实验 5

类 与 对 象

实验目的

(1) 熟练掌握类的定义方法,包括构造方法、属性及普通方法的定义;

(2) 掌握静态属性、最终属性的应用,理解静态属性与实例属性(非静态属性)的区别;

(3) 掌握方法的定义和调用,理解方法调用时的参数传递过程;

(4) 重点理解类与对象的关系、引用变量与普通变量的区别;

(5) 掌握包的定义和使用方法,能够基于包进行成员的访问控制;

(6) 培养对特定问题中的数据和功能进行抽象建模的能力。

5.1 类 的 设 计

练习题 5-1: 学生类 v1

【内容】

定义一个学生类 Student,包含:

- 3 个属性: 学号(no)、姓名(name)和年龄(age);
- 两个构造方法: 无参构造方法,以及三参数(name、age、no)的构造方法;
- 一个普通方法: 输出当前对象各属性值的 print()方法。

【思路】

① 确定属性的类型: 姓名和学号应为 String 类型,年龄应为 int 类型,定义如下:

```
String no, name;
int age;
```

② 按照构造方法的规则,无参构造方法定义如下:

```
public Student() {}
```

有参构造方法中的 3 个参数分别为当前对象的 3 个属性赋值,定义如下:

```
public Student(String no, String name, int age) {
    this.no =no;
    this.name =name;
    this.age =age;
}
```

③ print()方法的功能是输出当前对象各属性值,不需要参数,也不需要返回结果,定义如下:

```
void print() {
    System.out.printf("No.% s, Name:% s, Age:% d\n",this.no, this.name,
                      this.age);
}
```

④ 可以在 Student 类中添加 main()方法,用于创建对象并测试。

【程序代码】

```
class Student {
    String no, name;
    int age;
    public Student() {    }
    public Student(String no, String name, int age) {
        this.no =no;
        this.name =name;
        this.age =age;
    }
    void print() {
        System.out.printf("No.%s, Name:%s, Age:%d\n", this.no, this.name,
                          this.age);
    }
}
public class TestStudent{
    public static void main(String[] args) {
        Student stu =new Student("001", "Mike", 10);
        stu.print();
    }
}
```

【运行结果】

```
No.001, Name:Mike, Age:10
```

【思考】

在此例中,如果需要修改对象 stu 的属性值,该如何实现?

自测题 5-1: 简易计算器

【内容】

设计一个简易计算器类 Calculator,用来对两个整型属性 a 和 b 进行多种运算,包含:

(1) 两个属性 a 和 b,用来进行运算的数据;

(2) 两个构造方法:无参构造方法,以及两参数的构造方法对属性 a 和 b 赋初值;

(3) 对属性 a 和 b 进行运算的 4 个方法：add()(求和)、sub()(求差)、multiply()(求积)、divide()(求商)。

设计 Java 类 TestCalculator，在其 main()方法中根据输入的两个整数创建一个 Calculator 对象，在屏幕上输出两个整数的和、差、积、商。要求商保留两位小数。如果输入的除数为 0，则输出 error。

自测题 5-2：电视机类

【内容】

每台电视机都是一个对象，有多个状态(当前频道、当前音量、是否打开)，有多种操作(转换频道、调节音量、打开、关闭等)。可以设计 Television 类对电视机进行建模，其 UML 图如图 5-1 所示。

Television
−channel : int −volume : int −on : boolean
+Television() +Television(in chan : int, in vol : int, in isOn : boolean) +turnOn() +turnOff() +setChannel(in newChannel : int) +channelUp() +channelDown() +setVolume(in newVolume : int) +volumeUp() +volumeDown()

图 5-1　电视机类的设计

其中，channelUp()和 channelDown()分别实现开机状态下频道号加 1 和减 1，最小为 1，最大为 300；volumeUp()和 volumeDown()分别实现开机状态下音量加 1 和减 1，最小为 0，最大为 10。

设计 Java 类 TestTelevision，在其 main()方法中依次读取频道号、音量和开关机状态，创建一个 Television 对象，按顺序调用 turnOn()、volumeDown()和 channelUp()来调整该对象的频道和音量，最后调用 print()在屏幕上输出其状态。若输入的频道号和音量不合要求，直接提示 error。

自测题 5-3：股票类 v1

【内容】

对股票交易市场中的每只股票进行建模。每只股票都是一个对象，有股票代码、股票名字、股票总量、前一交易日收盘价、当前股价、当前是否可以交易等多个状态，有多种操作(计算总市值、计算当日市值增长率、输出信息等)。可以设计 Stock 类对股票进行建模，其 UML 图如图 5-2 所示。

设计 Java 类 TestStock，在其 main()方法中根据输入的股票代码、名称、总量、收盘

Stock
–stockCode : String –stockName : String –totalShares : int –previousClosingPrice : double –currentPrice : double –isTradable : boolean
+Stock() +Stock(in code : String, in name : String, in shares : int, in priviousClosingPrice : double, in currentPrice : double) +getMarketValue() : double +getChangePercent() : double +print()

图 5-2　股票类的设计

价和当前价格来创建一个 Stock 对象，调用以上方法来计算该股票的总市值和当日市值增长率，最后在屏幕上输出该股票的信息。输入输出的运行结果如下。

```
stock code:000012↙
stock name:NBA↙
stock shares(万):310820↙
stock closing price:7.38↙
stock current price:7.65↙
Total market value(万元):2377773.0
Changed percent:3.66%
```

5.2　创建对象与构造方法

练习题 5-2：学生类 v2

【内容】

为练习题 5-1 中的 Student 类设计以下多个构造方法，形成 Student2 类，实现创建对象的多种方式。

（1）无参构造方法：设置学号为 000，name 为 null，年龄为 0；

（2）三参数的构造方法：设置对象的 name、age、no 属性值；

（3）两参数的构造方法：设置对象的 no 和 name 属性值，年龄为 0；

（4）两参数的构造方法：设置对象的 no 和 age 属性值，name 为 untyped；

（5）单参数的构造方法：设置对象的 no 属性值，name 为 untyped，age 为 0。

定义 TestStudent2 类，在其 main()中创建对象并输出其信息。

【思路】

① 新建 Student2 类，在练习题 5-1 中 Student 类的基础上修改代码；

```
String name, no;
int age;
```

② 在 Java 继承机制下，创建子类对象时，Java 系统会默认调用父类的无参构造方法。如果一个类中显式定义了构造方法，系统就不会再提供默认的无参构造方法。为了避免子类对象创建时找不到无参构造方法的潜在风险，通常为类定义显式的无参构造方法。

③ 按照题目要求，无参构造方法定义如下：

```
public Student2() {
    this.name ="null";
    this.no ="000";
    this.age =0;
}
```

④ 三参数的构造方法中的 3 个参数分别为当前对象的 3 个属性赋值，定义如下：

```
public Student2(String no, String name, int age) {
    this.no =no;
    this.name =name;
    this.age =age;
}
```

⑤ 两参数的构造方法设置对象的 no 和 name 属性值。为了防止在多个构造方法中重复地出现赋值语句，可以在构造方法的首行使用 this()来调用本类的其他构造方法，实现代码复用。定义如下：

```
public Student2(String no, String name) {
    this(no, name, 0);  // 使用 this()调用三参数的构造方法，age 值设为默认的 0
}
```

⑥ 其他构造方法类似定义。

⑦ 创建对象时，系统根据提供的实际参数自动调用相应的构造方法，进行对象的初始化工作。

【程序代码】

```
class Student2 {
    String no, name;
    int age;
    public Student2() {
        this.no ="000";
        this.name ="null";
        this.age =0;
    }
    public Student2(String no, String name, int age) {
        this.no =no;
```

```
        this.name =name;
        this.age =age;
    }
    public Student2(String no, String name) {
        // 使用 this()调用三参数的构造方法,age 值设为 0
        this(no, name, 0);
    }
    public Student2(String no, int age) {
        // 使用 this()调用三参数的构造方法,name 值设为"untyped"
        this(no, "untyped", age);
    }
    public Student2(String no) {
        //调用三参数的构造方法,name 值设为"untyped",age 值设为 0
        this(no, "untyped", 0);
    }
    void print() {
        System.out.printf("No.%s, Name:%s, Age:%d\n", this.no, this.name,
                          this.age);
    }
}
public class TestStudent2{
    public static void main(String[] args) {
        // 3 个实参,自动调用三参数的构造方法
        Student2 s1 =new Student2("001", "Mike", 10);
        s1.print();
        // 2 个 String 实参,自动调用两参数的构造方法
        Student2 s2 =new Student2("002", "John");
        s2.print();
        // 1 个 String 实参,自动调用单参数的构造方法
        Student2 s3 =new Student2("003");
        s3.print();
    }
}
```

【运行结果】

```
No.001, Name:Mike, Age:10
No.002, Name:John, Age:0
No.003, Name:untyped, Age:0
```

【思考】

若已经定义好 public Student2(String no, String name) {…},能否在其基础上实现三参数的构造方法 public Student2(String no, String name, int age)?

注意:如果在构造方法中使用了 this()互相调用,必须保证至少有一个构造方法是

未使用 this()语句的，否则会出现"Recursive constructor invocation Student2(String, String, int)"的错误。

自测题 5-4：矩形类 v1

【内容】

对矩形进行建模。每个矩形都是一个对象，设计矩形类 Rectangle，其 UML 图如图 5-3 所示。

Rectangle
−height : int −width : int −x : int −y : int
+Rectangle() +Rectangle(in width : double, in height : double) +Rectangle(in width : double, in height : double, in x : int, in y : int) +print()

图 5-3　矩形类的设计

另设计一个 Java 类 TestRectangle，在其 main()方法中创建多个 Rectangle 对象，在屏幕上输出对象的各项信息。

自测题 5-5：股票类 v2

【内容】

在自测题 5-3 中的股票类的基础上，设计 Stock2 类，其 UML 图如图 5-4 所示。

Stock2
−stockCode : String −stockName : String −totalShares : int −previousClosingPrice : double −curentPrice : double −isTradable : boolean
+Stock2() +Stock2(in code : String, in name : String) +Stock2(in code : String, in name : String, in curPrice : double) +Stock2(in code : String, in name : String, in shares : int, in curPrice : double) +Stock2(in code : String, in name : String, in shares : int, in prePrice : double, in curPrice : double) +Stock2(in code : String, in name : String, in prePrice : double, in curPrice : double) +Stock2(in code : String, in name : String, in shares : int, in prePrice : double, in curPrice : double, in tradable : boolean) +getMarketValue() : double +getChangePercent() : double +print()

图 5-4　股票类 v2 的设计

特别注意构造方法的定义。对象属性的默认值按表 5-1 设计。

设计 Java 类 TestStock2，在其 main()方法中创建多个 Stock2 对象，在屏幕上输出对象的各项信息。

表 5-1 Stock2 类的属性默认值

属性	默认值	属性	默认值
totalShares	0	currentPrice	0.0
previousClosingPrice	0.0	isTradable	false

5.3 方法的设计：代码封装

练习题 5-3：二维空间的点

【内容】

二维空间的每个点都是一个对象，定义 Point2D 类进行建模，其成员如下：

(1) 属性 x 和 y：实型，表示点的 x 和 y 坐标。

(2) 无参构造方法：置 x 和 y 为 0。

(3) 两参数构造方法：初始化 x 和 y。

(4) getDistance()方法：计算当前点与指定点之间的距离。

(5) getQuadrant()方法：获取当前点在第几象限，返回整型 1、2、3、4，若在坐标轴上则返回 0。

在 TestPoint2D 类的 main()中定义两个 Point2D 对象 d1 和 d2，并调用方法计算两点之间的距离(要求输出两位小数)，计算 p2 点在第几象限。

【思路】

① 方法代表对象所具有的动态行为或操作，实现该行为或操作的具体代码被封装在方法体中，通过方法名和参数来对外提供被访问的途径，并通过方法返回值来体现行为或操作的结果，因此需要根据具体问题进行特定方法的设计。

② getDistance()方法的功能是计算当前点和指定点之间的距离：当前点即当前对象，还需要为该方法提供另一个点参与运算，即该方法需要一个 Point2D 类型的形参；该方法应得到一个距离值(应为实型)作为结果，即方法的类型应为实型。由此得到方法的整体结构：

```
double getDistance(Point2D p) {
    // 该功能的具体实现代码
}
```

方法体内是计算当前点和参数点 p 之间距离的具体代码，此处采用欧氏距离计算，得到的结果需要通过 return 语句返回，代码如下：

```
double getDistance(Point2D p) {
    double d;
    // 计算距离值
    d =Math.sqrt((x -p.x) * (x -p.x) +(y -p.y) * (y -p.y));
```

```
        return d; // 返回结果
    }
```

③ getQuadrant()方法的功能是获取当前点在第几象限：当前点即当前对象，即该方法无须形参；该方法应得到一个象限编号作为结果，即方法类型应为整型。由此得到方法的整体结构：

```
int getQuadrant() {
    // 该功能的具体实现代码
}
```

方法体内是计算象限编号的具体代码，如下：

```
int getQuadrant() {
    if (x >0 && y >0)     return 1;
    else if (x <0 && y >0)     return 2;
    else if (x <0 && y <0)     return 3;
    else if (x >0 && y <0)     return 4;
    else     return 0;
}
```

④ 一旦创建好对象，即可调用该对象的各个方法来实现相应的功能。

【程序代码】

```
class Point2D {
    double x, y;
    public Point2D() {}
    public Point2D(double x, double y) {
        this.x =x;
        this.y =y;
    }
    double getDistance(Point2D p) {
        double d;
        // 计算距离值
        d =Math.sqrt((x -p.x) * (x -p.x) +(y -p.y) * (y -p.y));
        return d; // 返回结果
    }
    int getQuadrant() {
        if (x >0 && y >0)
            return 1;
        else if (x <0 && y >0)
            return 2;
        else if (x <0 && y <0)
            return 3;
```

```
            else if (x > 0 && y < 0)
                return 4;
            else
                return 0;
        }
    }
    public class TestPoint2D{
        public static void main(String[] args) {
            Point2D p1 = new Point2D();             // 原点
            Point2D p2 = new Point2D(10, -5);       // 点(10,-5)
            //调用 getDistance()计算 p1 和 p2 的距离
            System.out.println(p1.getDistance(p2));
            //调用 getQuadrant()计算 p2 在第几象限
            System.out.println("p2 点在第" + p2.getQuadrant() + "象限");
        }
    }
```

【运行结果】

```
11.18
p2 点在第 4 象限
```

【思考】

在以上 Point2D 类的基础上定义方法来判断当前点和特定点是否在同一水平线上。请思考如何设置参数、方法类型等。

自测题 5-6：学生类 v3

【内容】

定义 Student3 类对每个学生进行建模，其成员如下。

(1) 属性 name 和 age：表示学生的姓名和年龄。

(2) 无参构造方法：置 name 为 null，age 为 0。

(3) 两参数构造方法：初始化 name 和 age。

(4) isSameAge()方法：比较当前学生和指定学生的年龄是否相等。

(5) print()方法：输出该学生的相关属性值。

设计 Java 类 TestStudent3 测试 Student3 类中定义的方法，并输出结果。

自测题 5-7：矩形类 v2

【内容】

对矩形进行建模。每个矩形都是一个对象，设计矩形类 Rectangle2，其 UML 图如图 5-5 所示。

其中，getArea()和 getPerimeter()方法分别计算矩形的面积和周长；draw()方法在