**浙 江 农 林 大 学 数 计 学 院**

**JAVA程序设计实验报告**

**实 验 七: 多态、抽象类和接口**

**学生姓名: 刘糠杰**

**专　 业: 智科232班**

**指导教师: 戴丹**

**完成日期** （ 2024年 11月 29日）

1. **实验任务**

**## 第一题：**

设计一个图形类 Shape 和它的两个子类 Circle（圆形）和 Rectangle（矩形），它们都有一个 calculateArea() 方法用于计算自己的面积，并定义一个 printArea(Shape shape) 方法，用于输出图形的面积。在 main 方法中创建一个 Circle 对象和一个 Rectangle 对象，并分别调用 printArea() 方法输出它们的面积。

package hello;

public class Test {

    public static void main(String[] args) {

        Circle c = new Circle();

        Rectangle rectangle = new Rectangle();

        c.setR(6);

        rectangle.setHigth(4);

        rectangle.setWidth(6);

        c.calculateArea();

        rectangle.calculateArea();

        System.out.println("圆形的面积：");

        c.printArea(c);

        System.out.println("矩形的面积：");

        rectangle.printArea(rectangle);

    }

}

class Shape {

    protected double area;

    public void calculateArea() { }

    public void printArea(Shape shape) {

        System.out.println("这个图形的面积是" + shape.area);

    }

}

class Circle extends Shape {

    private double r;

    public void setR(double r) {

        this.r = r;

    }

    public double getR() {

        return this.r;

    }

    public void calculateArea() {

        this.area = r \* r \*  3.14;

    }

}

class Rectangle extends Shape {

    private double higth;

    private double width;

    public void setHigth(double higth) {

        this.higth = higth;

    }

    public double getHigth() {

        return this.higth;

    }

    public void setWidth(double width) {

        this.width = width;

    }

    public double getWidth() {

        return this.width;

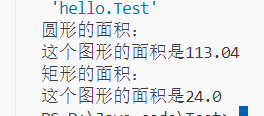
    }

    public void calculateArea() {

        this.area = this.higth \* this.width;

    }

}



**## 第二题：**

定义一个披萨店 PizzaStore，它有一个 order() 方法，输出 "订购了一份披萨"。定义两个子类 CheesePizzaStore 和 SeafoodPizzaStore，它们都重写了 order() 方法，分别输出 "订购了一份芝士披萨"和 "订购了一份海鲜披萨"。在 main 方法中创建一个 CheesePizzaStore 对象和一个 SeafoodPizzaStore 对象，并调用它们的 order() 方法。

package hello;

public class Test {

    public static void main(String[] args) {

        PizzaStore cheesePizzaStore = new CheesePizzaStore();

        PizzaStore seafoodPizzaStore = new SeafoodPizzaStore();

        cheesePizzaStore.order();

        seafoodPizzaStore.order();

    }

}

class PizzaStore {

    public void order() {

        System.out.println("订购了一份披萨");

    }

}

class CheesePizzaStore extends PizzaStore {

    @Override

    public void order() {

        System.out.println("订购了一份芝士披萨");

    }

}

class SeafoodPizzaStore extends PizzaStore {

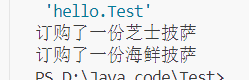
    @Override

    public void order() {

        System.out.println("订购了一份海鲜披萨");

    }

}



**## 第三题：**

1. 定义手机类

行为：打电话,发短信

2. 定义接口IPlay

行为：玩游戏

3. 定义旧手机类继承手机类

行为：继承父类的行为

4. 定义新手机继承手机类实现IPlay接口

行为：继承父类的行为,重写玩游戏方法

5. 定义测试类

在测试类中定义一个 用手机的方法,要求该方法既能接收老手机对象,也能接收新手机对象

在该方法内部调用打电话,发短信以及新手机特有的玩游戏方法

package hello;

public class Test {

    public static void main(String[] args) {

        NewPhone newPhone = new NewPhone();

        Test.use\_phone(newPhone);

    }

    public static void use\_phone(Phone phone) {

        phone.call();

        phone.sendMessage();

        if(phone instanceof NewPhone) {

           ((NewPhone)phone).play\_game();

        }

    }

}

interface IPlay {

    public void play\_game();

}

class Phone {

    public void call() {

        System.out.println("打电话");

    }

    public void sendMessage() {

        System.out.println("发短信");

    }

}

class OldPhone extends Phone{}

class NewPhone extends Phone implements IPlay {

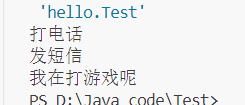
    @Override

    public void play\_game() {

        System.out.println("我在打游戏呢");

    }

}



**## 第四题：**

使用抽象类和抽象方法构建不同动物的扑食行为，要求如下：

1. 不同的动物都有进食的行为，但是在进食前需要捕获到食物后才可以进食。
2. 要求定义一个抽象的动物类，该类中有一个抽象的捕食方法，和一个非抽象的进食方法。在进食方法中调用捕食方法。
3. 定义3个不同的动物类，分别是老虎、猴子和狼，这些类继承抽象的动物类。

package hello;

public class Test {

    public static void main(String[] args) {

    }

}

abstract class Animal{

    public abstract void hunt();

    public void eat() {

        hunt();

        System.out.println("我要开吃了");

    }

}

class Tiger extends Animal {

    @Override

    public void hunt() {

        System.out.println("老虎捕食");

    }

}

class Monkey extends Animal {

    @Override

    public void hunt() {

        System.out.println("猴子偷桃");

    }

}

class Wolf extends Animal {

    @Override

    public void hunt() {

        System.out.println("狼群集结");

    }

}

**## 第五题：**

编写带有接口和抽象类的标准JAVABEAN类，满足以下条件：

我们现在有乒乓球运动员和篮球运动员，乒乓球教练和篮球教练。为了出国交流，跟乒乓球相关的人员都需要学习英语。这些对象具有的属性和行为如下：

1. 乒乓球运动员：姓名，年龄，学打乒乓，说英语
2. 篮球运动员：姓名，年龄，学打篮球
3. 乒乓球教练：姓名，年龄，教打乒乓，说英语
4. 篮球教练：姓名，年龄，教打篮球

package hello;

public class Test {

    public static void main(String[] args) {

        PingPongPlayer ppPlayer = new PingPongPlayer();

        ppPlayer.setName("张三");

        ppPlayer.setAge(20);

        System.out.println(ppPlayer.getName() + "，年龄：" + ppPlayer.getAge());

        ppPlayer.career();

        ppPlayer.SpeakEnglish();

        PingPongCoach ppCoach = new PingPongCoach();

        ppCoach.setName("李四");

        ppCoach.setAge(35);

        System.out.println(ppCoach.getName() + "，年龄：" + ppCoach.getAge());

        ppCoach.career();

        ppCoach.SpeakEnglish();

        BasketballPlayer bbPlayer = new BasketballPlayer();

        bbPlayer.setName("王五");

        bbPlayer.setAge(22);

        System.out.println(bbPlayer.getName() + "，年龄：" + bbPlayer.getAge());

        bbPlayer.career();

        BasketballCoach bbCoach = new BasketballCoach();

        bbCoach.setName("赵六");

        bbCoach.setAge(40);

        System.out.println(bbCoach.getName() + "，年龄：" + bbCoach.getAge());

        bbCoach.career();

    }

}

interface Language {

    void SpeakEnglish();

}

abstract class Person {

    private String name;

    private int age;

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public int getAge() {

        return age;

    }

    public void setAge(int age) {

        this.age = age;

    }

    public abstract void career();

}

class PingPongPlayer extends Person implements Language {

    @Override

    public void SpeakEnglish() {

        System.out.println("我会说英语");

    }

    @Override

    public void career() {

        System.out.println("打乒乓球");

    }

}

class PingPongCoach extends Person implements Language {

    @Override

    public void SpeakEnglish() {

        System.out.println("我会说英语");

    }

    @Override

    public void career() {

        System.out.println("教乒乓球");

    }

}

class BasketballPlayer extends Person {

    @Override

    public void career() {

        System.out.println("打篮球");

    }

}

class BasketballCoach extends Person {

    @Override

    public void career() {

        System.out.println("教打篮球");

    }

}