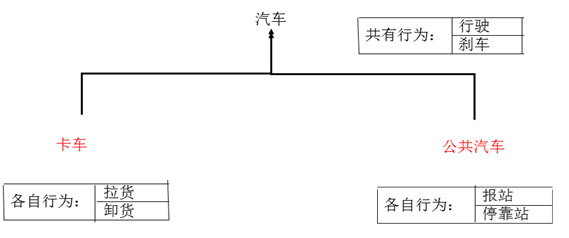
**作业**

任务一：继承



要求：

不定义属性：实现行驶和刹车的方法重写。

package day09.homeWork;

class QiChe{

public void drive() {

System.out.println("父类行驶");

}

public void brake(){

System.out.println("父类刹车");

}

}

class KaChe extends QiChe{

public void laHuo() {

System.out.println("拉货");

}

public void xieHuo() {

System.out.println("卸货");

}

@Override

public void drive() {

System.out.println("卡车类行驶");

}

@Override

public void brake(){

System.out.println("卡车类刹车");

}

}

class Bus extends QiChe{

public void baoZhan() {

System.out.println("报站");

}

public void tingKaoZhan() {

System.out.println("停靠站");

}

@Override

public void drive() {

System.out.println("公共汽车类行驶");

}

@Override

public void brake(){

System.out.println("公共汽车类刹车");

}

}

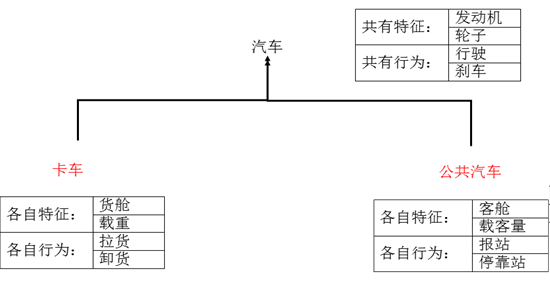
public class ZuoYe1 {

public static void main(String[] args) {

}

}

任务二：继承



要求：

定义属性：

使用构造方法初始化

package day09.homeWork;

class QiChe2{

private String engine;

private String wheel;

public QiChe2() {

super();

}

public QiChe2(String engine, String wheel) {

super();

this.engine = engine;

this.wheel = wheel;

}

public void drive() {

System.out.println("父类行驶");

}

public void brake(){

System.out.println("父类刹车");

}

public String getEngine() {

return engine;

}

public void setEngine(String engine) {

this.engine = engine;

}

public String getWheel() {

return wheel;

}

public void setWheel(String wheel) {

this.wheel = wheel;

}

}

class KaChe2 extends QiChe2{

String huoCang;

int zaiZhong;

public KaChe2() {

super();

}

public KaChe2(String huoCang, int zaiZhong,String engine, String wheel) {

super(engine,wheel);

this.huoCang = huoCang;

this.zaiZhong = zaiZhong;

}

public void laHuo() {

System.out.println("拉货");

}

public void xieHuo() {

System.out.println("卸货");

}

@Override

public void drive() {

System.out.println("卡车类行驶");

}

@Override

public void brake(){

System.out.println("卡车类刹车");

}

public String getHuoCang() {

return huoCang;

}

public void setHuoCang(String huoCang) {

this.huoCang = huoCang;

}

public int getZaiZhong() {

return zaiZhong;

}

public void setZaiZhong(int zaiZhong) {

this.zaiZhong = zaiZhong;

}

}

class Bus2 extends QiChe2{

String keCang;

int zaiKeLiang;

public Bus2() {

super();

}

public Bus2(String keCang, int zaiKeLiang,String engine, String wheel) {

super(engine,wheel);

this.keCang = keCang;

this.zaiKeLiang = zaiKeLiang;

}

public void baoZhan() {

System.out.println("报站");

}

public void tingKaoZhan() {

System.out.println("停靠站");

}

@Override

public void drive() {

System.out.println("公共汽车类行驶");

}

@Override

public void brake(){

System.out.println("公共汽车类刹车");

}

public String getKeCang() {

return keCang;

}

public void setKeCang(String keCang) {

this.keCang = keCang;

}

public int getZaiKeLiang() {

return zaiKeLiang;

}

public void setZaiKeLiang(int zaiKeLiang) {

this.zaiKeLiang = zaiKeLiang;

}

}

public class ZuoYe2 {

public static void main(String[] args) {

KaChe2 k = new KaChe2("货仓", 20, "引擎", "轮子");

Bus2 b = new Bus2("货仓", 20, "引擎", "轮子");

}

}

任务三：

编写程序实现比萨制作。需求说明编写程序，接收用户输入的信息，选择需要制作的比萨。可供选择的比萨有：培根比萨和海鲜比萨。

实现思路及关键代码

1. 分析培根比萨和海鲜比萨
2. 定义比萨类
3. 属性：名称、价格、大小
4. 方法：展示
5. 定义培根比萨和海鲜比萨继承自比萨类

定义比萨工厂类，根据输入信息产生具体的比萨对象

package day09.homeWork;

import java.util.Scanner;

class Pizza{

private String name;

private double price;

private int size;

public Pizza() {

super();

}

public Pizza(String name, double prive, int size) {

super();

this.name = name;

this.price = prive;

this.size = size;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public int getSize() {

return size;

}

public void setSize(int size) {

this.size = size;

}

@Override

public String toString() {

return "我是 " + name + ", 价格：" + price + ", 大小：" + size + "寸";

}

}

class HaiXianPizza extends Pizza{

private int money = 10;

public void setPrice(int size) {

super.setName("海鲜披萨");

super.setSize(size);

super.setPrice(money \* size);

}

public int getMoney() {

return money;

}

public void setMoney(int money) {

this.money = money;

}

}

class PeiGenPizza extends Pizza{

private int money = 15;

public void setPrice(int size) {

super.setName("培根披萨");

super.setSize(size);

super.setPrice(money \* size);

}

public int getMoney() {

return money;

}

public void setMoney(int money) {

this.money = money;

}

}

public class ZuoYe3 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("请输入要定做的Pizza：（1. 海鲜 2.培根）");

switch(sc.nextInt()){

case 1:

HaiXianPizza h = new HaiXianPizza();

System.out.println("您要定做的海鲜披萨单价为"+h.getMoney()+"\n请输入需要定做的尺寸:");

int size = sc.nextInt();

h.setPrice(size);

System.out.println(h);

break;

case 2:

PeiGenPizza p = new PeiGenPizza();

System.out.println("您要定做的培根披萨单价为"+p.getMoney()+"\n请输入需要定做的尺寸:");

int size1 = sc.nextInt();

p.setPrice(size1);

System.out.println(p);

break;

}

}

}

**任务四：**

### 任务描述

编写员工类Employee。该类有如下属性：

id(编号int型)

sex(性别 String型)

name(姓名String型)

duty(职务String型)

salary(薪水double型)

holidays(请假天数 int型)

该类有如下方法

display()，无返回值，该方法能打印员工的姓名、性别以及职务三个属性

getDecMoney(int day) 返回值是int型。该方法能计算员工因请假扣除的工资。计算扣除工资的方法是：如果请假天数<=3，则扣款为30×请假天数；如果请假天数超过3天，则扣款为50×请假天数。参数day表示请假天数。（不用考虑是什么原因请假的问题）

该类的构造方法如下

Employee(int id, String sex,String name,String duty, double salary,int holidays)

编写Main类，在Main类的main方法中，创建一个员工类的对象，并调用其display()和getDecMoney方法。

编写Director类（董事），该类继承Employee类，在Employee类的基础上，新添加assistantName（助手姓名）属性。

为Director编写构造方法，以初始化assistantName属性。

董事收入高，责任也大，因此其请假扣款算法与普通员工不同。董事请假扣款算法如下：

如果请假天数<=3，则扣款为50×请假天数；如果请假天数超过3天，则扣款为80×请假天数。

覆盖父类getDecMone(int day)方法。

package day09.homeWork;

class Employee {

private int id;

private String sex;

private String name;

private String duty;

private double salary;

private int holidays;

public Employee() {}

public Employee(int id, String sex, String name, String duty, double salary, int holidays) {

super();

this.id = id;

this.sex = sex;

this.name = name;

this.duty = duty;

this.salary = salary;

this.holidays = holidays;

}

public void display() {

System.out.println("姓名：" + name + " 性别：" + sex + " 职务：" + duty

+ " 薪水：" + salary + " 请假天数：" + holidays);

}

public int getDecMoney(int day) {

if(day<=3) {

return day\*30;

}else {

return day\*50;

}

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDuty() {

return duty;

}

public void setDuty(String duty) {

this.duty = duty;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public int getHolidays() {

return holidays;

}

public void setHolidays(int holidays) {

this.holidays = holidays;

}

}

class Director extends Employee{

private String assistantName;

public Director(String assistantName) {

super();

this.assistantName = assistantName;

}

@Override

public int getDecMoney(int day) {

if(day<=3) {

return day\*50;

}else {

return day\*80;

}

}

public String getAssistantName() {

return assistantName;

}

public void setAssistantName(String assistantName) {

this.assistantName = assistantName;

}

}

public class ZuoYe4 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Employee e = new Employee(1, "男", "张五", "程序员", 5000, 3);

e.setSalary(e.getSalary()-e.getDecMoney(e.getHolidays()));

e.display();

}

}

任务五：

某公司要开发新游戏，请用面向对象的思想，设计游戏中的蛇怪和蜈蚣精

设定

1. 蛇怪类:

属性包括：怪物名字，生命值，攻击力

方法包括：攻击，移动（曲线移动），补血（当生命值<10时，可以补加20生命值）

1. 蜈蚣精类：

属性包括：怪物名字，生命值，攻击力

方法包括：攻击，移动（飞行移动）

要求

1. 分析蛇怪和蜈蚣精的公共成员，提取出父类—怪物类
2. 利用继承机制，实现蛇怪类和蜈蚣精类
3. 攻击方法，描述攻击状态。内容包括怪物名字，生命值，攻击力
4. 编写测试类，分别测试蛇怪和蜈蚣精的对象及相关方法
5. 定义名为mon的包存怪物类，蛇怪类，蜈蚣精类和测试类

package day09.homeWork;

class Monster{

private String name;

private int health;

private int atk;

public void attack() {

System.out.println(name + " 生命值：" +health+" 攻击力："+atk);

}

public void move() {

System.out.println("移动");

}

public Monster(String name, int health, int atk) {

super();

this.name = name;

this.health = health;

this.atk = atk;

}

public Monster() {

super();

// TODO Auto-generated constructor stub

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getHealth() {

return health;

}

public void setHealth(int health) {

this.health = health;

}

public int getAtk() {

return atk;

}

public void setAtk(int atk) {

this.atk = atk;

}

}

class SheGuai extends Monster{

public SheGuai() {

super("蛇怪", 110, 25);

// TODO Auto-generated constructor stub

}

@Override

public void move() {

System.out.println("曲线移动");

}

public void buXie() {

if(getHealth()>=10)

return;

System.out.println("生命值低于10了 补加20生命值 ");

setHealth(getHealth()+20);

}

}

class WuGongJing extends Monster{

public WuGongJing() {

super("蜈蚣精", 120, 30);

// TODO Auto-generated constructor stub

}

@Override

public void move() {

System.out.println("飞行移动");

}

}

public class ZuoYe5 {

public static void main(String[] args) {

SheGuai s = new SheGuai();

WuGongJing w = new WuGongJing();

s.attack();

w.setHealth(w.getHealth()-s.getAtk());

s.buXie();

s.move();

w.attack();

s.setHealth(s.getHealth()-w.getAtk());

w.move();

}

}