第 3 章

类 的 重 用

要点导读

本章内容需要与配套的主教材《Java 语言程序设计》(第 3 版)第 3 章配合学习。

主教材第 3 章介绍了类的重用。继承是面向对象程序设计的基石之一,是一种由已有类创建新类的机制。Java 要求声明的每个类都有超类,当没有显式指定超类时,超类隐含为 java.lang.Object 类。Java 不支持类的多重继承,只支持类的单继承,即每个子类只能有一个直接超类。一个对象继承的内容取决于此对象所属的类在类层次中的位置。一个对象从其所有的直接和间接超类中继承属性及行为。

隐藏和覆盖是指子类对从超类继承来的属性变量及方法可以重新定义。如果子类对从超类继承来的属性变量重新定义,则从超类继承的属性将被隐藏。如果子类在声明方法时,使用相同的方法名及参数表,但执行不同的功能,这种情况称为方法覆盖。

有继承时的构造方法遵循以下的原则:子类不能从超类继承构造方法;好的程序设计方法是在子类的构造方法中调用某一个超类构造方法;super 关键字也可以用于构造方法中,其功能为调用超类的构造方法;如果在子类的构造方法的声明中没有明确调用超类的构造方法,则系统在执行子类的构造方法时会自动调用超类的默认构造方法(即无参的构造方法);如果在子类的构造方法的声明中调用超类的构造方法,则调用语句必须出现在子类构造方法的第一行。

Object 类是 Java 程序中所有类的直接或间接超类,也是类库中所有类的超类,处在类层次最高点。如果两个对象具有相同的类型及相同的属性值,则称这两个对象相等(equal);如果两个引用变量指向的是同一个对象,则称这两个对象同一(identical)。

final 类是指被 final 修饰的类。final 类不可能有子类。final()方法是被 final 修饰的方法,不能被当前类的子类覆盖。

抽象类是指被 abstract 修饰的类,表示不能使用 new 进行实例化,即不能有具体实例对象。抽象类既可以包含抽象方法,也可以包含非抽象方法。抽象方法使用 abstract 修饰,有方法的声明,而没有方法的实现,并将在抽象类的子类中实现。不能在非抽象的类中声明抽象方法。

泛型的本质是参数化类型,即所操作的数据类型被指定为一个参数。这种数据类型的指定可以使用在类、接口以及方法中,分别称为泛型类、泛型接口和泛型方法。通配符泛型用参数“?”表示,代表任意一种类型。有限制的泛型是指对类型参数加上一定的限制,例如,必须是某类的子类或者实现了某接口。

Java 的类中可以有其他类的对象作为成员，这便是类的组合。

Java 提供了用于语言开发的类库，称为应用程序编程接口（Application Programming Interface，API），分别放在不同的包中。Java 提供的包主要有：java.lang、java.io、java.math、java.util、java.applet、java.awt、java.awt.datatransfer、java.awt.event、java.awt.image、java.beans、java.net、java.rmi、java.security、java.sql 等。

实验 3　类的重用

一、实验目的

（1）理解类的组合与继承，知道何时使用哪种重用机制。

（2）了解 final 类、final()方法、抽象类、抽象方法的概念。

（3）熟练掌握主教材第 3 章提到的 Java 基础类库中的一些常见类，学会查阅 JavaDoc。

（4）了解 Java 包的概念。知道为什么要用包，养成良好的命名习惯。

（5）初步了解 JAR 文件的概念和 jar 命令的格式。

二、实验任务

1. 继承关系举例

举出两个例子，各包含一个超类和它的三个子类。对于超类，要给出两个成员变量，一个方法的定义。要求两个例子一个是具体事物，另一个是抽象事物。不要使用教材中已有的例子，最好能和自己所学的专业有些关系。

例如：

具体事物：主教材 3.4.1 节中的 Shape、Circle、Triangle、Rectangle 的例子。超类成员变量可以有周长、面积。方法可以有 draw()，功能是把图形画出来。

抽象事物：超类，如银行相关交易类。子类，如存款交易、贷款交易、外汇交易。超类成员变量可以有交易时间、交易金额。方法可以有 save()，功能是把交易记录保存。

2. equals()方法

下面的代码实现了复数类和复数的减法。equals()方法是比较两个对象是否相等，每个 Java 类都有一个继承自 Object 类的默认的 equals()方法。下面的类有 main()方法，那它的执行结果是什么？为什么会有这样的结果？这个执行结果真的是我们期待的结果吗？重写 ComplexNumber 类的 equals()方法，使得它真正能够比较两个复数是否相等。你添加的 equals()方法应该使这个 main()方法返回应有的结果（true true true true false false）。

注意：equals()方法的参数是 Object 对象，所以应该考虑如果输入其他对象（例如 String）或者空对象（null）应该如何处理，应该避免 equals()方法出现什么样的异常。也可以使用自己正确编写的复数类。

```
public class ComplexNumber {
    double real = 0;

    double imagine = 0;
```

```
    public ComplexNumber(double real, double imagine) {
        this.real = real;
        this.imagine = imagine;
    }

    public ComplexNumber minus(ComplexNumber operand) {
        return new ComplexNumber(this.real - operand.real, this.imagine
                - operand.imagine);
    }

    public static void main(String[] args) {
        ComplexNumber complex1 = new ComplexNumber(2.02d, 3.1d);
        ComplexNumber complex2 = new ComplexNumber(2d, 3d);
        ComplexNumber complex3 = complex2;
        ComplexNumber complex4 = new ComplexNumber(2d, 3d);
        ComplexNumber complex5 = new ComplexNumber(0.02d, 0.1d);
        ComplexNumber complex6 = complex1.minus(complex2);

        System.out.println(complex2 == complex3);
        System.out.println(complex2.equals(complex3));
        System.out.println(complex2.equals(complex4));
        System.out.println(complex6.equals(complex5));
        System.out.println(complex1.equals(null));
        System.out.println(complex1.equals(new String("abc")));
    }
}
```

3. Java 类库常用包使用练习 1

输入一个包括 4 个小数的字符串，数之间用分号分隔，格式为“a;b;c;d”。计算如下表达式：

$$\sin a \times \cos b \times \sqrt{c^d}$$

返回与结果最接近的整数并按照格式输出计算时间。例如：

```
输入:
0.5;-0.8;3;6.3
输出:
2021-10-28 8:00:06
result = 11
```

如果输入的数字格式不正确，输出“Invalid Input”字符串。

提示： 使用 java.util.StringTokenizer，java.text.SimpleDateFormat，java.lang.Math 类。

4. 定义并实现一个记账软件的类

设计、定义、实现一个交易管理软件中会使用到的类。该软件可以记录和管理如下信息。

(1) 交易,包括日常收支交易、转账交易、投资买卖交易(外汇、股票)。

(2) 人员机构,包括人员、机构。

(3) 账户,包括活期账户、定期账户、信用卡账户、投资买卖交易账户。

定义所需类的超类、子类;合理地使用抽象类、final 类;合理地定义每个类的成员变量及其类型;合理地给出几个类的方法(无须实现方法)。

5. Java 类库常用包使用练习 2

用两种方法实现 encode()、decode()方法给字符串加解密。输出原始字符串、加密后的字符串和解密后的字符串。测试字符串"We will break out of prison at dawn"。

方法一: 按照下列规则对字符串进行替换加密,包括大小写字母,"_"代表空格字符。

a b c d e f g h i j k l m n o p q r s t u v w x y z _

v e k n o h z f _ i l j x d m y g b r c s w q u p t a

方法二: 就像在 JavaDoc 中看到的一样,计算机生成的是伪随机数。给 Random 对象设置一个固定的种子 Random.setSeed(long),它就能输出一个固定的随机序列出来。控制这个随机序列的范围,然后用这个序列和字符串序列进行可逆的运算,求得加密后的串。解密时通过相同的 seed 再现那个随机序列,并使用逆运算恢复原来的字符串。

提示: 字符串里的字符是以数字保存的,它们可以进行加减求余等的运算。

读者也可以使用自己的方法来替代方法二并附加简要说明。可以考虑如何将这个功能打包成一个 jar 文件给别人使用。

三、实验步骤

1. 复习理解类的继承并创建本章的工程

(1) 复习、理解类的继承,区分类的继承与组合关系。

(2) 描述出实验任务 1 要求的例子。

(3) 使用 IntelliJ IDEA 创建工程 exp3,本章的包都在 exp3 中。

说明:此后各章的实验代码都分别以"exp 章号"为工程名,创建工程的步骤在后续各章的实验步骤叙述中将略去。后续各章的实验不再使用命令行编译的方式,建议使用 IntelliJ IDEA 作为开发环境,读者也可以选择自己熟悉的集成开发环境。

2. 实现 Complex 类的 equals()方法

(1) 创建包"problem2",在"problem2"包下创建 ComplexNumber.java,实现实验任务 2 中的代码。

(2) 理解 Object 类在整个 Java 语言中的地位,以及它所提供的常用方法。

(3) 不实现 equals()方法,使用 Object 的默认实现,运行 main()方法,看所得结果与自己的预测是否一致。

(4) 按照要求给 ComplexNumber 类实现 equals()方法,并通过 main()方法的测试,返回正确结果。或者使用自己编写的 ComplexNumber 类。

注意: 浮点数在计算机中的表示是不精确的。因此,如果要判断两个浮点数是否相等,最好不要使用符号"==",而是要判断二者的差的绝对值是否很小,例如,是否小于 10^{-5},如果很小,则可以认为这两个浮点数是相等的。

3. 熟悉常用 Java 类库一

（1）打开 Java API 文档，熟悉各部分的内容，能够快速找到所需的类。

（2）查看 java.util.StringTokenizer，java.text.DateFormat，java.lang.Math 类，熟悉其方法。

（3）建立名为“problem3”的包，在包 problem3 下创建 Compute.java，实现实验任务 3 的要求。

4. 设计并实现记账软件的类

（1）按照实验任务 4 的要求，设计记账软件的类。

（2）考虑设计的合理性，合理使用抽象类、final 类等。

（3）设计类的成员变量、常用方法。

（4）新建 Java 包 MyAccount，实现自己设计的类。

记账软件的参考类图见图 3-1。

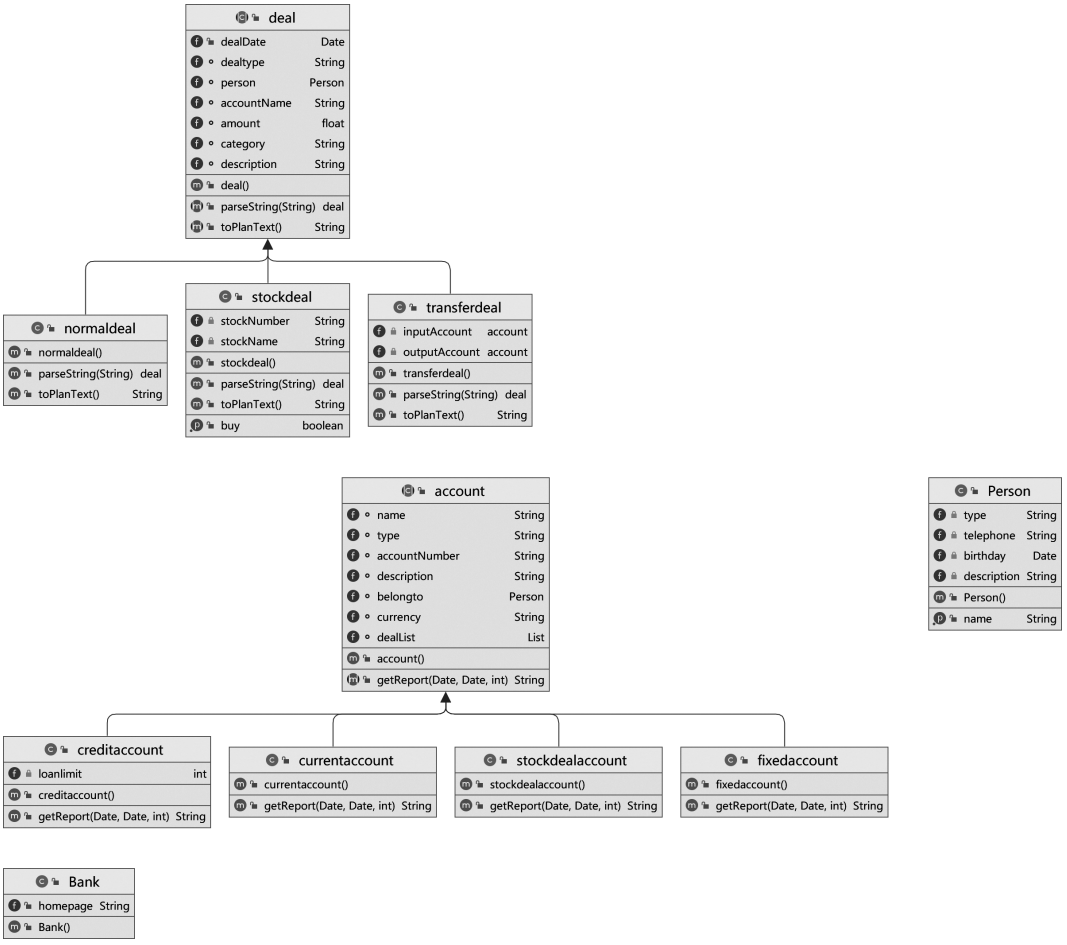

图 3-1　记账软件的参考类图

5. 熟悉常用 Java 类库二

（1）打开 Java API 文档，查看 java.lang.StringBuffer、java.util.HashMap 等类，熟悉其

方法。也可以使用其他类来协助完成实验要求。

(2) 新建名为“problem5”的包,在 problem5 包下创建 StringEncryptor.java。创建方法 encode1()、decode1()、encode2()、decode2(),实现实验任务要求。自己在 main()里写测试方法,以验证正确性。

提示: 检查加密后的字符串能否通过解密复原。

(3) 复习 jar 文件的概念。可以尝试写一个处理 String args[]参数的 main()方法。然后将这个类打包成 jar 包,供他人使用。

习题解答

1. 子类将继承超类所有的属性和方法吗? 为什么?

解:

子类不从超类继承构造方法,但可通过“super.方法名”访问;除此之外,子类从超类继承其他所有属性和方法,但 private 属性/方法不能直接访问。

2. 方法覆盖与方法重载有何不同?

解:

方法覆盖是指如果子类不需要使用从超类继承来的方法的功能,则可以声明自己的方法。在声明时,使用相同的方法名及参数表,但执行不同的功能。而重载是指名字一样但参数表不一样的方法。二者的不同主要在于:方法覆盖时,子类的参数表和父类一样,方法重载时,参数表不一样。

3. 泛型的本质是什么? 泛型可以使用在哪些场合?

解:

其本质是参数化类型,即所操作的数据类型被指定为一个参数。

泛型可以使用在类、接口以及方法中,分别称为泛型类、泛型接口和泛型方法。

4. 声明两个带有无参构造方法的类 A 和 B,声明 A 的子类 C,并且声明 B 为 C 的一个成员,不声明 C 的构造方法。编写测试代码,生成类 C 的实例对象,并观察结果。

解:

新建 Exe3_4.java 文件,其内容为:

```
class A {
    public A() {
        System.out.println("In Constructor of A");
    }
}
class B {
    public B() {
        System.out.println("In Constructor of B");
    }
}
class C extends A {
    B b;
}
```

```
public class Exe3_4 {
    public static void main(String[] args) {
        C c = new C();
    }
}
```

程序的输出结果为：

```
In Constructor of A
```

从输出结果可以看出，在通过默认构造方法创建类 C 的对象 c 时，自动调用了类 C 的超类 A 的无参数构造方法，而没有初始化 C 的成员 B。

5. 声明一个超类 A，它只有一个非默认构造方法；声明 A 的子类 B，B 具有默认方法及非默认方法，并在 B 的构造方法中调用超类 A 的构造方法。

解：

新建 Exe3_5.java 文件，其内容为：

```
class A {
    int i;
    A (int i) {
        this.i = i;
        System.out.println("In A (int i), i = " + i);
    }
}
class B extends A {
    int j;
    B () {
        super(0);
        System.out.println("In B ()");
        j = 0;
    }
    B (int j) {
        super(j);
        System.out.println("In B (int j)");
        this.j = j;
    }
}
public class Exe3_5 {
    public static void main (String[] args) {
        B b1 = new B();
        System.out.println("**********************");
        B b2 = new B(5);
    }
}
```

程序的输出结果为：

```
In A (int i), i = 0
In B ()
**********************
In A (int i), i = 5
In B (int j)
```

读者可以尝试在类 B 的无参数构造方法中不调用“super(0)”，此时会有编译错误，即找不到类 A 的无参数构造方法。

6. 声明一个类，它具有一个方法，此方法被重载三次，派生一个新类，并增加一个新的重载方法，编写测试类验证四个方法对于子类都有效。

解：

新建 Exe3_6.java 文件，其内容为：

```
class Base {
    int i;
    public void setValue(int i) {
        this.i = i;
        System.out.println("In setValue(int i), i = " + this.i);
    }
    public void setValue(float f) {
        this.i = (int)f;
        System.out.println("In setValue(float f), i = " + this.i);
    }
    public void setValue(double d) {
        this.i = (int)d;
        System.out.println("In setValue(double d), i = " + this.i);
    }
}
class Sub extends Base{
    //在派生类中重载 setValue()方法
    public void setValue(byte b) {
        this.i = (int)b;
        System.out.println("In setValue(byte b), i = " + this.i);
    }
}
public class Exe3_6 {
    public static void main(String args[]) {
        Sub s = new Sub();
        s.setValue(2);
        s.setValue(3.5f);
        s.setValue(2.8d);
        s.setValue((byte)8);
    }
}
```

程序的输出结果为：

```
In setValue(int i), i = 2
In setValue(float f), i = 3
In setValue(double d), i = 2
In setValue(byte b), i = 8
```

7. 声明一个具有 final 方法的类，声明一个子类，并试图对这个方法进行覆盖(override)，观察会有什么结果。

解：

出现编译错误，因为 final()方法不能在子类中对其进行覆盖。

8. 声明一个 final 类，并试图声明其子类，观察会有什么结果。

解：

将会出现编译错误，并提示无法从 final 类继承。

9. 什么是抽象类？抽象类中是否一定要包括抽象方法？

解：

抽象类就是不能使用 new 进行实例化的类，即没有具体实例对象的类。抽象类中不一定要包括抽象方法。

10. this 和 super 分别有哪些特殊含义？都有哪些用法？

解：

关键词 this 是当前对象的引用，关键词 super 表示超类。二者都可以用于：调用本类或超类的方法，访问本类或超类的属性，调用本类或超类的构造方法。

11. 完成下面超类及子类的声明：

(1) 声明 Student 类。

属性包括：学号、姓名、英语成绩、数学成绩、计算机成绩和总成绩。

方法包括：构造方法、getter()方法、setter()方法、toString()方法、equals()方法、compare()方法(比较两个学生的总成绩，结果可以是大于、小于、等于)，sum()方法(计算总成绩)和 testScore()方法(计算评测成绩)。

注：评测成绩可以取三门课成绩的平均分，另外，任何一门课的成绩的改变都需要对总成绩进行重新计算，因此，在每一个 setter()方法中应调用 sum()方法计算总成绩。

(2) 声明 StudentXW(学习委员)类为 Student 类的子类。

在 StudentXW 类中增加责任属性，并覆盖 testScore()方法(计算评测成绩，评测成绩＝三门课的平均分＋3)。

(3) 声明 StudentBZ(班长)类为 Student 类的子类。

在 StudentBZ 类中增加责任属性，并重写 testScore()方法(计算评测成绩，评测成绩＝三门课的平均分＋5)。

(4) 声明测试类，生成若干个 Student 类、StudentXW 类及 StudentBZ 类对象，并分别计算它们的评测成绩。

解：

新建 Exe3_11.java 文件，其内容为：

```
class Student {
    int ID;
    String name;
    int englishScore;
    int mathScore;
    int computerScore;
    int sumScore;
    //构造方法
    Student(int ID, String name, int englishScore, int mathScore,
            int computerScore) {
        this.ID = ID;
        this.name = name;
        this.englishScore = englishScore;
        this.mathScore = mathScore;
        this.computerScore = computerScore;
        sum();
    }
    //设置属性的方法
    public void setID(int ID) {
        this.ID = ID;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setEnglishScore(int englishScore) {
        this.englishScore = englishScore;
        sum();
    }
    public void setMathScore(int mathScore) {
        this.mathScore = mathScore;
        sum();
    }
    public void setComputerScore(int computerScore) {
        this.computerScore = computerScore;
        sum();
    }
    //得到属性的方法
    public int getID() {
        return ID;
    }
    public String getName() {
        return name;
    }
    public int getEnglishScore() {
```

```
        return englishScore;
    }
    public int getMathScore() {
        return mathScore;
    }
    public int getComputerScore() {
        return computerScore;
    }
    public String toString() {
        String s = name;
        s += "(学号为" + ID + ")的成绩如下: \n";
        s += "英语: " + englishScore + "\n";
        s += "数学: " + mathScore + "\n";
        s += "计算机: " + computerScore + "\n";
        s += "总分: " + sumScore + "\n";
        return s;
    }
    @Override
    public boolean equals(Object o) {
        Student s = (Student) o;
        //当二者的总成绩相等时,返回 true
        return this.sumScore == s.sumScore;
    }
    public int compare(Student s) {
        //大于时返回 1,等于时返回 0,小于时返回-1
        if (this.sumScore > s.sumScore)
            return 1;
        else if(this.sumScore == s.sumScore)
            return 0;
        else
            return -1;
    }
    public void sum() {     //计算总分
        sumScore = englishScore + mathScore + computerScore;
    }
    public int testScore() {
        return sumScore / 3;
    }
}
class StudentXW extends Student{
    int duty;
    StudentXW(int ID, String name, int englishScore, int mathScore,
            int computerScore, int duty) {
        super(ID, name, englishScore, mathScore, computerScore);
        this.duty = duty;
```

```
    }
    @Override
    public int testScore() {
        return sumScore / 3 + 3;            //平均分加上 3
    }
}
class StudentBZ extends Student{
    int duty;
    StudentBZ(int ID, String name, int englishScore, int mathScore,
            int computerScore, int duty) {
        super(ID, name, englishScore, mathScore, computerScore);
        this.duty = duty;
    }
    @Override
    public int testScore() {
        return sumScore / 3 + 5;            //平均分加上 5
    }
}
//测试类
public class Exec3_11 {
    public static void main(String[] args) {
        Student s = new Student(20100023, "张三", 85, 92, 90);
        StudentXW xw = new StudentXW(20100015, "李四", 80, 90, 95);
        StudentBZ bz = new StudentBZ(20100005, "王五", 82, 85, 88);
        System.out.print(s);
        System.out.println("评测成绩: " + s.testScore());
        System.out.print(xw);
        System.out.println("评测成绩: " + s.testScore());
        System.out.print(bz);
        System.out.println("评测成绩: " + s.testScore());
    }
}
```

程序的输出结果为：

```
张三(学号为 20100023)的成绩如下:
英语: 85
数学: 92
计算机: 90
总分: 267
评测成绩: 89
李四(学号为 20100015)的成绩如下:
英语: 80
数学: 90
计算机: 95
```

```
总分：265
评测成绩：89
王五(学号为 20100005)的成绩如下：
英语：82
数学：85
计算机：88
总分：255
评测成绩：89
```

12. 包有什么作用？如何声明包和引用包中的类？

解：

包可以管理命名空间，可以解决同名类的冲突问题；也可以组织相关的类，并控制类的访问权限。

使用 package 关键字声明包。使用 import 关键字引用包中的类。