**实 验 报 告**

**课程名称：** 面向对象技术

**实验项目：** 类和对象

**实验仪器：** PC机

**学 院： 计算机学院**

**专 业：**

**姓 名：**

**班 级：**

**学 号：**

**日 期：**

**指导教师：**

**同组成员： 无**

**成 绩：**

# **实验目的**

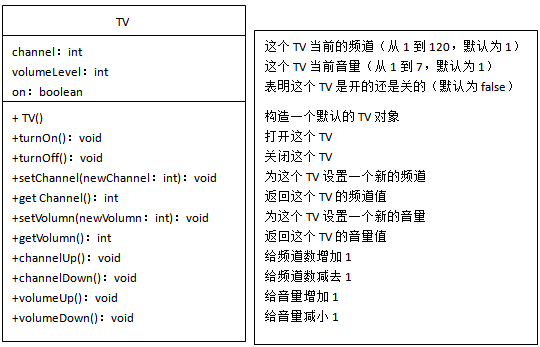
1. 掌握JAVA 开发环境的搭建；
2. 掌握JAVA 程序开发流程；
3. 掌握JAVA 基本语法；
4. 掌握类的定义和对象的创建以及使用方法；
5. 掌握UML类图的绘制。

# **实验内容**

1. 编写电视类TV，UML类图和说明如下： 注意：

（1）电视只有在打开的状态下才可以调音量和设置新频道；

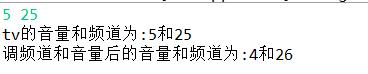
（2）设置音量和频道时，应该在规定的范围，如音量值为1-7。



解题要求：编写测试类，按照如下的顺序测试TV的功能：

* 1. 创建一个TV对象；
  2. 打开TV对象，从键盘上读入要设置的音量和频道；
  3. 输出TV对象的音量和频道；
  4. 给TV对象的频道加1，音量减1；
  5. 输出TV对象的当前音量和频道；
  6. 关闭TV对象。

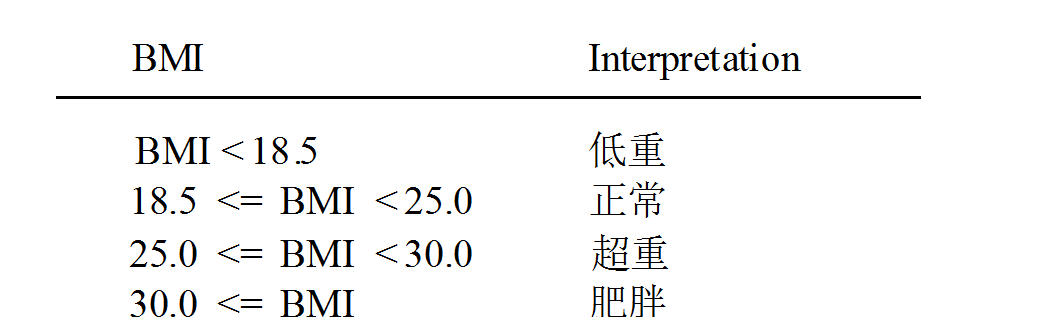
按照如下样例输入输出相关信息，绿色内容为输入的信息，黑色内容为输出信息。



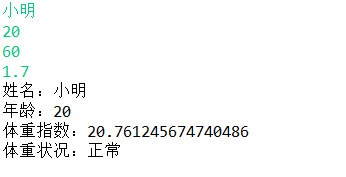
1. 创建用于计算身体质量指数的BMI类，成员变量和成员方法如下：
2. 一个字符串类型的私有数据域（数据域、成员变量和属性含义相同）name，描述人的名字；
3. 一个int类型的私有数据域age，描述人的年龄；
4. 一个double类型的私有数据域weight，描述人的体重；
5. 一个double类型的私有数据域height，描述人的身高；
6. 一个参数为空的构造方法；
7. 一个创建带特定体重和身高的BMI对象的构造方法；
8. 一个创建带特定名字、年龄、体重和身高的BMI对象的构造方法；
9. 所有数据域的get和set方法；
10. 计算身体质量指数的方法getBMI()，返回值为double类型。

BMI计算公式：    BMI = 体重公斤数/身高米数2

1. 按照表格提供的体重状况判断方法，返回身体状态的方法getStatus()，返回值类型为String。

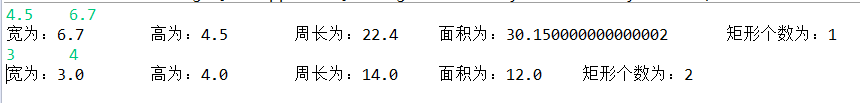


编写测试类，创建BMI对象，从键盘输入某人的姓名、年龄、体重和身高，判断其体重状况，按照如下样例输入输出相关信息。



1. 设计一个名为Rectangle的类表示矩形,这个类包括：
2. 一个名为int类型的静态变量numberOfRectangle，记录创建的矩形的个数；
3. 两个名为width和height的double型数据域，它们分别表示矩形的宽和高。width和height的默认值都为1。
4. 创建默认矩形的无参构造方法。
5. 一个创建width和height为指定值的矩形的构造方法。
6. 两个数据域的get和set方法。
7. 一个名为getArea()的方法返回这个矩形的面积。
8. 一个名为getPerimeter()的方法返回周长。
9. 一个名为getNumberOfRectangle()的静态方法，获取已经创建的矩形对象的个数。

编写一个测试程序，分别使用两个构造方法，创建两个Rectangle对象，并按照下面的样例输出结果：



1. 编程实现计算贷款偿还额的程序。根据用户输入的贷款信息，包括：年利率、贷款额、贷款年限，计算用户该笔贷款的月还款额和总还款额。

具体要求：设计一个Loan类表示贷款。这个类包括：

1. 一个名为annualInterestRate的私有double数据域，表示一笔贷款的年利率；
2. 一个名为numberOfYears的私有int数据域，表示一笔贷款的贷款年限；
3. 一个名为loanAmount的私有double数据域，表示一笔贷款的贷款额；
4. 一个名为loanDate的Date类型数据域，表示贷款创建的日期；

提示：Date类是JDK中的类，位于java.util包中，下面的代码创建了date对象，该对象保存了创建该对象的那一时刻所对应的时间；

Date date = new Date();

1. 一个以用户指定值创建一笔贷款的构造方法；

提示：在两个构造方法中都要创建账户创建日期对象；

1. 一个以默认值创建一笔贷款的无参构造方法；

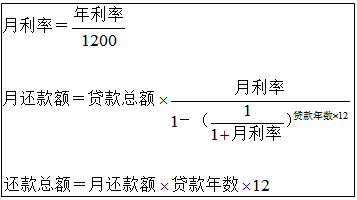
要求：通过this关键字调用上面的构造方法给成员变量赋值，各个成员变量的默认值自己随便写

1. 数据域loanDate的get方法，以及其他数据域的get方法和set方法；

思考：loanDate表示创建账号的日期，在构造方法中初始化，没有为其提供set方法，那loanDate中的日期，可以修改吗？

1. 一个名为getMonthlyPayment()的方法，计算并返回月还款额；
2. 一个名为getTotalPayment()的方法，计算并返回总还款额。

月还款额和总还款额的计算公式如下：



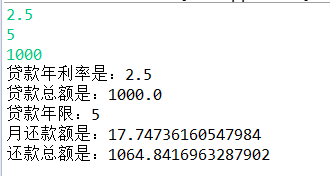
提示：使用JDK提供的Math类的**静态**方法pow计算指数，该方法的定义如下：

public static double pow(double a, double b)

该方法返回第一个参数的第二个参数次幂的值。

由于是静态方法，使用类名直接调用方法实现指数计算。

画出Loan类的UML图，并实现这个类。编写一个测试程序，根据用户输入的年利率、贷款年限和贷款额，计算该用户该笔贷款的月还款额和总还款额，按照如下样例输出。



1. 设计一个名为Account的银行账户类，包括：
2. 一个名为id的int类型私有账户数据域（默认值为0）。
3. 一个名为balance的double类型私有账户数据域（默认值为0）。
4. 一个能创建默认账户的无参构造方法。
5. 一个能创建带特定id和初始余额的账户的构造方法。
6. id、balance的访问器和修改器。
7. 一个名为withDraw的方法从账户提取特定数额，超过账户余额时，提示用户余额不足。
8. 一个名为deposit的方法向账户存储特定数额。

解题要求：实现测试类，在测试方法中，创建一个账户，使用withdraw方法取款，再使用deposit方法存款，然后打印账户和余额。

1. 设计一个名为MyPoint的类，代表一个以x坐标和y坐标表示的点。该类包括：
2. 一个创建点（0,0）的无参构造方法；
3. 以指定坐标构建点的构造方法；
4. 成员变量x和y的get和set方法；
5. 一个名为distance的静态方法，返回两个MyPoint对象之间的距离。

提示：

* 方法头：public static double distance(MyPoint p1, MyPoint p2)
* 调用Math类的sqrt方法计算平方根，方法定义如下：

public static double sqrt(double a) //返回正确舍入的 double 值的正平方根。

1. 一个名为distance的方法，返回从该点到MyPoint类型的指定点之间的距离；

提示：

* 方法头： public double distance(MyPoint p)
* 调用上面的静态distance方法计算该点和指定点之间的距离

1. 一个名为distance的方法，返回从该点到指定x和y坐标的另一个指定点之间的距离；

提示：

* 方法头：public double distance(double x, double y)
* 调用上面的静态distance方法计算该点和指定点之间的距离

要求如下：

1. 所有成员变量用private修饰，所有方法用public修饰，画出该类的UML图并实现这个类；
2. 编写一个测试程序，分别测试3个distance方法，将测试程序和MyPoint类放在不同的包里；
3. 思考3个distance方法之间的关系。
4. 设计一个名为MyInteger的类，这个类包括：
5. 一个名为value的int类型数据域，存储这个对象表示的int值；
6. 一个为指定的int值创建MyInteger对象的构造方法；
7. 一个返回int值的获取方法；
8. 如果对象中的值分别为偶数、奇数或素数，那么实例方法isEven()、isOdd()和isPrime()方法会分别返回true；

提示：分别调用下面的静态方法即可

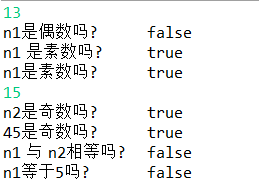
1. 如果指定的值分别为偶数、奇数或素数，那么相应的静态方法（类方法）isEvent(int)、isOdd(int)和isPrime(int)会分别返回true。
2. 如果指定的值分别为偶数、奇数或素数，那么相应的静态方法（类方法）isEvent(MyInteger)、isOdd(MyInteger)和isPrime(MyInteger)会分别返回true；
3. 如果该对象的值与指定的值相等，那么equals(int)和equals(MyInteger)方法返回true。

思考：MyInteger类中已经有分别用于判断偶数、奇数和素数的非静态方法，为什么还要提供静态的方法？这些静态方法可以怎么使用？

下面的测试类：

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** TestMyInteger {  **public** **static** **void** main(String[] args) {  Scanner input = **new** Scanner(System.***in***);  MyInteger n1 = **new** MyInteger(input.nextInt());  System.***out***.println("n1是偶数吗? " + '\t' + n1.isEven());  /\*  \* 下面分别用非静态方法和静态方法，判断你n1是否为素数  \*/  System.***out***.println("n1 是素数吗? " + '\t' + n1.isPrime());  System.***out***.println("n1是素数吗? " + '\t'+ MyInteger.*isPrime*(n1 ));      MyInteger n2 = **new** MyInteger(input.nextInt());  System.***out***.println("n2是奇数吗? " + '\t' + n2.isOdd());  System.***out***.println("45是奇数吗? " + '\t' + MyInteger.*isOdd*(45));  System.***out***.println("n1 与 n2相等吗? " + '\t' + n1.equals(n2));  System.***out***.println("n1等于5吗? " + '\t' + n1.equals(5));  }  } |

要求：所有成员变量用private修饰，所有方法用public修饰，画出该类的UML类图并实现这个类。下面是输入输出样例。



# **实验过程**

【此部分写程序，主要思路与关键代码需要以注释方式加以详细说明】

1.

**public** **class** TV {

**public** **int** channel; //定义变量

**public** **int** volumeLevel;

**public** **boolean** on;

**public** TV() //构造方法

{

}

**public** **void** turnOn() //变量的修改与获得

{

**this**.on=**true**;

}

**public** **void** turnOff()

{

**this**.on=**false**;

}

**public** **void** setChannel(**int** newChannel)

{

**if**(on)

channel=newChannel;

}

**public** **int** getChannel()

{

**return** channel;

}

**public** **void** setVolumn(**int** newVolumn)

{

**if**(on)

volumeLevel=newVolumn;

}

**public** **int** getVolumn()

{

**return** volumeLevel;

}

**public** **void** channelDown()

{

**if**(on)

**if**(channel>1)

channel=channel-1;

}

**public** **void** channelUP()

{

**if**(on)

**if**(channel<120)

channel=channel+1;

}

**public** **void** volumeUp()

{

**if**(on)

**if**(volumeLevel<7)

volumeLevel++;

}

**public** **void** volumeDown()

{

**if**(on)

**if**(volumeLevel>1)

volumeLevel--;

}

}

**import** java.util.Scanner;

**public** **class** testTV **extends** TV{

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

{

TV tv1=**new** TV();

tv1.turnOn();

Scanner sc=**new** Scanner(System.***in***);

tv1.setVolumn(sc.nextInt()); //输入TV参数

tv1.setChannel(sc.nextInt());

sc.close();

System.***out***.println("tv的音量和频道为"+tv1.getVolumn()+"和"+tv1.getChannel());

tv1.channelUP(); //调节TV参数

tv1.volumeDown();

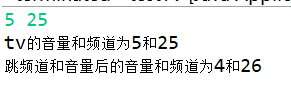
System.***out***.println("调频道和音量后的音量和频道为"+tv1.getVolumn()+"和"+tv1.getChannel());

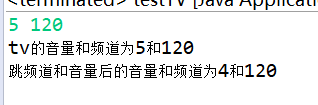
tv1.turnOff();

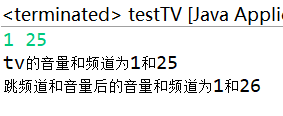
}

}

运行展示







2.

**public** **class** BMI {

**private** String name; //变量定义

**private** **int** age;

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

**private** **double** height;

**double** x;

**public** BMI() //三种构造方法

{

}

**public** BMI(String name,**int** age,**double** weight,**double** height)

{

**this**.name=name;

**this**.age=age;

**this**.weight=weight;

**this**.height=height;

}

**public** BMI(**double** weight,**double** height)

{

**this**.weight=weight;

**this**.height=height;

}

**public** String getName() { //变量修改与获取

**return** name;

}

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

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

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

**return** weight;

}

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

**this**.weight = weight;

}

**public** **double** getHeight() {

**return** height;

}

**public** **void** setHeight(**double** height) {

**this**.height = height;

}

**public** **double** getBMI()

{

**return** weight/(height\*height);

}

**public** String getStatus() //判断Status

{

**if**(getBMI()>=30.0)

**return** "肥胖";

**else** **if**(getBMI()>=25.0)

**return** "超重";

**else** **if**(getBMI()>=18.5)

**return** "正常";

**else**

**return** "低重";

}

}

**import** java.util.Scanner;

**public** **class** testBMI {

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

{

BMI a=**new** BMI();

Scanner sc=**new** Scanner(System.***in***);

a.setName(sc.next()); //数据输入

a.setAge(sc.nextInt());

a.setHeight(sc.nextDouble());

a.setWeight(sc.nextDouble());

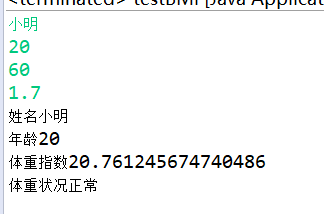
System.***out***.print("姓名"+a.getName()+"\n年龄"+a.getAge()+"\n体重指数"+a.getBMI()+"\n体重状况"+a.getStatus());

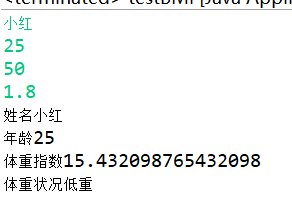
sc.close();

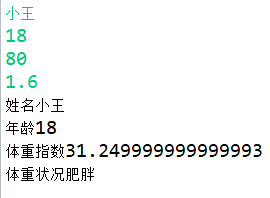
}

}

运行展示







3.

**public** **class** Rectangle {

**static** **int** *numberOfRectangle*; //变量定义

**double** width=1,height=1;

**public** Rectangle() //两种构造方法

{

}

**public** Rectangle(**double** width,**double** height)

{

**this**.width=width;

**this**.height=height;

}

**public** **double** getWidth() { //变量修改与获取

**return** width;

}

**public** **void** setWidth(**double** width) {

**this**.width = width;

}

**public** **double** getHeight() {

**return** height;

}

**public** **void** setHeight(**double** height) {

**this**.height = height;

}

**double** getArea()

{

**return** width\*height;

}

**double** getPerimeter()

{

**return** 2\*(width+height);

}

**static** **int** getNumberOfRectangle()

{

**return** *numberOfRectangle*;

}

}

**import** java.util.Scanner;

**public** **class** testRectangle {

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

{

Scanner sc=**new** Scanner(System.***in***);

Rectangle.*numberOfRectangle*++; //第一个r1对象的输入

Rectangle r1=**new** Rectangle(sc.nextDouble(),sc.nextDouble());

System.***out***.print("宽为"+r1.width+"\t高为 "+r1.height+"\t面积为 "+r1.getArea()+"\t周长为 "+r1.getPerimeter()+"\t矩形对象的个数为"+Rectangle.*getNumberOfRectangle*()+"\n");

Rectangle.*numberOfRectangle*++; //第二个r2对象的输入

Rectangle r2=**new** Rectangle();

r2.setWidth(sc.nextDouble());

r2.setHeight(sc.nextDouble());

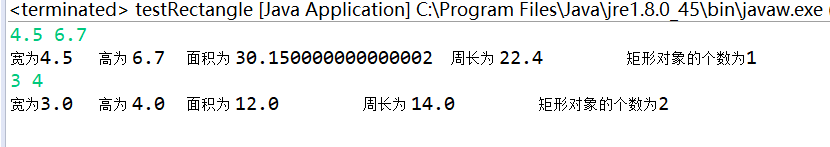
System.***out***.print("宽为"+r2.width+"\t高为 "+r2.height+"\t面积为 "+r2.getArea()+"\t周长为 "+r2.getPerimeter()+"\t矩形对象的个数为"+Rectangle.*getNumberOfRectangle*());

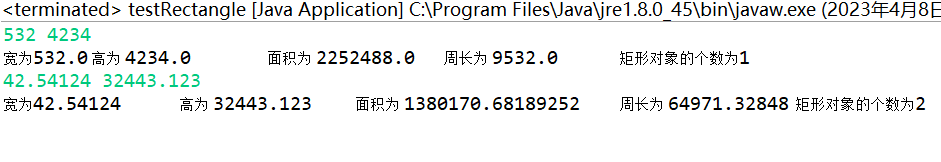
sc.close();

}

}

运行展示





4.

**import** java.lang.Math;

**import** java.util.\*;

**import** java.util.Scanner;

**public** **class** Loan {

**private** **double** annuallnteresRate; //变量定义

**private** **int** numberOfYears;

**private** **double** loanAmount;

Date loanDate;

**public** Loan() //两种构造方法

{

Date date=**new** Date();

loanDate=date;

}

**public** Loan(**double** a,**int** b,**double** c)

{

Date date=**new** Date();

loanDate=date;

**this**.annuallnteresRate=a;

**this**.numberOfYears=b;

**this**.loanAmount=c;

}

**public** **double** getAnnuallnteresRate() { //变量修改与获取

**return** annuallnteresRate;

}

**public** **void** setAnnuallnteresRate(**double** annuallnteresRate) {

**this**.annuallnteresRate = annuallnteresRate;

}

**public** **int** getNumberOfYears() {

**return** numberOfYears;

}

**public** **void** setNumberOfYears(**int** numberOfYears) {

**this**.numberOfYears = numberOfYears;

}

**public** **double** getLoanAmount() {

**return** loanAmount;

}

**public** **void** setLoanAmount(**double** loanAmount) {

**this**.loanAmount = loanAmount;

}

**public** **double** getMonthlyPayment(Loan a) //计算MonthlyPayment

{

**double** i;

i=(a.annuallnteresRate/1200)\*a.loanAmount;

**double** x=Math.*pow*(1+a.annuallnteresRate/1200,a.numberOfYears\*12);

**double** y=(**double**)(1/x);

**double** p=1.0-y;

**double** l=i/p;

**return** l;

}

**public** **double** getTotalPayment(Loan a)

{

**return** a.getMonthlyPayment(a)\*a.numberOfYears\*12;

}

**public** Date getLoanDate()

{

**return** loanDate;

}

}

**import** java.util.Scanner;

**public** **class** testLoan {

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

{

**double** a,c;

**int** b;

Scanner sc=**new** Scanner(System.***in***);

a=sc.nextDouble(); //对象输入数据

b=sc.nextInt();

c=sc.nextDouble();

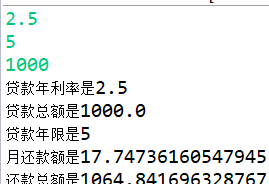
sc.close();

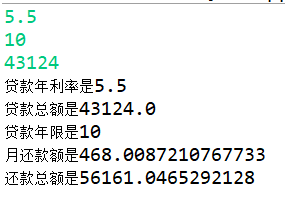
Loan loan=**new** Loan(a,b,c);

System.***out***.print("贷款年利率是"+loan.getAnnuallnteresRate()+"\n贷款总额是"+loan.getLoanAmount()+"\n贷款年限是"+loan.getNumberOfYears()+"\n月还款额是"+loan.getMonthlyPayment(loan)+"\n还款总额是"+loan.getTotalPayment(loan));

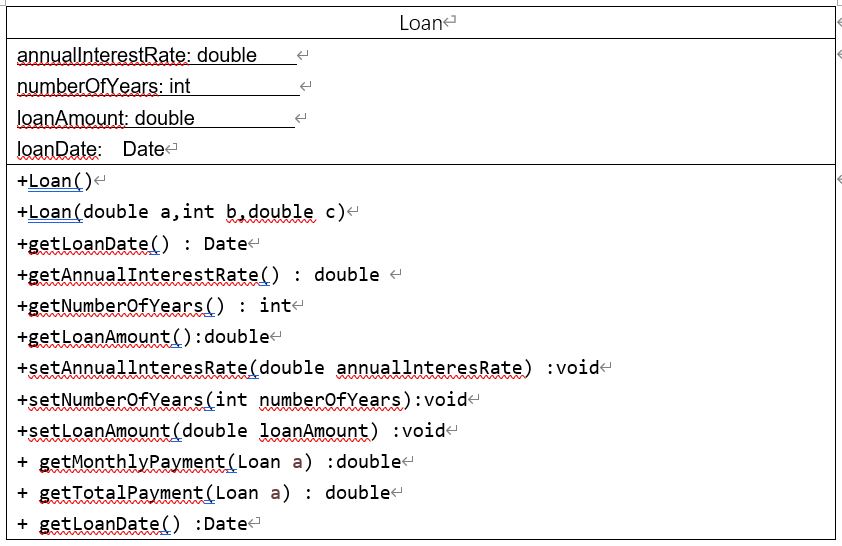
}

}





UML图



5.

**public** **class** Account {

**private** **int** id=0; //变量定义

**private** **double** balance=0;

**public** Account() //两种构造方法

{

}

**public** Account(**int** id,**double** balance)

{

**this**.id=id;

**this**.balance=balance;

}

**public** **int** getId() { //变量修改与获取

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** **double** getBalance() {

**return** balance;

}

**public** **void** setBalance(**double** b) {

balance = b;

}

**public** **void** withdraw(**double** b)

{

**if**(b>**this**.balance) //进行余额判断

{

System.***out***.print("余额不足");

}

**else**

{

balance=balance-b;

System.***out***.print("取款后余额是"+getBalance());

}

}

**public** **void** deposit(**double** b)

{

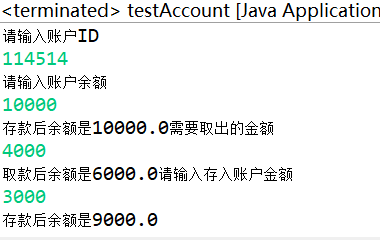
balance=balance+b;

System.***out***.print("存款后余额是"+getBalance());

}

}

运行展示



6.

**package** point;

**public** **class** MyPoint {

**private** **double** x; //变量定义

**private** **double** y;

**public** MyPoint(){ //两种构造方法

x=0;

y=0;

}

**public** MyPoint(**double** a,**double** b)

{

**this**.x=a;

**this**.y=b;

}

**public** **double** getX() { //变量修改与获取

**return** x;

}

**public** **void** setX(**double** x) {

**this**.x = x;

}

**public** **double** getY() {

**return** y;

}

**public** **void** setY(**double** y) {

**this**.y = y;

}

**public** **static** **double** distance(MyPoint p1, MyPoint p2){

**double** a=p1.getX(); //三种计算distance方法

**double** b=p1.getY();

**double** q=p2.getX();

**double** w=p2.getY();

**return** Math.*sqrt*((a-q)\*(a-q)+(b-w)\*(b-w));

}

**public** **double** distance(MyPoint p){

**double** a=**this**.x;

**double** b=**this**.y;

**double** q=p.getX();

**double** w=p.getY();

**return** Math.*sqrt*((a-q)\*(a-q)+(b-w)\*(b-w));

}

**public** **double** distance(**double** x,**double** y){

**double** a=**this**.x;

**double** b=**this**.y;

**double** q=x;

**double** w=y;

**return** Math.*sqrt*((a-q)\*(a-q)+(b-w)\*(b-w));

}

}

**package TestMyPoint;**

**import point.MyPoint;** //引入point包中的MyPoint类

**import java.util.Scanner;**

**public class TextMyPoint {**

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

**Scanner input=new Scanner(System.in);**

**double a=input.nextDouble();** //数据输入与对象创建和三种distance

**double b=input.nextDouble();**

**double q=input.nextDouble();**

**double w=input.nextDouble();**

**MyPoint p=new MyPoint();**

**MyPoint p1=new MyPoint();**

**MyPoint p2=new MyPoint();**

**p1.setX(a);**

**p1.setY(b);**

**p2.setX(q);**

**p2.setY(w);**

**p.setX(12.3);**

**p.setY(32.1);**

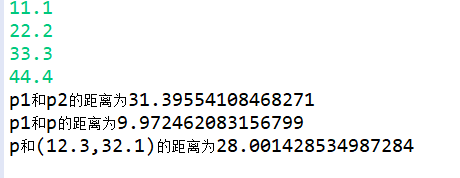
**System.out.println("p1和p2的距离为"+MyPoint.distance(p1,p2));**

**System.out.println("p1和p的距离为"+p1.distance(p));**

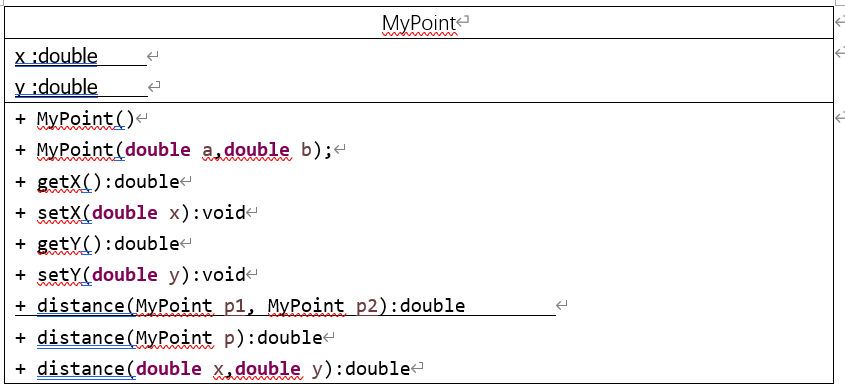
**System.out.println("p和(12.3,32.1)的距离为"+p.distance(32.1,12.3));**

**}**

**}**运行展示



UML图



思考题：

第一个：公共静态方法，可以不借助于对象即可调用的方法，其中参数为两个对象。

第二个：公共方法，参数为MyPoint对象，必须要在对象下进行调用，且参数必须为一个MyPoint对象

第三个:公共方法，参数为x,y即两个点的坐标，必须在对象下进行调用，且参数必须为两个double类型的坐标。

第一个与第二个第三个区别在于不需要依附与对象，可以直接调用。

第二个与第三个区别在于，参数一个为MyPoint对象，一个为x,y坐标点。若已知MyPoint对象用第二个，没有MyPoint对象而知道点的坐标时，用第三个。

7.

**public** **class** MyInteger {

**private** **int** value;

**public** **void** MyInteger(){ //两种构造方法

}

**public** **void** MyInteger(**int** q){

**this**.value=q;

}

**public** **boolean** isEven(){ //三种无参判断方法

**if**(**this**.value%2==0)

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** isOdd(){

**if**(**this**.value%2!=0)

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** isPrime(){

**int** i;

**for**(i=2;i<=**this**.value;i++)

{

**if**(**this**.value%i==0)

{

**break**;

}

}

**if**(i==**this**.value)

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** isEven(**int** a){ //三种有参判断方法

**if**(a%2==0)

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** isOdd(**int** b){

**if**(b%2!=0)

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** isPrime(**int** c){

**int** i;

**for**(i=2;i<=c;i++)

{

**if**(c%i==0)

{

**break**;

}

}

**if**(i==c)

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** equals(**int** d){

**if**(**this**.value==d)

**return** **true**;

**else**

**return** **false**;

}

**public** **int** getValue() {

**return** **this**.value;

}

**public** **static** **boolean** isEven(MyInteger c1){ //三种静态判断方法

**if**(c1.getValue()%2==0)

**return** **true**;

**else**

**return** **false**;

}

**public** **static** **boolean** isOdd(MyInteger c1){

**if**(c1.getValue()%2!=0)

**return** **true**;

**else**

**return** **false**;

}

**public** **static** **boolean** isPrime(MyInteger c1){

**int** i;

**for**(i=2;i<=c1.getValue();i++)

{

**if**(c1.getValue()%i==0)

{

**break**;

}

}

**if**(i==c1.getValue())

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** equals(MyInteger c1){

**if**(**this**.value==c1.value)

**return** **true**;

**else**

**return** **false**;

}

}

**import** java.util.Scanner;

**public** **class** testMyInteger {

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

Scanner input=**new** Scanner(System.***in***);

**int** a=input.nextInt(); //对象创建和数据输入

MyInteger n1=**new** MyInteger();

n1.MyInteger(a);

System.***out***.println("n1是偶数吗？"+"\t"+n1.isEven());

System.***out***.println("n1是素数吗？"+"\t"+n1.isPrime());

System.***out***.println("n1是素数吗？"+"\t"+MyInteger.*isPrime*(n1));

MyInteger n2=**new** MyInteger(); //对象创建和数据输入

**int** b=input.nextInt();

n2.MyInteger(b);

System.***out***.println("n2是奇数吗？"+"\t"+n2.isOdd());

System.***out***.println("45是奇数吗？"+"\t"+n1.isOdd(45));

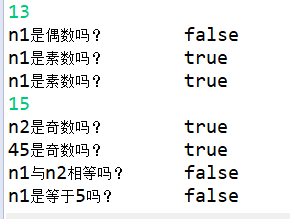
System.***out***.println("n1与n2相等吗？"+"\t"+n1.equals(n2));

System.***out***.println("n1是等于5吗？"+"\t"+n1.equals(5));

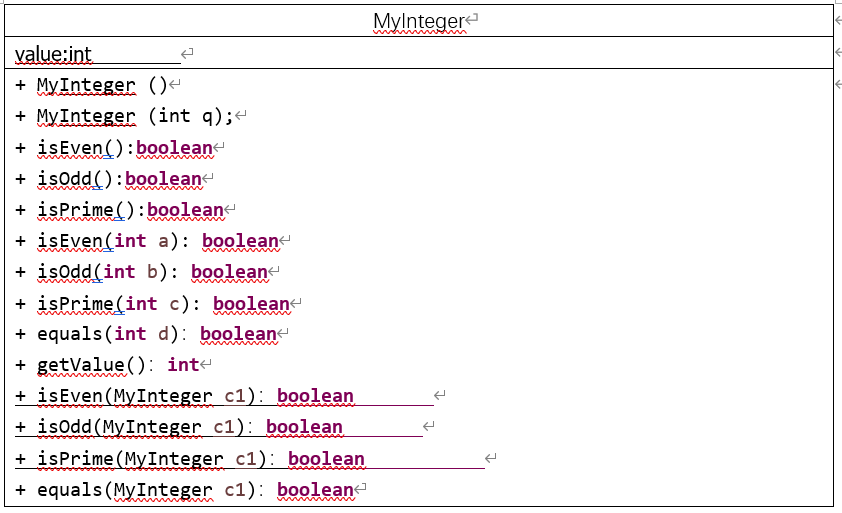
}

}

运行展示



UML图



思考：MyInteger类中已经有分别用于判断偶数、奇数和素数的非静态方法，为什么还要提供静态的方法？这些静态方法可以怎么使用？

MyInteger类中所提供的判断数字累心改动非静态方法是在自己类中可以使用的方法，可以用于自身成员变量的调用，不需要形参。而那些静态的判断方法，在调用时可以不需要有类的构造的前提下，可以通过直接赋予一个指定的形参来进行判断，还可以通过调用已知的类名，来判断其类型，不需要类的支持下。

# **实验总结**

【此部分需要分别描述（1）程序涉及的知识点；（2）编程过程中遇到的技术困难及解决方法。上述内容需要按照列表的形式一一列出。要求总结得当，专业术语准确，描述详细。

不要写类似于：“本次实验非常难，在老师和同学的帮助下，……，最终完成。”此类非学术非技术同时缺乏细节梳理与描述的文字。】

|  |  |  |
| --- | --- | --- |
| 实验编号 | 程序设计的知识点 | 编程过程遇到的技术困难及解决方法 |
| 1 | 对象的创建和构造方法的建立，对象内变量的设置于调用和改变的方法。以及对象变量界限的设置。 | 在设置和调用变量值的时候，发现无法改变变量的初值，之后上网查了之后，发现需要调用set和get函数来改变和调用对象内的变量。在改变变量的时候也可以用this关键字 |
| 2 | 对象的创建和构造方法的建立，对象内变量的设置于调用和改变的方法。  私有数据域的建立和调用，和计算方法。变量的精度控制 | 在进行计算方法的设计时，出现了预计结果和实际结果不想匹配。通过上网查询了之后，发现是精度的问题，在设置变量的时候，把weight和height设置为了int类型，因为BMI需要返回一个double类的结果。所以导致了精度出错。把weight和height定义时就定义为double类或者在计算的时候变量前面加上（double）来强制改变变量的类型，从而使精度达到一致 |
| 3 | 对象的创建和构造方法的建立，对象内变量的设置于调用和改变的方法。  计算方法的设计与指定对象的构造方法。以及静态方法和静态变量的设计。 | 在设计静态变量和静态方法的时候，一开始静态变量的定义没有用static来修饰导致了其类型出错变成了成员变量，不能在静态方法中直接调用。以及静态方法中不能直接引用成员变量，需要类名修饰。 |
| 4 | 对象的创建和构造方法的建立，对象内变量的设置于调用和改变的方法。  私有数据域的建立。以及一个复杂的计算方法的设计和Math类里sqrt的静态方法和Date类中的时间方法的引用 | 在设计复杂计算方法的时候，其括号太多，导致在计算的时候经常发生顺序错误或者精度错误。通过不断的调试与改动，最后改出了一个的正确的计算顺序的方法。在调用sqrt和时间方法中时，需要引用的类名不能在类中途在引用。需要在类外最开始处才有效 |
| 5 | 对象的创建和构造方法的建立，对象内变量的设置于调用和改变的方法。  私有数据域的建立，无参和指定参数的构造方法。以及变量的修改器设置最值的判断 | 再进行withdraw和deposit方法对其成员变量进行成员更改的时候，会出现成员变量不变的情况。经过调试发现，原来在进行变量更改的时候，改变的不是成员变量的参数，而是方法的形参，从而导致了成员变量不变的情况 |
| 6 | 对象的创建和构造方法的建立，对象内变量的设置于调用和改变的方法。以及在不同包之间调用方法的方法 | 在进行跨包调用时，报了个修改不了的错误，观察后猜测应该是包名和类名重复导致的。把包名和类名改成不一样的之后，程序就能正常运行了 |
| 7 | 对象的创建和构造方法的建立，对象内变量的设置于调用和改变的方法。  静态方法与静态变量的建立。和方法的重构。 | 在进行实例方法和静态方法的设计时，需要改变的部分仅仅是变量的引用。在调用变量的时候，需要保持和形参相同类型的变量使用方式。不能把不同类型的形参和方法搞混，否则方法就是错误的 |