后端

java06

task1

这是做完后来再补上的前言,java06相比前面难上好多……三个task的代码时逐渐完善的,所以最终上传的只有一个代码,文件名为Java06答题代码

Q1&Q2:

> 1. 请你了解Java的继承机制，编写Dish作为基类，包含上面类的属性和方法，然后编写两个类Dish_1和Dish_2继承这个基类，并在菜品类的构造方法里把菜品名字和价格写好（菜品名字和价格自行决定）。

> 2. 显然不同的菜品介绍是不一样的，也就是说调用该菜品类的profile方法时，应该输出符合的介绍，请做到这一点。

这个查查extend的用法之后不是很难,先写,看看会遇到什么问题吧(^_^)

(代码文件已上传github,文件名为Java-06答题代码)

```
package com.ISEKAI;

public class jicheng {
    public static class Dish{
        //基类
        private String name;
        private double price;

        public void profile(){
            System.out.println("番茄炒蛋是普通的大众菜肴，烹调方法比较简单，而且营养搭配合理。色泽鲜艳，口味宜人，深受大众喜爱。其营养价值丰富，具有营养素互补的特点以及健美抗衰老的作用。番茄含有丰富的胡萝卜素、维生素C和B族维生素，番茄红素具有独特的抗氧化能力。鸡蛋含有大量的维生素和矿物质及有高生物价的蛋白质。蛋黄中含有丰富的卵磷脂、固醇类等，对神经系统和身体发育有非常好的作用，深受人们的喜爱。");
        }

        Dish(String name,double price){
            this.name=name;
            this.price=price;
        }
    }

    public static class Dish1 extends Dish {
//菜品一
        public void profile(){
            System.out.println("康师傅大食物桶让你拜托方便面一碗吃不饱，两碗吃不了的烦恼，是做微光招新题时足不出户就能让你填饱肚子的家具必备单单品");
        }
        Dish1() {
           super("康师傅老坛酸菜大食桶",200);
        }
    }
```

```
    public static class Dish2 extends Dish{
//菜品二
        public void profile(){
            System.out.println("德芙纵享丝滑");
        }

        Dish2() {
            super("德芙小清新柠檬曲奇巧克力",300 );
        }
    }

    public static void main(String[] args) {
        Dish1 dish1 = new Dish1();
        dish1.profile();
        Dish2 dish2 = new Dish2();
        dish2.profile();
    }
}
```

接下来是我做题过程中的一些想法:

1. 基于为什么要继承,一开始我想既然有基类Dish了,直接用基类声明多个对象(dish1 dish2)。对于菜品描述,把它也附加到基类要初始化的成员变量里应该也是可以实现功能的。但是遇到一定要在子类中特殊定义成员变量和函数的时候应该还是只能用extends了。
2. 基于子类的初始化方法,按题目的意思应该是我在子类的初始化函数里就将信息输入,而不是在使用初始化函数的时候进行输入。

   当然两种应该都可以,不过要是这个子类被外部调用的话还是第一种方法更方便。

task2

Q3&4:

> 请你了解接口相关的知识，创建一个名为Order的接口，里面有cook和check两个方法，cook输出该菜品的烹饪方法，check随机返回true（表示原料足够可以烹饪该菜品）或者false（表示不能烹饪该菜品），模拟实际情况中可能某些菜品原料不够的情况。让你的菜品类继承这个接口，并完成厨师类System的manageOrder函数。

> 我们都知道面向对象的三大特性：封装继承多态，其实接口就是多态的一种实现机制，厨师调用的明明是Order类的方法，实际执行的却是Dish_1或者Dish_2的方法

查了很多资料,感觉不太好理解,感觉网上的文章都没有告诉我接口到底在解决一个问题的时候发挥的作用是什么

所以我边猜边试的敲代码咯(过程中的问题附于代码后)(代码已上传github)

```
package com.ISEKAI;
import java.util.Random;
import java.util.List;
import java.util.ArrayList;

public class jicheng  {
    //接口,Order
    interface Order {
        void cook();
        boolean check();
```

```
        String nameout();

    }
    //Dish基类
    public static class Dish implements Order {
        private String name;
        private double price;

        @Override
        public void cook() {

        }

        @Override
        public boolean check() {
            return new Random().nextBoolean();
        }

        @Override
        public String nameout() {
            return name;
        }

        Dish(String name, double price) {
            this.name = name;
            this.price = price;
        }

    }
//Dish1
    public static class Dish1 extends Dish {
        public void profile() {
            System.out.println("康师傅大食物桶让你拜托方便面一碗吃不饱，两碗吃不了的烦恼，是做微光招新题时足不出户就能让你填饱肚子的家具必备单单品");
        }

        Dish1() {
            super("康师傅老坛酸菜大食桶", 200);
        }

        @Override
        public void cook() {
            System.out.println("将泡面饼与调料一并放入口腔中，然后倒入开水至淹没面饼，等待3分钟后直接食用");
        }

    }
//Dish2
    public static class Dish2 extends Dish{
        public void profile() {
            System.out.println("德芙纵享丝滑");
        }

        Dish2() {
            super("德芙小清新柠檬曲奇巧克力", 300);
        }
```

```
        @Override
        public void cook() {
            System.out.println("必须在长椅上制作，制作时请扶好扶手，太丝滑可能导致不必要的
碰撞");
        }

    }
    //厨师类
    public static class system {
        private static int Ordernum=1;
        //请补全处理订单的函数
        public void manageOrder(List<Order> dishes) {
            boolean canIcook=true;
            for (Order dish:dishes) {
                if(!dish.check()){
                    canIcook=false;
                    break;
                }
            }
            if(canIcook){
                for(Order dish:dishes){
                    //输出做好的菜名和对应做法
                    System.out.println(dish.nameout());
                    dish.cook();
                }
                Ordernum++;
                System.out.println("订单"+Ordernum+"已完成");
            }
            else{
                System.out.println("取消订单");
                Ordernum++;
            }
            //要求1：一旦订单里有一个菜品的原料不足以烹饪，就输出“取消订单”，否则输出所有菜品
的烹饪方法，最后再输出该订单的编号，编号从1开始递增。
        }
    }
    //测试代码
    public static void main(String[] args) {
        Dish1 dish1=new Dish1();
        Dish2 dish2=new Dish2();
        system sys=new system();
        System.out.println("菜品列表");
        System.out.print(dish1.nameout()+":");
        dish1.profile();
        System.out.print(dish2.nameout()+":");
        dish2.profile();
        List<Order> list1=new ArrayList<>();
        list1.add(dish1);
        list1.add(dish2);
        sys.manageOrder(list1);
    }
}
```

1. **有一个小插曲**,由于定义接口的时候把order打成了oerder就一直报错,我又重新定义了一个类叫order来存订单,幸好发现了错误(lll¬ω¬)后来改回去了

2. 一开始我遍历list是用的for（int i=0；……）的形式，后来找到了一个更加简便的方式，那就是for(Oreder dish:dishes)，它是什么意思，后来查到是将dishes中每个元素赋值给dish进行循环遍历(神奇!)
3. 为什么我查到的代码class类都是不加static的但是我每个都加了?(再去看static的用法)越看越迷糊了,变成历史遗留问题吧,en
4. 最后修改代码的时候,我想通过Order来调用dish1中的name失败了,然后选在再定义一个nameout函数来输出菜名,有什么办法可以直接调用吗?(我没查到,还是根本没有这种办法)

task3

Q5:

查了一下java泛型,就是template<>是吧,那我懂了

(可以判断类型的函数叫instanceof,我一直在想怎么泛型进去顾客的类型还能引用对应顾客类型的参数,怎么通过顾客类型的不同进行代码的ifelse,还是得问问ai才行啊!)

三个task后最终代码如下:

```
package com.ISEKAI;
import java.util.Iterator;
import java.util.Random;
import java.util.List;
import java.util.ArrayList;

public class jicheng  {
    //接口,Order
    interface Order {
        void cook();
        boolean check();
        String nameout();

    }
    //Dish基类
    public static class Dish implements Order {
        private String name;
        private double price;

        @Override
        public void cook() {

        }

        @Override
        public boolean check() {
            return new Random().nextBoolean();
        }

        @Override
        public String nameout() {
            return name;
        }

        Dish(String name, double price) {
            this.name = name;
            this.price = price;
        }
```

```
    }
//Dish1
    public static class Dish1 extends Dish {
        public void profile() {
            System.out.println("康师傅大食物桶让你拜托方便面一碗吃不饱，两碗吃不了的烦恼，是做微光招新题时足不出户就能让你填饱肚子的家具必备单单品");
        }

        Dish1() {
            super("康师傅老坛酸菜大食桶", 200);
        }

        @Override
        public void cook() {
            System.out.println("将泡面饼与调料一并放入口腔中，然后倒入开水至淹没面饼，等待3分钟后直接食用");
        }

    }
//Dish2
    public static class Dish2 extends Dish{
        public void profile() {
            System.out.println("德芙纵享丝滑");
        }

        Dish2() {
            super("德芙小清新柠檬曲奇巧克力", 300);
        }


        @Override
        public void cook() {
            System.out.println("必须在长椅上制作，制作时请扶好扶手，太丝滑可能导致不必要的碰撞");
        }

    }

    //顾客类
        public static class TableCustomer {
            public int tableId;
            private TableCustomer(int tableId){
                this.tableId=tableId;
            }
        }
        public static class WechatCustomer {
            public String address;//顾客地址
            public boolean takeout;//true代表该顾客是外卖，false代表该顾客是堂食
            private WechatCustomer(String address,boolean takeout){
                this.address=address;
                this.takeout=takeout;
            }
        }

    //厨师类
    public static class system<Type> {
        private static int Ordernum=1;
```

```
        //请补全处理订单的函数
        public void manageOrder(List<Order> dishes,Type custom) {
            boolean canIcook=true;
            for (Order dish:dishes) {
                if(!dish.check()){
                    canIcook=false;
                    break;
                }
            }
            if(canIcook){
                System.out.println(Ordernum+"号顾客所需菜品如下");
                for(Order dish:dishes){
                    //输出做好的菜名和对应做法
                    System.out.println(dish.nameout());
                    System.out.print("制作方法：");
                    dish.cook();
                }
                System.out.println("订单"+Ordernum+"已完成");
                Ordernum++;
                if(custom instanceof TableCustomer){
                    TableCustomer tableCustomer=(TableCustomer) custom;
                    System.out.println("送餐至"+tableCustomer.tableId+"桌");
                }
                else{
                    WechatCustomer wechatcustomer=(WechatCustomer) custom;
                    if(wechatcustomer.takeout){
                        System.out.println("送餐至"+wechatcustomer.address);
                    }
                    else{
                        System.out.println("顾客到店领取");
                    }
                }
            }
            else{
                System.out.println(Ordernum+"号顾客取消订单");
                Ordernum++;
            }
            //要求1：一旦订单里有一个菜品的原料不足以烹饪，就输出“取消订单”，否则输出所有菜品的烹饪方法，最后再输出该订单的编号，编号从1开始递增。
        }
    }
    //测试代码
    public static void main(String[] args) {
        //菜单初始化
        Dish1 dish1=new Dish1();
        Dish2 dish2=new Dish2();
        //两个系统平台初始化
        system<TableCustomer> sys1=new system<>();
        system<WechatCustomer> sys2=new system<>();
        //三个顾客初始化(懒得手动输入，就直接初始化了，嗯)
        TableCustomer one=new TableCustomer(610);
        WechatCustomer two=new WechatCustomer("电子科技大学欣苑学生宿舍",true);
        WechatCustomer three=new WechatCustomer("电子科技大学欣苑学生宿舍",false);
        //点餐与准备
        System.out.println("菜品列表");
        System.out.print(dish1.nameout()+":");
        dish1.profile();
        System.out.print(dish2.nameout()+":");
```

```
        dish2.profile();
        List<Order> list1=new ArrayList<>();
        list1.add(dish1);
        list1.add(dish2);
        //配送
        sys1.manageOrder(list1,one);
        sys2.manageOrder(list1,two);
        sys2.manageOrder(list1,three);
    }
}
```

学到这里也挺感慨,确实掌握了不少新的知识,多算一份收获，主要把遇到的问题和想法写在前面了，显得头重脚轻了，算了，学长包容一下，(^^ゞ

XY致上