# 07 周

1. 定义并测试一个代表员工的Employee类。员工属性包括“编号”、“姓名”、“基本薪水”、“薪水增长额”；还包括 “计算增长后的工资总额”。

|  |  |
| --- | --- |
| **class** Employee {  **private** **int** empno ; // 雇员编号  **private** String ename ; // 雇员姓名  **private** **double** sal ; // 基本工资  **private** **double** rate ; // 工资增长额  **public** Employee() {  }  **public** Employee(**int** empno, String ename, **double** sal, **double** rate) {  **super**();  **this**.empno = empno;  **this**.ename = ename;  **this**.sal = sal;  **this**.rate = rate;  }  **public** String toString() {  **return** "雇员编号：" + **this**.empno + "，雇员姓名：" + **this**.ename + "，基本工资：" + **this**.sal ;  }  **public** **void** growthin() { // 增长薪水  **this**.sal = **this**.sal \* **this**.rate ;  }  // setter、getter略  }  **public** **class** TestDemo {  **public** **static** **void** main(String args[]) {  Employee emp = **new** Employee(7369, "SMITH", 1000, 1.5);  emp.growthin() ; // 工资增长  System.*out*.println(emp);  }  } | |
| **程序运行结果：** | 雇员编号：7369，雇员姓名：SMITH，基本工资：1500.0 |

1. 编写并测试一个代表地址的Address类，地址信息由：国家，省份，城市，街道，邮编组成，并可以返回完整的地址信息。

|  |  |
| --- | --- |
| **class** Address {  **private** String national;  **private** String provincial;  **private** String city;  **private** String street;  **private** String zipcode;  **public** Address() {  }  **public** Address(String national, String provincial, String city,  String street, String zipcode) {  **super**();  **this**.national = national;  **this**.provincial = provincial;  **this**.city = city;  **this**.street = street;  **this**.zipcode = zipcode;  }  **public** String toString() {  **return** "国家：" + **this**.national + "，省份：" + **this**.provincial + "，城市："  + **this**.city + "，街道：" + **this**.street + "，邮政编码：" + **this**.zipcode;  }  // setter、getter略  }  **public** **class** TestDemo {  **public** **static** **void** main(String args[]) {  Address ad = **new** Address("中国", "北京", "北京市", "MLDN", "100088");  System.*out*.println(ad);  }  } | |
| **程序运行结果：** | 国家：中国，省份：北京，城市：北京市，街道：MLDN，邮政编码：100088 |

# 08 周

一、编写一个银行帐户类,类的构成包括：

· 数据成员：

|- 用户的帐户名称、用户的帐户余额；

· 方法包括：

|- 开户（设置帐户名称，及余额），利用构造方法完成

· 查询余额

|  |  |
| --- | --- |
| **class** Account {  **private** String name ;  **private** **double** balance ;  **public** Account() {  }  **public** Account(String name, **double** balance) {  **super**();  **this**.name = name;  **this**.balance = balance;  }  **public** String toString() {  **return** "账户名称：" + **this**.name + "，余额：" + **this**.balance;  }  **public** **double** getBalance() {  **return** balance;  }  // setter、getter略  }  **public** **class** TestDemo {  **public** **static** **void** main(String args[]) {  Account acc = **new** Account("张三", 5000.0);  System.*out*.println(acc);  System.*out*.println("账户余额：" + acc.getBalance());  }  } | |
| **程序运行结果：** | 账户名称：张三，余额：5000.0  账户余额：5000.0 |

二、建立一个人类（Person）和学生类（Student）功能要求：

A、 Person中包含4个保护型的数据成员name、address、sex、age分别为字符串，字符串，字符及整型。表示：姓名、地址、性别和年龄。一个四参构造方法，一个无参构造方法，及一个输出方法用于显示四种属性。

B、 Student继承Person，并增加输出成员math、english存放数学和英语成绩。一个六参构造方法，一个两参构造方法，一个无参构造方法，重写输出方法用于显示全部六种种属性.

|  |  |
| --- | --- |
| **class** Person {  **private** String name ;  **private** String address ;  **private** **char** sex ;  **private** **int** age ;  **public** Person() {  }  **public** Person(String name, String address, **char** sex, **int** age) {  **super**();  **this**.name = name;  **this**.address = address;  **this**.sex = sex;  **this**.age = age;  }  **public** String toString() {  **return** "姓名：" + **this**.name + "，地址：" + **this**.address + "，性别：" + **this**.sex  + "，年龄：" + **this**.age;  }  // setter、getter略  }  **class** Student **extends** Person {  **private** **double** math ;  **private** **double** english ;  **public** Student() {  }  **public** Student(String name, String address, **char** sex, **int** age, **double** math,  **double** english) {  **super**(name, address, sex, age);  **this**.math = math;  **this**.english = english;  }  **public** String toString() {  **return** **super**.toString() + "，数学成绩：" + **this**.math + "，英语成绩：" + **this**.english;  }  // setter、getter略  }  **public** **class** TestDemo {  **public** **static** **void** main(String args[]) {  Student stu = **new** Student("张三", "北京西城区甲11号德外大街德胜科技园美江大厦 A座 - 6层", '男', 25, 90.0, 99.0);  System.*out*.println(stu);  }  } | |
| **程序运行结果：** | 姓名：张三，地址：北京西城区甲11号德外大街德胜科技园美江大厦 A座 - 6层，性别：男，年龄：25，数学成绩：90.0，英语成绩：99.0 |

**实训四： 继承**

1、定义员工类，具有姓名，年龄，性别属性，并具有构造方法，显示数据方法，定义管理层类，继承员工类，并有自己的属性：职务，年薪。定义职员类，继承员工类，并有自己的属性：所属部门，月薪。

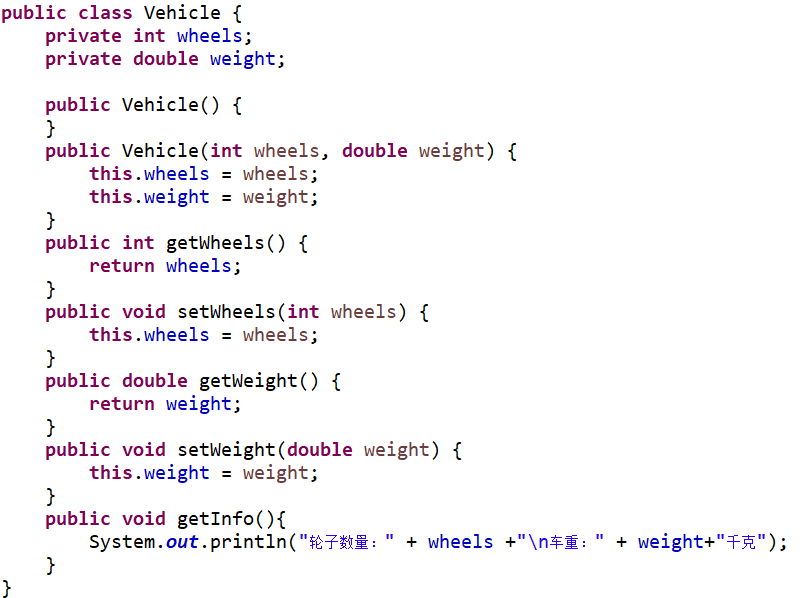
|  |  |
| --- | --- |
| **class** Employee {  **private** String name ;  **private** **int** age ;  **private** **char** sex ;  **public** Employee() {  }  **public** Employee(String name, **int** age, **char** sex) {  **super**();  **this**.name = name;  **this**.age = age;  **this**.sex = sex;  }  **public** String toString() {  **return** "雇员姓名：" + **this**.name + "，年龄：" + **this**.age + "，性别：" + **this**.sex;  }  // setter、getter略  }  **class** Manager **extends** Employee {  **private** String job ;  **private** **double** income ;  **public** Manager() {  }  **public** Manager(String name, **int** age, **char** sex, String job, **double** income) {  **super**(name, age, sex);  **this**.job = job;  **this**.income = income;  }  **public** String toString() {  **return** **super**.toString() + "，职位：" + **this**.job + "，年薪：" + **this**.income ;  }  // setter、getter略  }  **class** Staff **extends** Employee {  **private** String dept ;  **private** **double** salary ;  **public** Staff() {  }  **public** Staff(String name, **int** age, **char** sex, String dept, **double** salary) {  **super**(name, age, sex);  **this**.dept = dept;  **this**.salary = salary;  }  **public** String toString() {  **return** **super**.toString() + "，部门：" + **this**.dept + "，月薪：" + **this**.salary ;  }  // setter、getter略  }  **public** **class** TestDemo {  **public** **static** **void** main(String args[]) {  Employee ea = **new** Manager("张三", 30, '男', "总监", 200000.0);  Employee eb = **new** Staff("李四", 25, '女', "业务部", 1500.0);  System.*out*.println(ea);  System.*out*.println(eb);    }  } | |
| **程序运行结果：** | 雇员姓名：张三，年龄：30，性别：男，职位：总监，年薪：200000.0  雇员姓名：李四，年龄：25，性别：女，部门：业务部，月薪：1500.0 |

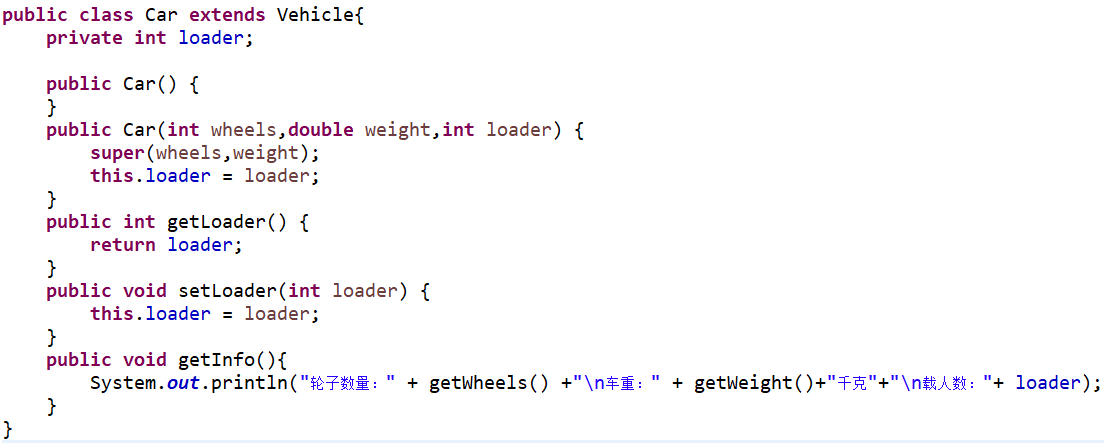
2、编写一个程序，设计一个汽车类Vehicle，包含的属性有车轮个数wheels和车重weight。小车类Car是Vehicle的子类，其中包含的属性有载人数loader。卡车类Truck是Car类的子类，其中包含的属性有载重量payload。

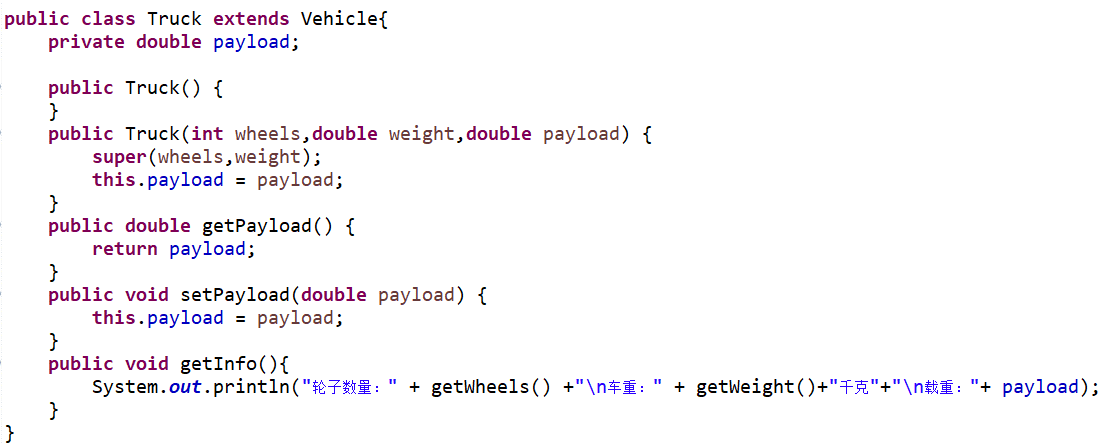
（1）每个类都有构造方法进行属性初始化；

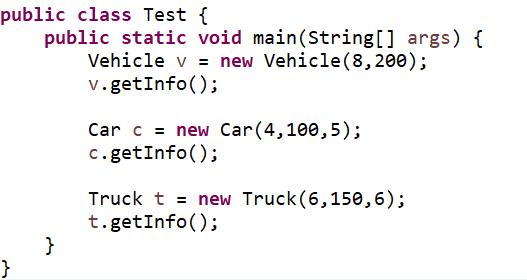
（2）每个类都输出相关数据信息的成员方法；

（3）使用Test类中的main方法实例化各类，并输出相关信息。









**实训五： 抽象类**

1、编写一个程序，设计一个汽车抽象类Vehicle，包含的属性有车轮个数wheels和车重weight。小车类Car是Vehicle的子类，其中包含的属性有载人数loader。卡车类Truck是Car类的子类，其中包含的属性有载重量payload。

（1）每个类都有构造方法进行属性初始化；

（2）抽象类Vehicle类具有输出相关数据信息的抽象方法；

（3）Car，Truck子类覆写抽象类的抽象方法，并给子类分别添加一个子类特有的方法（内容自定）；

（4）使用Test类中的main方法实例化Car，Truck，并输出相关信息（分别练习向上转型和向下转型）。

**abstract** **class** Vehicle{

**private** **int** wheels;

**private** **double** weight;

**public** Vehicle() {

}

**public** Vehicle(**int** wheels, **double** weight) {

**this**.wheels = wheels;

**this**.weight = weight;

}

**public** **int** getWheels() {

**return** wheels;

}

**public** **void** setWheels(**int** wheels) {

**this**.wheels = wheels;

}

**public** **double** getWeight() {

**return** weight;

}

**public** **void** setWeight(**double** weight) {

**this**.weight = weight;

}

**public** **abstract** **void** prininfo();

}

**class** Car **extends** Vehicle{

**private** **int** loader;

**public** Car() {

**super**();

}

**public** Car(**int** loader,**int** wheels ,**double** weight) {

**super**(wheels,weight);

**this**.loader = loader;

}

**public** **int** getLoader() {

**return** loader;

}

**public** **void** setLoader(**int** loader) {

**this**.loader = loader;

}

**public** **void** prininfo(){

System.***out***.println("车重："+ getWeight()+"千克"+"\n车轮数：" + getWheels());

}

**public** **void** prinLoader(){

System.***out***.println("小车载人数为：" + loader);

}

}

**class** Truck **extends** Car{

**private** **int** payload;

**public** Truck() {

**super**();

}

**public** Truck(**int** loader,**int** wheels ,**double** weight,**int** payload) {

**super**(loader,wheels,weight);

**this**.payload = payload;

}

**public** **int** getPayload() {

**return** payload;

}

**public** **void** setPayload(**int** payload) {

**this**.payload = payload;

}

**public** **void** prininfo(){

System.***out***.println("车重："+ getWeight()+"千克"+"\n车轮数：" + getWheels()+"\n载人数：" + getLoader());

}

**public** **class** TestDemo01 {

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

Vehicle v1 = **new** Car(5,4,100); //向上转型

v1.prininfo();

//v1.prinLoader(); 错误，不能调用子类的独有方法

Car v2 = (Car) v1; //向下转型

v2.prinLoader();

System.***out***.println("---------------");

Vehicle v3 = **new** Truck(10,6,1000,500);

v3.prininfo();

Truck v4 = (Truck)v3;

v4.prinLoader();

v4.prinPayload();

}

}

2、定义Shape抽象类，包含私有属性color，创建构造器为color赋值；包含计算周长的方法celPerimeter();定义子类Triangle，包含三边；定义子类Circle，包含半径radius；子类分别实现父类的计算周长功能。

**abstract** **class** Shape{

**private** String color;

**public** Shape(String color) {

**this**.color = color;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** **abstract** **double** celPerimeter();

}

**class** Triangle **extends** Shape{

**private** **double** x,y,z;

**public** Triangle(String color, **double** x, **double** y, **double** z) {

**super**(color);

**this**.x = x;

**this**.y = y;

**this**.z = z;

}

**public** **double** celPerimeter(){

**return** x + y + z;

}

}

**class** Circle **extends** Shape{

**private** **double** radius;

**public** Circle(String color, **double** radius) {

**super**(color);

**this**.radius = radius;

}

**public** **double** celPerimeter(){

**return** 2 \* 3.14 \* radius;

}

}

**public** **class** TestDemo02 {

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

Shape s1 = **new** Triangle("白色",3,4,5);

System.***out***.println("三角形颜色：" + s1.getColor());

System.***out***.println("三角形周长：" + s1.celPerimeter());

System.***out***.println("-------------------");

Shape s2 = **new** Circle("蓝色",2);

System.***out***.println("圆形的颜色：" + s2.getColor());

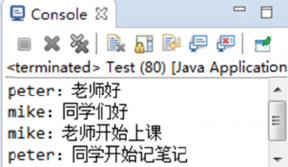
System.***out***.println("圆周长：" + s2.celPerimeter());

}

}

**实训六： 接口**

1. 模拟上课场景：创建教师类和学生类，两个类都实现了问候接口和工作接口，运行效果如下：



**public** **interface** interfaceWork {

**public** **abstract** **void** work();

}

**public** **interface** interfaceSay {

**public** **abstract** **void** say();

}

**public** **class** Teacher **implements** interfaceWork,interfaceSay{

**private** String name;

**public** Teacher() {

}

**public** Teacher(String name) {

**this**.name = name;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** say(){

System.***out***.println(name + ":同学们好");

}

**public** **void** work(){

System.***out***.println(name +":老师开始上课");

}

}

**public** **class** InterfaceTest1 {

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

Teacher teacher = **new** Teacher("mike");

Student student = **new** Student("peter");

student.say();

teacher.say();

teacher.work();

student.work();

}

}

**public** **class** Student **implements** interfaceWork,interfaceSay{

**private** String name;

**public** Student() {

}

**public** Student(String name) {

**this**.name = name;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** say(){

System.***out***.println(name + ":老师好");

}

**public** **void** work(){

System.***out***.println(name +":同学开始记笔记");

}

}

## 请使用接口编码实现如下需求：乐器（Instrument）分为：钢琴(Piano)、小提琴(Violin).各种乐器的弹奏（ play ）方法各不相同。编写一个测试类InstrumentTest，要求：编写方法testPlay，对各种乐器进行弹奏测试。要依据乐器的不同，进行相应的弹奏。在main方法中创建不同的乐器对象，通过testPlay的弹奏测试方法进行测试。

**public** **class** InstrumentTest {

**public** **void** testPlay(Instrument instrument){

System.***out***.println(instrument.play());

}

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

InstrumentTest t = **new** InstrumentTest();

Instrument m1 = **new** Piano();

Instrument m2 = **new** Violin();

t.testPlay(m1);

t.testPlay(m2);

// 或者：

// t.testPlay(new Piano());

// t.testPlay(new Violin());

}

}

**public** **class** Violin **implements** Instrument{

**public** String play(){

**return** "拉小提琴";

}

}

**public** **class** Piano **implements** Instrument{

**public** String play(){

**return** "弹钢琴";

}

}

**public** **interface** Instrument {

**public** **abstract** String play();

}