

day03课后练习

基础题目

第一题：概念辨析

1. 什么叫做多态，条件是什么？

- * 一类事物的多种表现形式
- * 条件：
 - * 继承或者实现【二选一】
 - * 方法的重写

2. 使用多态特性，带来了什么样的好处？

提高代码的扩展性

3. 使用多态特性，注意什么样的弊端？

不能调用子类特有方法，必须向下转型。

第二题：语法练习

- 语法点：匿名内部类使用
- 根据需求，完成如下代码，并在测试类中进行测试。

员工类 Employee：
属性：姓名 工号 工资
抽象方法：工作(work)，开会(meet)

使用匿名内部类方式创建一个员工类,并调用其工作和开会的功能

参考答案：

Employee.java

```
public abstract class Employee {  
    // 属性：姓名 工号 工资  
    private String name;  
    private String id;  
    private String salary;  
  
    public Employee() {  
    }  
}
```



```
public Employee(String name, String id, String salary) {
    this.name = name;
    this.id = id;
    this.salary = salary;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public String getSalary() {
    return salary;
}

public void setSalary(String salary) {
    this.salary = salary;
}

// 抽象方法：工作work，开会meet
public abstract void work();
public abstract void meet();
}
```

Test03.java

```
public class Test03 {
    public static void main(String[] args) {
        // 创建员工子类对象
        Employee e = new Employee() {
            @Override
            public void work() {
                System.out.println("工作方法");
            }

            @Override
            public void meet() {
                System.out.println("开会方法");
            }
        };
        e.work();
        e.meet();
    }
}
```



```
}  
}
```

第三题：语法练习

- 语法点：匿名内部类使用
- 请编写一个接口Phone,两个抽象方法,call() sendMessage() 使用匿名内部类的方式调用 call 和 sendMessage方法

参考答案:

Phone.java

```
public interface Phone {  
    // 抽象方法  
    public abstract void call();  
    public abstract void sendMessage();  
}
```

Test03.java

```
public class Test03 {  
    public static void main(String[] args) {  
        Phone p = new Phone() {  
            @Override  
            public void sendMessage() {  
                System.out.println("发短信");  
            }  
  
            @Override  
            public void call() {  
                System.out.println("打电话");  
            }  
        };  
  
        p.call();  
        p.sendMessage();  
    }  
}
```

第四题：语法练习

- 语法点：匿名内部类使用
- 请编写一个接口 Player,包含 playBasketball,playPingpong 2 个抽象方法,在测试类中使用匿名 内部类方式创建对象,并调用这 2 个功能
- 参考答案:

Player.java

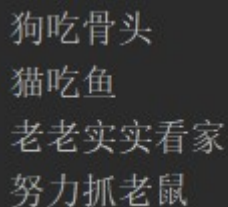
```
public interface Player {  
    public void playBasketball();  
    public void playPingpong();  
}
```

Test03.java

```
public class Test03 {  
    public static void main(String[] args) {  
        Player player = new Player() {  
            @Override  
            public void playPingpong() {  
                System.out.println("打乒乓球");  
            }  
  
            @Override  
            public void playBasketball() {  
                System.out.println("打篮球");  
            }  
        };  
  
        player.playBasketball();  
        player.playPingpong();  
    }  
}
```

第五题：语法练习

- 语法点：接口，多态
- 定义一个父类Animal 包含name,weight属性和一个抽象的eat方法, 定义两个子类Dog和Cat,Dog特有方法lookHome,Cat特有方法catchMouse;并且重写eat方法,Dog吃骨头,Cat吃鱼 要求:使用多态形式创建Dog和Cat对象,调用eat方法,并且使用向下转型,如果是Cat类型调用catchMouse功能,如果是Dog类型调用lookHome功能
- 按步骤编写代码，效果如图所示：



狗吃骨头
猫吃鱼
老老实实看家
努力抓老鼠

- 编写步骤
 1. 定义抽象类Animal
 2. 在抽象类Animal中包含name,weight属性和一个抽象的eat方法
 3. 定义Cat类继承Animal类
 4. 在Cat类中重写eat方法
 5. 在Cat类中定义catchMouse方法
 6. 定义Dog类继承Animal类



7. 在Dog类中重写eat方法
8. 在Dog类中定义lookHome方法
9. 使用多态创建狗对象d
10. 使用多态创建猫对象c
11. 调用d对象的eat方法
12. 调用c对象的eat方法
13. 使用instanceof判断d对象是否是Dog类
14. 如果d对象是Dog类,将d对象向下转型为Dog类型,并调用lookHome方法
15. 使用instanceof判断c对象是否是Cat类
16. 如果c对象是Cat类,将c对象向下转型为Cat类型,并调用catchMouse方法

• 参考答案:

```
public class Test5 {  
    public static void main(String[] args) {  
        // 9.使用多态创建狗对象d  
        Animal d = new Dog("大黄", 30);  
        // 10.使用多态创建猫对象c  
        Animal c = new Cat("加菲猫", 15);  
        // 11.调用d对象的eat方法  
        d.eat();  
        // 12.调用c对象的eat方法  
        c.eat();  
  
        // 13.使用instanceof判断d对象是否是Dog类  
        if (d instanceof Dog) {  
            // 14.如果d对象是Dog类,将d对象向下转型为Dog类型,并调用lookHome方法  
            Dog dog = (Dog)d;  
            dog.lookHome();  
        }  
  
        // 15.使用instanceof判断c对象是否是Cat类  
        if (c instanceof Cat) {  
            // 16.如果c对象是Cat类,将c对象向下转型为Cat类型,并调用catchMouse方法  
            Cat cat = (Cat)c;  
            cat.catchMouse();  
        }  
    }  
}  
  
// 1.定义抽象类Animal  
abstract class Animal {  
    // 2.在抽象类Animal中包含name,weight属性和一个抽象的eat方法  
    // 姓名  
    private String name;  
    // 体重  
    private int weight;  
  
    public Animal() {  
    }  
  
    public Animal(String name, int weight) {  
        this.name = name;  
    }  
}
```



```
        this.weight = weight;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getWeight() {
        return weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

    // 吃方法
    public abstract void eat();
}

// 3.定义Cat类继承Animal类
class Cat extends Animal {
    public Cat() {
        super();
    }

    public Cat(String name, int weight) {
        super(name, weight);
    }

    // 4.在Cat类中重写eat方法
    @Override
    public void eat() {
        System.out.println("猫吃鱼");
    }

    // 5.在Cat类中定义catchMouse方法
    public void catchMouse() {
        System.out.println("努力抓老鼠");
    }
}

// 6.定义Dog类继承Animal类
class Dog extends Animal {
    public Dog() {
        super();
    }

    public Dog(String name, int weight) {
        super(name, weight);
    }
}
```

```
}

// 7.在Dog类中重写eat方法
@Override
public void eat() {
    System.out.println("狗吃骨头");
}

// 8.在Dog类中定义lookHome方法
public void lookHome() {
    System.out.println("老老实实看家");
}
}
```

扩展题目

第六题：需求实现

- 请使用代码描述:

学生都有年龄和姓名属性,有吃饭(学生餐)和学习方法,但是有部分学生会打篮球
老师都有年龄和姓名属性,有吃饭(工作餐)和讲课方法,但是有部分老师会打篮球
定义一个方法模拟去打篮球,只要会打篮球的人都可以传入。(提示通过在测试类中定义一个方法参数为接口)

- 代码实现,效果如图所示:

年龄为35岁 大姚 的老师在打篮球
年龄为21岁 王中王 的学生在打篮球

- 编写步骤

1. 定义Person类
2. Person类包含name,age属性和抽象的eat方法
3. 定义Sport接口,包含playBasketball方法
4. 定义Teacher类继承Person类,重写抽象方法eat()
5. 定义SportTeacher类继承Teacher类,实现Sport接口,重写Sport接口中的playBasketball方法
6. 定义Student类继承Person类,重写抽象方法eat()
7. 定义SportStudent类继承Student类,实现Sport接口,重写Sport接口中的playBasketball方法
8. 在测试类中定义静态的goToSport方法,参数为Sport接口类型
9. 在main方法中创建普通的老师t1,姓名为马云,年龄为45岁
10. 在main方法中创建会打篮球的老师t2,姓名为大姚,年龄为35岁
11. 在main方法中创建普通的学生s1,姓名为小王,年龄为20
12. 在main方法中创建会打篮球的学生s2,姓名为王中王,年龄为21
13. 在main方法中调用goToSport方法.传入t1,t2,s1,s2四个对象.我们会发现只有实现Sport接口的对象才能传入

- 参考答案:

```
public class Test6 {
    public static void main(String[] args) {
        // 9.在main方法中创建普通的老师t1,姓名为马云,年龄为45岁
```



```
Teacher t1 = new Teacher("马云", 45);
// 10.在main方法中创建会打篮球的老师t2,姓名为大姚,年龄为35岁
SportTeacher t2 = new SportTeacher("大姚", 35);

// 11.在main方法中创建普通的学生s1,姓名为小王,年龄为20
Student s1 = new Student("小王", 20);
// 12.在main方法中创建会打篮球的学生s2,姓名为王中王,年龄为21
SportStudent s2 = new SportStudent("王中王", 21);

// 13.在main方法中调用goToSport方法.传入t1,t2,s1,s2四个对象.我们会发现只有实现Sport接口的
对象才能传入
// goToSport(t1); // 没有实现Sport接口不能传入
goToSport(t2);
// goToSport(s1); // 没有实现Sport接口不能传入
goToSport(s2);
}

// 8.在测试类中定义静态的goToSport方法,参数为Sport接口类型
public static void goToSport(Sport s){
    // 在goToSport方法中调用传入参数的playBasketball方法
    s.playBasketball();
}
}

// 1.定义Person类
abstract class Person {
    // 2.Person类包含name,age属性和抽象的eat方法
    private String name;
    private int age;

    public abstract void eat();

    public Person() {
        super();
    }

    public Person(String name, int age) {
        super();
        this.name = name;
        this.age = 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;
        }
    }

// 3.定义Sport接口,包含playBasketball方法
interface Sport {
    public abstract void playBasketball();
}

// 4.定义Teacher类继承Person类,重写抽象方法eat()
class Teacher extends Person {
    public void eat() {
        System.out.println();
    }

    public Teacher() {
        super();
    }

    public Teacher(String name, int age) {
        super(name, age);
    }
}

// 5.定义SportTeacher类继承Teacher类,实现Sport接口,重写Sport接口中的playBasketball方法
class SportTeacher extends Teacher implements Sport {
    public void playBasketball() {
        System.out.println("年龄为" + getAge() + "岁 " + getName() + " 的老师在打篮球");
    }

    public SportTeacher() {
        super();
    }

    public SportTeacher(String name, int age) {
        super(name, age);
    }
}

// 6.定义Student类继承Person类,重写抽象方法eat()
class Student extends Person {
    public void eat() {
        System.out.println("年龄" + getAge() + "岁的 " + getName() + " 在吃学生餐");
    }

    public Student() {
        super();
    }

    public Student(String name, int age) {
        super(name, age);
    }
}
```



```
}  
}  
  
// 7.定义SportStudent类继承Student类,实现Sport接口,重写Sport接口中的playBasketball方法  
class SportStudent extends Student implements Sport {  
    public SportStudent() {  
    }  
  
    public SportStudent(String name, int age) {  
        super(name, age);  
    }  
  
    public void playBasketball() {  
        System.out.println("年龄为" + getAge() + "岁 " + getName() + " 的学生在打篮球");  
    }  
}
```

第七题：需求实现

- 模拟公司给员工发工资
- 代码实现，效果如图所示：

```
给张小强发工资 9000.0 元,公司剩余: 991000.0 元  
给李小亮发工资 5000.0 元,公司剩余: 986000.0 元
```

- 开发提示：
 1. 定义员工Employee类。包含属性：姓名，薪资
 2. 定义经理Manager类继承Employee类
 3. 定义程序员Coder类继承Employee类
 4. 定义Money接口，提供抽象方法paySalary,参数为Employee
 5. 定义公司Company类，实现Money接口,Company类包含公司总资金属性
 6. 定义测试类，创建Company对象，Manager对象，Coder对象，调用公司paySalary方法，给Manager和Coder发工资
- 编写步骤
 1. 定义Employee类,包含属性：姓名，薪资
 2. 定义经理Manager类继承Employee类
 3. 定义程序员Coder类继承Employee类
 4. 定义Money接口包含抽象的paySalary方法,参数为(Employee emp)
 5. 定义Company类,实现Money接口,Company类包含公司总资金属性
 6. 在Company类中重写paySalary方法.当给一个员工发工资的时候.公司总资金减去已发工资
 7. 在main方法中创建Manager对象m
 8. 在main方法中创建Coder对象c
 9. 在main方法中创建Company对象
 10. 在main方法中调用Company的paySalary方法,传入m和c对象
- 参考答案：

```
public class Test7 {  
    public static void main(String[] args) {  
        // 7.在main方法中创建Manager对象m
```



```
        Manager m = new Manager("M001", "张小强", 9000);
        // 8.在main方法中创建Coder对象c
        Coder c = new Coder("C001", "李小亮", 5000);

        // 9.在main方法中创建Company对象
        Company company = new Company(1000000);

        // 10.在main方法中调用Company的paySalary方法,传入m和c对象
        company.paySalary(m);
        company.paySalary(c);
    }
}

// 4.定义Money接口包含抽象的paySalary方法,参数为(Employee emp)
interface Money {
    public abstract void paySalary(Employee emp);
}

// 5.定义Company类,实现Money接口,Company类包含公司总资金属性
class Company implements Money {
    private double totalMoney;

    public Company() {
    }

    public Company(double totalMoney) {
        this.totalMoney = totalMoney;
    }

    // 6.在Company类中重写paySalary方法.当给一个员工发工资的时候.公司总资金减去已发工资
    @Override
    public void paySalary(Employee emp) {
        totalMoney -= emp.getSalary();
        System.out.println("给" + emp.getName() + "发工资 " + emp.getSalary() + " 元,公司剩
余: " + totalMoney + " 元");
    }

    public double getTotalMoney() {
        return totalMoney;
    }

    public void setTotalMoney(double totalMoney) {
        this.totalMoney = totalMoney;
    }
}

// 1.定义Employee类,包含属性: 姓名, 薪资
class Employee {
    private String id;
    private String name;
    private double salary;

    public double getSalary() {
```



```
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public Employee() {
        super();
    }

    public Employee(String id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

// 2.定义经理Manager类继承Employee类
class Manager extends Employee {
    public Manager(String id, String name, double salary) {
        super(id, name, salary);
    }
}

// 3.定义程序员Coder类继承Employee类
class Coder extends Employee {
    public Coder(String id, String name, double salary) {
        super(id, name, salary);
    }
}
```

第八题：需求实现

- 模拟各种商品添加到购物车，电子商品打8.8折优惠，展示所有商品信息。
- 代码实现，效果如图所示：



```
=====添加商品=====
加入 笔记本 成功
加入 手机 成功
加入 苹果 成功
=====打印商品=====
您选购的商品为:
      g10000, 笔记本, 10000.0
      g10001, 手机, 5000.0
      g20000, 苹果, 50.0
-----
原  价为:15050.0 元
折后价为:13250.0 元
```

- 开发提示:

- 定义购物车类。
 - 使用ArrayList作为成员变量，保存各种商品对象。
 - 提供添加商品，移除商品，计算总价的方法。
- 定义商品类Goods，包含商品名称，id，价格等属性。
- 定义电子商品类EGoods继承Goods。
- 定义笔记本类Laptop继承电子商品EGoods类。
- 定义手机类继承继承电子商品EGoods类类。
- 定义水果类Fruit继承商品类。

- 编写步骤

1. 定义Goods商品类,包含商品编号id,商品名称name,商品价格price属性
2. 定义EGoods继承Goods类
3. 定义Phone继承EGoods类
4. 定义Laptop继承EGoods类
5. 定义Fruit继承Goods类
6. 定义购物车类GouWuChe
7. 在购物车类GouWuChe中定义ArrayList成员变量,用于保存购物车中的商品
8. 在购物车类GouWuChe中定义addGoods方法,参数为(Goods goods).addGoods方法功能是将商品保存到ArrayList集合中
9. 在购物车类GouWuChe中定义showGoods方法.showGoods方法功能是遍历ArrayList集合中的所有商品信息并打印
10. 在购物车类GouWuChe中定义total方法.total方法功能是计算ArrayList集合中的所有商品的总价和折后价格,并输出
11. 在main方法中创建GouWuChe对象gouWuChe
12. 在main方法中创建商品Laptop,名称为:笔记本,id为:g10000,价格为:10000
13. 在main方法中创建商品Phone,名称为:手机,id为:g10001,价格为:5000
14. 在main方法中创建商品Fruit,名称为:苹果,id为:g20000,价格为:50
15. 调用购物车的addGoods方法将3个商品添加到购物车中
16. 调用购物车的showGoods方法,显示购物车中的商品信息



17. 调用购物车的total方法,显示购物车中所有商品的价格

- 参考答案:

```
public class Test8 {  
    public static void main(String[] args) {  
  
        // 11.在main方法中创建GouwuChe对象gouwuChe  
        GouwuChe gouwuChe = new GouwuChe();  
  
        // 12.在main方法中创建商品Laptop,名称为:笔记本,id为:g10000,价格为:10000  
        Goods g1 = new Laptop("笔记本", "g10000", 10000);  
        // 13.在main方法中创建商品Phone,名称为:手机,id为:g10001,价格为:5000  
        Goods g2 = new Phone("手机", "g10001", 5000);  
        // 14.在main方法中创建商品Fruit,名称为:苹果,id为:g20000,价格为:50  
        Goods g3 = new Fruit("苹果", "g20000", 50);  
  
        System.out.println("=====添加商品=====");  
        // 15.调用购物车的addGoods方法将3个商品添加到购物车中  
        gouwuChe.addGoods(g1);  
        gouwuChe.addGoods(g2);  
        gouwuChe.addGoods(g3);  
        System.out.println("=====打印商品=====");  
        // 16.调用购物车的showGoods方法,显示购物车中的商品信息  
        gouwuChe.showGoods();  
  
        // 17.调用购物车的total方法,显示购物车中所有商品的价格  
        gouwuChe.total();  
    }  
}  
  
// 6.定义购物车类GouwuChe  
class GouwuChe {  
    // 7.在购物车类GouwuChe中定义ArrayList成员变量,用于保存购物车中的商品  
    ArrayList<Goods> list = new ArrayList<>();  
  
    public GouwuChe() {  
    }  
  
    // 8.在购物车类GouwuChe中定义addGoods方法,参数为(Goods goods).addGoods方法功能是将商品保存到ArrayList集合中  
    public void addGoods(Goods goods) {  
        System.out.println("加入 " + goods.getName() + " 成功");  
        list.add(goods);  
    }  
  
    // 9.在购物车类GouwuChe中定义showGoods方法.showGoods方法功能是遍历ArrayList集合中的所有商品信息并打印  
    public void showGoods() {  
        System.out.println("您选购的商品为:");  
        for (int i = 0; i < list.size(); i++) {  
            Goods goods = list.get(i);  
            System.out.println("\t" + goods.getId() + "," + goods.getName() + "," +  
goods.getPrice());  
        }  
    }  
}
```



```
    }  
}  
  
// 10.在购物车类Gouwuche中定义total方法.total方法功能是计算ArrayList集合中的所有商品的总价和  
折后价格,并输出  
public void total() {  
    double off = 0; // 折扣价  
    double sum = 0; // 原价  
  
    for (int i = 0; i < list.size(); i++) {  
        Goods goods = list.get(i);  
        double price = goods.getPrice();  
        sum += price;  
        // 如果商品为电子产品,就打折计算  
        if (goods instanceof EGoods) {  
            price *= 0.88;  
        }  
  
        off += price;  
    }  
    System.out.println("-----");  
    System.out.println("原  价为:" + sum + " 元");  
    System.out.println("折后价为:" + off + " 元");  
}  
}  
  
// 5.定义Fruit继承Goods类  
class Fruit extends Goods {  
    public Fruit(String name, String id, double price) {  
        super(name, id, price);  
    }  
}  
  
// 4.定义Laptop继承EGoods类  
class Laptop extends EGoods {  
    public Laptop(String name, String id, double price) {  
        super(name, id, price);  
    }  
}  
  
// 3.定义Phone继承EGoods类  
class Phone extends EGoods {  
    public Phone(String name, String id, double price) {  
        super(name, id, price);  
    }  
}  
  
// 2.定义EGoods继承Goods类  
class EGoods extends Goods {  
    public EGoods(String name, String id, double price) {  
        super(name, id, price);  
    }  
}
```



```
// 1.定义Goods商品类,包含商品编号id,商品名称name,商品价格price属性
class Goods {
    private String name;
    private String id;
    private double price;

    public Goods() {
    }

    public Goods(String name, String id, double price) {
        this.name = name;
        this.id = id;
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
}
```