**实验1机动车**

**1.相关知识点**

类是Java中最重要的数据类型。类的目的是抽象出一类事物的共有属性和行为，即抽象出数据以及在数据上所进行的操作。类的类体由两部分组成：变量的声明和方法的定义，其中的构造方法（方法名与类名相同，无类型）用于创建对象，其他方法供该类创建的对象调用。

抽象的目的是产生类，而类的目的是创建具有属性和行为的对象。使用new运算符和类的构造方法为声明的对象分配变量，即创建对象。对象不仅可以操作自己的变量改变状态，而且能调用类中的方法产生一定的行为。通过使用运算符“.”，对象可以实现对自己的变量访问和方法的调用。

Java程序以类为“基本单位”，即一个Java程序就是由若干个类所构成。一个Java程 序可以将它使用的各个类分别存放在不同的源文件中，也可以将它使用的类存放在一个源文件中。因此，要学习Java编程就必须学会怎样去写类，即怎样用Java的语法去描述一类事物共有的属性和行为。

**2.实验目的**

本实验的目的是让学生**使用类来封装对象的属性和功能。**

**3.实验要求**

编写一个Java应用程序，该程序中有两个类：Vehicle (用于刻画机动车）和User (主类)。具体要求如下：

* Vehicle类有一个double类型的变量speed,用于刻画机动车的速度，一个int型变量power，用于刻画机动车的功率。方法定义了 speedUp(int s)方法，体现机动车有减速功能。定义了speedDown()方法，体现机动车有减速功能。方法定义了setPower(int p)方法，用于设置机动车的功率。定义了getPower()方法，用于获取机动车的功率。
* 在主类User的main()方法中用Vehicle类创建对象，并让该对象调用方法设置功率，演示加速和减速功能。

**4. 程序模板**

请按模板要求，将【代码】替换为Java程序代码。

**Vehicle.java**

public class Vehicle {

【代码1】//声明double型变量speed,刻画速度

【代码2】//声明int型变量power,刻画功率

void speedUp(int s) {

【代码3】 //将参数s的值与成员变量speed的和赋值给成员变量speed

}

void speedDown(int d) {

【代码4】 //将成员变量speed与参数d的差赋值给成员变量speed

}

void setPower(int p) {

【代码5】 //将参数p的值赋值给成员变量power

}

int getPower() {

【代码6】 //返回成员变量power的值

}

double getSpeed() {

return speed;

}

}

**User.java**

public class User {

public static void main(String args[]) {

Vehicle car1,car2;

【代码7】 //使用new 运算符和默认的构造方法创建对象car1

【代码8】 //使用new 运算符和默认的构造方法创建对象car2

car1.setPower(128);

car2.setPower(76);

System.out.println("car1的功率是："+car1.getPower());

System.out.println("car2的功率是："+car2.getPower());

【代码9】 //car1调用speedUp方法将自己的speed的值增加80

【代码10】 //car2调用speedUp方法将自己的speed的值增加80

System.out.println("car1目前的速度："+car1.getSpeed());

System.out.println("car2目前的速度："+car2.getSpeed());

car1.speedDown(10);

car2.speedDown(20);

System.out.println("car1目前的速度："+car1.getSpeed());

System.out.println("car2目前的速度："+car2.getSpeed());

}

}

**public class Vehicle {**

**//【代码1】//声明double型变量speed,刻画速度**

**double speed;**

**//【代码2】//声明int型变量power,刻画功率**

**int power;**

**void speedUp(int s) {**

**//【代码3】 //将参数s的值与成员变量speed的和赋值给成员变量speed**

**this.speed=s+this.speed;**

**}**

**void speedDown(int s) {**

**//【代码4】 //将成员变量speed与参数d的差赋值给成员变量speed**

**this.speed=this.speed-s;**

**}**

**void setPower(int p) {**

**//【代码5】 //将参数p的值赋值给成员变量power**

**this.power=p;**

**}**

**int getPower() {**

**//【代码6】 //返回成员变量power的值**

**return power;**

**}**

**double getSpeed() {**

**return speed;**

**}**

**}**

**public class User {**

**public static void main(String args[]) {**

**Vehicle car1,car2;**

**//【代码7】 //使用new 运算符和默认的构造方法创建对象car1**

**car1=new Vehicle();**

**//【代码8】 //使用new 运算符和默认的构造方法创建对象car2**

**car2=new Vehicle();**

**car1.setPower(128);**

**car2.setPower(76);**

**System.out.println("car1的功率是："+car1.getPower());**

**System.out.println("car2的功率是："+car2.getPower());**

**//【代码9】 //car1调用speedUp方法将自己的speed的值增加80**

**car1.speedUp(80);**

**//【代码10】 //car2调用speedUp方法将自己的speed的值增加80**

**car2.speedUp(80);**

**System.out.println("car1目前的速度："+car1.getSpeed());**

**System.out.println("car2目前的速度："+car2.getSpeed());**

**car1.speedDown(10);**

**car2.speedDown(20);**

**System.out.println("car1目前的速度："+car1.getSpeed());**

**System.out.println("car2目前的速度："+car2.getSpeed());**

**}**

**}**

**类的创建：car2=new Vehicle();**

**关于User.java:48: 错误: 编码 GBK 的不可映射字符 (0x9A) 错误：**

**在命令行输入：**

**javac -encoding UTF-8 文件名.java**

**5. 实验指导**

创建一个对象时，成员变量被分配内存空间，这些内存空间称做该对象的实体或变量，而对象中存放着引用，以确保这些变量被该对象操作使用。

**空对象**(**没有实际遇到过**)不能使用，**即不能让一个空对象去调用方法产生行为**。假如程序中使用了空对象，在运行时会出现异常：NullPointerException。对象是动态地分配实体的，

Java的编译器对空对象不做检查。因此，在编写程序时要避免使用空对象。

**6. 实验后的练习**

(1)改进speedUP()方法，使得Vehicle类的对象加速时不能将speed值超过200。

(2) 改进speedDown()方法，使得Vehicle类的对象在减速时不能将speeds小于0。

(3) 增加一个刹车方法voidbrake()，Vehicle类的对象调用它能将speed的值变成0。

**public class User {**

**public static void main(String args[]) {**

**Vehicle car1,car2;**

**//【代码7】 //使用new 运算符和默认的构造方法创建对象car1**

**car1=new Vehicle();**

**//【代码8】 //使用new 运算符和默认的构造方法创建对象car2**

**car2=new Vehicle();**

**car1.setPower(128);**

**car2.setPower(76);**

**System.out.println("car1的功率是："+car1.getPower());**

**System.out.println("car2的功率是："+car2.getPower());**

**//【代码9】 //car1调用speedUp方法将自己的speed的值增加80**

**car1.speedUp(280);**

**//【代码10】 //car2调用speedUp方法将自己的speed的值增加80**

**car2.speedUp(80);**

**System.out.println("car1目前的速度："+car1.getSpeed());**

**System.out.println("car2目前的速度："+car2.getSpeed());**

**car1.speedDown(-10);**

**car2.speedDown(20);**

**System.out.println("car1目前的速度："+car1.getSpeed());**

**System.out.println("car2目前的速度："+car2.getSpeed());**

**}**

**}**

**public class Vehicle {**

**//【代码1】//声明double型变量speed,刻画速度**

**double speed;**

**//【代码2】//声明int型变量power,刻画功率**

**int power;**

**void speedUp(int s) {**

**//【代码3】 //将参数s的值与成员变量speed的和赋值给成员变量speed**

**if(s<=200)**

**this.speed=s+this.speed;**

**}**

**void brake(){**

**this.speed=0;**

**}**

**void speedDown(int s) {**

**//【代码4】 //将成员变量speed与参数d的差赋值给成员变量speed**

**if(s>0)**

**this.speed=this.speed-s;**

**}**

**void setPower(int p) {**

**//【代码5】 //将参数p的值赋值给成员变量power**

**this.power=p;**

**}**

**int getPower() {**

**//【代码6】 //返回成员变量power的值**

**return power;**

**}**

**double getSpeed() {**

**return speed;**

**}**

**}**

当仅仅执行 java Vehicle时，会报错

**错误: 在类 Vehicle 中找不到 main 方法, 请将 main 方法定义为:**

**public static void main(String[] args)**

**否则 JavaFX 应用程序类必须扩展javafx.application.Application**

**问题解释的很清楚了**

**7. 填写实验报告**

**实验2家中的电视**

**1. 相关知识点**

**类的成员变量可以是某个类的对象，**如果用这样的类创建对象，那么该对象中就会有其他对象，也就是说该类的对象将其他对象作为自己的组成部分，这就是人们常说的 Has-A。一个对象a通过组合对象b来复用对象b的方法，即对象a委托对象b调用其方法。当前对象随时可以更换所组合对象，使得当前对象与所组合的对象是弱耦合关系。

**2. 实验目的**

本实验的目的是让学生掌握**对象的组合以及参数传递**。

**3. 实验要求**

编写一个Java应用程序，模拟家庭买一台电视，即家庭将电视作为自己的一个成员，即通过调用一个方法将某个电视的引用传递给自己的电视成员。具体要求如下。

•有三个源文件：TV.java、Familiy.java 和 MainClass.java，其中 TV.java 中的 TV类负责创建“电视”对象，Family.java中的Family类负责创建“家庭”对象，MainCiass.java 是主类。

•在主类的main()方法中首先使用TV类创建一个对象haierTV，然后使用Familiy类再创建一个对象zhangSanFamily，并将先前TV类的实例haierTV的引用传递给 zhangSanFamily对象的成员变量homeTV。

Family类组合TV类的实例的UML图如图4.4所示。

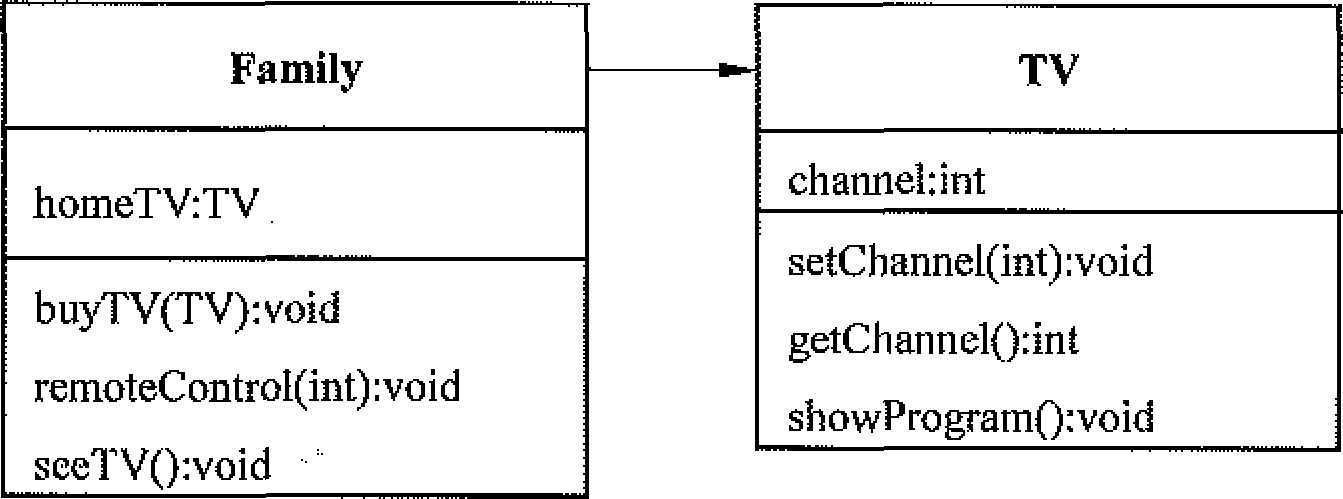


图4.4 Family组合TV的实例的UML图

**4. 程序模板**

请按模板要求，将【代码】替换为Java程序代码。

TV.java

public class TV {

int channel; //电视频道

void setChannel(int m) {

if(m>=1){

channel=m;

}

}

int getChannel(){

return channel;

}

void showProgram(){

switch(channel) {

case 1 : System.out.println("综合频道");

break;

case 2 : System.out.println("经济频道");

break;

case 3 : System.out.println("文艺频道");

break;

case 4 : System.out.println("国际频道");

break;

case 5 : System.out.println("体育频道");

break;

default : System.out.println("不能收看"+channel+"频道");

}

}

}

**Family.java**

public class Family {

TV homeTV;

void buyTV(TV tv) {

【代码1】 //将参数tv赋值给homeTV

**homeTV=tv;**

}

void remoteControl(int m) {

homeTV.setChannel(m);

}

void seeTV() {

homeTV.showProgram(); //homeTV调用showProgram()方法

}

}

**MainClass.java**

public class MainClass {

public static void main(String args[]) {

TV haierTV = new TV();

【代码2】 //haierTV调用setChannel(int m),并向参数m传递5

**haierTV.setChannel(5);**

System.out.println("haierTV的频道是"+haierTV.getChannel());

Family zhangSanFamily = new Family();

【代码3】//zhangSanFamily调用void buyTV(TV tv)方法,并将haierTV传递给参数TV

**zhangSanFamily.buyTV(haierTV);**

System.out.println("zhangSanFamily开始看电视节目");

zhangSanFamily.seeTV();

int m=2;

System.out.println("hangSanFamily将电视更换到"+m+"频道");

zhangSanFamily.remoteControl(m);

System.out.println("haierTV的频道是"+haierTV.getChannel());

System.out.println("hangSanFamily再看电视节目");

zhangSanFamily.seeTV();

}

}

**5.实验指导**

当参数是引用类型时，“传值”传递的是变量中存放的“引用”，而不是变量所引用的实体。需要注意的是，对于两个同类型的引用型变量，如果具有同样的引用，就会有同样的实体，因此，如果改变参数变量所引用的实体，就会导致原变量的实体发生同样的变化。

通过组合对象来复用方法也称“黑盒”复用，因为当前对象只能委托它所包含的对象调用其方法，这样一来，当前对象对它所包含的对象的方法的细节是一无所知的。

**6.实验后的练习**

(1) 省略【代码2】程序能否通过编译？若能通过编译，程序输出的结果是怎样的？

**可以的**

**运行结果：**

**haierTV的频道是0**

**zhangSanFamily开始看电视节目**

**不能收看0频道**

**hangSanFamily将电视更换到2频道**

**haierTV的频道是2**

**hangSanFamily再看电视节目**

**经济频道**

**第一行：hairTV的频道为0(默认)，一开始的代码是修改为5**

(2) 在主类的main()方法的最后增添下列代码，并解释运行效果。

Family lisiFamily = new Family();

lisiFamily.buyTV(haierTV);

lisiFamily.seeTV();

**hangSanFamily再看电视节目**

**经济频道**

**经济频道**

**在类中对成员类的对象进行修改的作用是，永久的**

**7.填写实验报告**

**实验3共饮同井水**

**1.相关知识点**

类有两种基本的成员：变量和方法，变量用来刻画对象的属性，方法用来体现对象的功能，即方法使用某种算法操作变量来实现一个具体的行为(功能)。

成员变量用来刻画类创建的对象的属性，其中一部分成员变量称作实例变量，另一部分称作静态变量或类变量。类变量是与类相关联的数据变量，而实例变量是仅仅和对象相关联的数据变量。不同的对象的实例变量将被分配不同的内存空间，如果类中有类变量，那么所有对象的这个类变量都分配给相同的一处内存，改变其中一个对象的这个类变量会 影响其他对象的这个类变量。也就是说，对象共享类变量。

方法是类体的重要成员之一。其中的构造方法是具有特殊地位的方法，供类创建对象时使用，用来给出类所创建的对象的初始状态，另一部分方法可分为实例方法和类方法，类所创建的对象可以调用这些方法形成一定的算法，体现对象具有某种行为。一个类的类方法也可以直接通过该类的类名调用。

当对象调用方法时，方法中出现的成员变量就是指分配给该对象的变量。类中的方法可以操作成员变 量，当对象调用方法时，方法中出现的成员变量就是指 分配给该对象的变量，其中的类变量和所有的其他对象共享。

实例方法可操作实例成员变量和静态成员变量，静态方法只能操作静态成员变量，如图4.6所示。

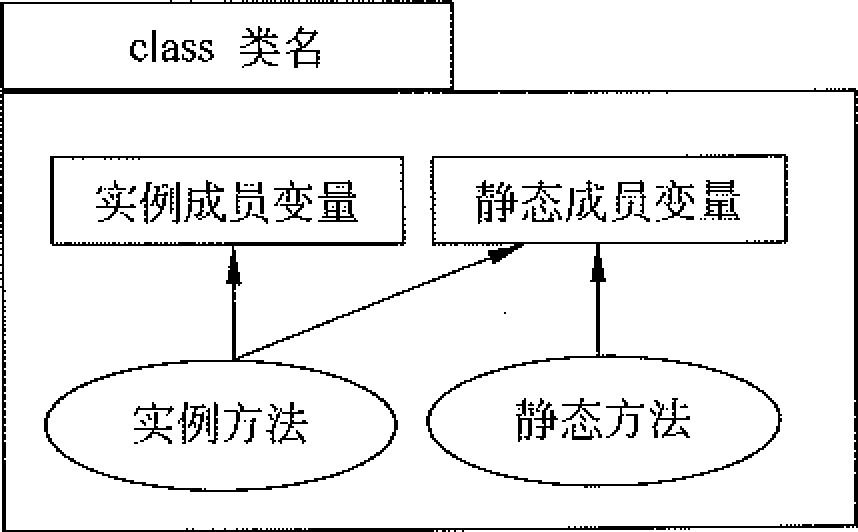


图4.6实例成员与类成员

**2. 实验目的**

本实验的目的是让学生掌握**类变量与实例变量，以及类方法与实例方法的区别**。

**3. 实验要求**

编写程序模拟两个村庄共用同一口井水。编写一个Village类，该类有一个静态的int 型成员变量waterAmount，用于**模拟井水的水量**。在主类Land的main()方法中**创建两个村庄**，**一个村庄改变了 waterAmount的值，另一个村庄查看waterAmount的值。**

**4. 程序模板**

请按模板要求，将【代码】替换为Java程序代码。

**Village.java**

public class Village {

static int waterAmount; //模拟水井的水量

int peopleNumber; //村庄的人数

String name; //村庄的名字

Village(String s) {

name = s;

}

static void setWaterAmount(int m) {

if(m>0)

waterAmount = m;

}

void drinkWater(int n){

if( waterAmount-n>=0) {

waterAmount = waterAmount-n;

System.out.println(name+"喝了"+n+"升水");

}

else

waterAmount = 0;

}

static int lookWaterAmount() {

return waterAmount;

}

void setPeopleNumber(int n) {

peopleNumber = n;

}

int getPeopleNumber() {

return peopleNumber;

}

}

**Land.java**

public class Land {

public static void main(String args[]) {

【代码1】 //用类名调用setWaterAmount(int m),并向参数传值200

int leftWater =【代码2】 //用Village类的类名访问waterAmount

System.out.println("水井中有 "+leftWater+" 升水");

Village zhaoZhuang,maJiaHeZhi;

zhaoZhuang = new Village("赵庄");

maJiaHeZhi = new Village("马家河子");

zhaoZhuang.setPeopleNumber(80);

maJiaHeZhi.setPeopleNumber(120);

【代码3】//zhaoZhuang调用drinkWater(int n),并向参数传值50

leftWater = 【代码4】//maJiaHeZhi调用lookWaterAmount()方法

String name=maJiaHeZhi.name;

System.out.println(name+"发现水井中有 "+leftWater+" 升水");

maJiaHeZhi.drinkWater(100);

leftWater = 【代码5】//zhaoZhuang调用lookWaterAmount()方法

name=zhaoZhuang.name;

System.out.println(name+"发现水井中有 "+leftWater+" 升水");

int peopleNumber = zhaoZhuang.getPeopleNumber();

System.out.println("赵庄的人口:"+peopleNumber);

peopleNumber = maJiaHeZhi.getPeopleNumber();

System.out.println("马家河子的人口:"+peopleNumber);

}

}

**public class Village {**

**static int waterAmount; //模拟水井的水量**

**int peopleNumber; //村庄的人数**

**String name; //村庄的名字**

**Village(String s) {**

**name = s;**

**}**

**static void setWaterAmount(int m) {///设置水井的水量**

**if(m>0)**

**waterAmount = m;**

**}**

**void drinkWater(int n){///喝水了**

**if( waterAmount-n>=0) {**

**waterAmount = waterAmount-n;**

**System.out.println(name+"喝了"+n+"升水");**

**}**

**else**

**waterAmount = 0;**

**}**

**static int lookWaterAmount() {///查看水井的水量**

**return waterAmount;**

**}**

**void setPeopleNumber(int n) {///设置村庄的人数**

**peopleNumber = n;**

**}**

**int getPeopleNumber() {///返回村庄的人数**

**return peopleNumber;**

**}**

**}**

**public class Land {**

**public static void main(String args[]) {**

**//【代码1】 //用类名调用setWaterAmount(int m),并向参数传值200**

**Village.setWaterAmount(200);**

**//int leftWater =【代码2】 //用Village类的类名访问waterAmount**

**int leftWater = Village.lookWaterAmount();**

**System.out.println("水井中有 "+leftWater+" 升水");**

**Village zhaoZhuang,maJiaHeZhi;**

**zhaoZhuang = new Village("赵庄");**

**maJiaHeZhi = new Village("马家河子");**

**zhaoZhuang.setPeopleNumber(80);**

**maJiaHeZhi.setPeopleNumber(120);**

**//【代码3】//zhaoZhuang调用drinkWater(int n),并向参数传值50**

**zhaoZhuang.drinkWater(50);///喝了50单位的水**

**//leftWater = 【代码4】//maJiaHeZhi调用lookWaterAmount()方法**

**leftWater = maJiaHeZhi.lookWaterAmount();///马家还剩多少单位的水**

**String name=maJiaHeZhi.name;**

**System.out.println(name+"发现水井中有 "+leftWater+" 升水");**

**maJiaHeZhi.drinkWater(100);**

**//leftWater = 【代码5】//zhaoZhuang调用lookWaterAmount()方法**

**leftWater = zhaoZhuang.lookWaterAmount();**

**name=zhaoZhuang.name;**

**System.out.println(name+"发现水井中有 "+leftWater+" 升水");**

**int peopleNumber = zhaoZhuang.getPeopleNumber();**

**System.out.println("赵庄的人口:"+peopleNumber);**

**peopleNumber = maJiaHeZhi.getPeopleNumber();**

**System.out.println("马家河子的人口:"+peopleNumber);**

**}**

**}**

**5. 实验指导**

当Java程序执行时，类的字节码文件被加载到内存，如果该类没有创建对象，类的实例变量不会被分配内存。但是，类中的类变量，在该类被加载到内存时，就分配了相应的内存空间。如果该类创建对象，那么不同对象的实例变量互不相同，即分配不同的内存空间，而类变量不再重新分配内存，所有的对象共享类变量。

当类的字节码文件被加载到内存时，类的实例方法不会被分配入口地址，只有当该类创建对象后，类中的实例方法才分配入口地址。当使用new运算符和构造方法创建对象时，首先分配成员变量给该对象，同时实例方法分配入口地址，然后再执行构造方法中的语句，完成必要的初始化。因而实例方法必须由对象调用执 行。**需要注意的是**，当创建第一个对象时，类中的实例方法就分配了入口地址， 当再创建对象时，不再分配入口地址，也就是说，方法的入口地址被所有的对象共享。对于类中的类方法，在该类被加载到内存时，就分配了相应的入口地址，即使该类没有创建对象，也可以直接通过类名调用类方法（当然，类方法也可以通过对象调用）。

**6. 实验后的练习**

(1) 【代码3】是否可以是Village.drinkWater(50);

**不可以**

**Land.java:15: 错误: 无法从静态上下文中引用非静态 方法 drinkWater(int)**

**Village.drinkWater(50);**

**^**

**1 个错误**

(2) 【代码 4】是否可以是 Village.lookWaterAmount();

**可以的**

**水井中有 200 升水**

**赵庄喝了50升水**

**马家河子发现水井中有 200 升水**

**马家河子喝了100升水**

**赵庄发现水井中有 50 升水**

**赵庄的人口:80**

**马家河子的人口:120**

(3) Land类main()方法中倒数第2行代码是否可以更改为：

peopleNumber = Village .getPeopleNumber ();

**Land.java:29: 错误: 无法从静态上下文中引用非静态 方法 getPeopleNumber()**

**peopleNumber = Village .getPeopleNumber ();**

**^**

**1 个错误**

**7.填写实验报告**