

    // 测试类
    public static void main(String[] args) {
        Hospital hospital = new Hospital();

        // 添加患者和医生
        hospital.addPerson(new Patient("张三", 35, "发热"));
        hospital.addPerson(new Patient("李四", 28, "咳嗽"));
        hospital.addPerson(new Doctor("王医生", 40, "发热门诊"));
        hospital.addPerson(new Doctor("张医生", 45, "呼吸科"));
```

```
43
44          // 诊疗过程
45          hospital.processDiagnosis();
46      }
47  }
```

七、实验考核

本实验考核采用【实验随堂查】方式开展。

每个实验完成后，在实验课上通过现场演示的方式向实验指导教师进行汇报，并完成现场问答交流。

每个实验考核满分 100 分，其中实验成果汇报 60 分，现场提问交流 40 分。

实验考核流程：

（1）学生演示汇报实验内容的完成情况，实验指导老师现场打分。

（2）指导老师结合实验内容进行提问，每位学生提问 2-3 个问题，根据回答的情况现场打分。

（3）实验考核结束后，进行公布成绩。