第一部分
JAVA 面向对象编程基础

1 实验 1：熟悉 JAVA 程序的开发步骤

1.1 实验目的

本实验的目的是掌握开发 Java 应用程序的三个步骤：编写源文件、编译源文件和运行应用程序。

1.2 实验要求

编写一个简单的 Java 应用程序。该程序在命令行窗口输出两行文字："你好，很高兴学习 Java 语言"和"We are students"。

1.3 程序模版

请按模版要求，将【代码】字段替换为 Java 程序代码。

Hello.java

```
public class Hello{
    public static void main (String args[ ]){
        【代码 1】          //命令行窗口输出"你好，很高兴学习 Java语言"
        Student zhang = new Student();
        zhang.speak();
    }
}
class Student{
    void speak(){
        【代码 2】          //命令行窗口输"We are students"
    }
}
```

1.4 实验指导

（1）打开一个文本编辑器

如果是 Windows 操作系统，打开 “记事本” 编辑器。可以通过 “开始” - “程序” - “附件” - “记事本” 来打开文本编辑器。如果是其他操作系统，请在指导老师的帮助下打开一个纯文本编辑器。按照 “程序模板” 的要求编辑输入源程序，在命令行输出"你好，很高兴学习 Java 语言"的代码是：System.out.println("你好，很高兴学习 Java 语言")或 System.out.print("你好，很高兴学习 Java 语言")。二者的区别是，前者在输出字符序列"你好，很高兴学习 Java 语言"后，将输出光标移动到下一行，后者不移动输出光标到下一行。

（2）保存源文件并命名为 Hello.java。

要求将源文件保存到 C:盘的某个文件夹中。例如 C:\1000。

（3）编译源文件

打开命令行窗口，对于 Windows 操作系统，打开 MS-DOS 窗口。可以通过单击“开始”-“程序”-“附件”-“MS-DOS”打开命令行窗口，也可以单击“开始”按钮，选择“运行”命令。在弹出的对话框的输人命令栏中输人 cmd 打开命令行窗口。如果当前 MS-DOS 窗口显示的逻辑符是“D:\”，请输人“C:\”回车确认，使得当前 MS-DOS 窗口的状态是“C:\”。如果目前 MS-DOS 窗口的状态是 C 盘符的某个子目录，请输人“cd\”，使得当前 MS-DOS 窗口的状态是“cd\”，MS-DOS 窗口的状态是“C:\”时，输入进入文件夹目录的命令，例如“CD 1000”。然后执行下列编译命令。

```
C:\1000> javac Hello.java
```

初学者在这一步可能会遇到下列错误提示。

- Command not Found 出现该错误的原因是没有设置好系统变量 path。
- File not Found 出现该错误的原因是没有将源文件保存在当前目录中，例如 C:\1000，或源文件的名字不符合有关规定，例如，错误地将源文件命名为“hello.java”或“Hello.java.txt”，要特别注意：Java 语言的标识符号是区分大小写的。
- 出现一些语法错误提示，例如，在汉语输人状态下输入了程序中需要的语句分号等，Java 源程序中语句所涉及的小括号及标点符号都是英文状态下输入的，比如“你好，很学习 Java 语言”中的引号必须是英文状态下的引号，而字符串里面的符号不受汉字或英文的限制。

（4）运行程序

```
C:\1000> java Hello
```

这一步可能会遇到错误提示：Exception in thread "main" java.lang.NoClassFoundError。出现该错误的原因是没有设置好系统变量 classpath 或运行的不是主类的名字或程序没有主类。JDK 安装目录的\jre 文件夹中包含着 Java 应用程序运行时所需要的 Java 类库，这些类库被包含在\jre\lib 中的压缩文件 rt.jar 中。安装 JDK 一般不需要设置环境变量 classpath，如果主类正确，但程序仍然遇到错误提示，例如：Exception in thread "main" java.lang.NoClassFoundError，那么可以重新编辑系统环境变量 classpath 的值。对于 Windows 7/ XP，右击“计算机”选择“我的电脑”，在弹出的菜单中选择“属性”-“系统特性”-“高级系统设置”-“高级”选项，然后单击“环境变量”按钮，添加如图 1.10 所示的系统环境变量，如果曾经设置过环境变量 classpath，可单击该变行编辑操作，将需要的值加入即可。

注：环境变量 classpath 可以加载应用程序当前目录及其子目录中的类。rt.jar 包含了 Java 运行环境提供的类库中的类。

1.5 扩展练习

- 编译器怎样提示丢失大括号的错误？
- 编译器怎样提示语句丢失分号的错误？
- 编译器怎样提示将 System 写成 system 这一错误？
- 编译器怎样提示将 String 写成 string 这一错误？

2 实验 2：熟悉 JAVA 程序的基本结构

2.1 实验目的

熟悉 Java 应用程序的基本结构，并能联合编译应用程序所需要的类。

2.2 实验要求

编写 3 个源文件：MainClass.java、A.java 和 B.java，每个源文件只有一个类。MainClass.java 含有应用程序的主类（含有 main 方法），并使用了 A 和 B。将 3 个源文件保存到同一 0 录中，例如 C:\1000,然后编译 MainClass.iava。程序运行参考效果如图 2.8 所示 A

2.3 程序模板

请按模板要求，将【代码】替换为 Java 程序代码。

MainClass.java

```
public class MainClass {
    public static void main (String args[ ]) {
        【代码 1】          //命令行窗口输出"你好，只需编译我"
        A a = new A();
        a.fA();
        B b = new B();
        b.fB();
    }
}
```

A.java

```
public class A {
    void fA() {
        【代码 2】          //命令行窗口输出"I am A"
    }
}
```

B. java

```
public class B {
    void fB() {
        【代码 3】          //命令行窗口输出"I am B"
    }
}
```

2.4 实验指导

编译 MainClass.java 的过程中，Java 系统会自动地先编译 A.java、B.java。编译通过后，C:\1000 中将会有 MainClass.class、A.class 和 B.class 3 个字节码文件。当运行上述 Java 应用程序时，虚拟机将 MainClass.class 和 A.class、B.class 加载到了内存。当虚拟机将 MainClass.class 加载到内存时，就为主类中的 main 方法分配了入口地址，以便 Java 解释器调用 main 方法开始运行程序。如果编写程序时错误地将主类中的 main 方法写成：public void main(String args[ ])，那么，程序可以编译通过，但无法运行。

2.5 扩展练习

2.5.1 思考

将 MainClass.class 编译通过以后，不断地修改 A.java 源文件中的代码，比如，在命令行窗口输出"Nice to meet you!"或"Can you help me?"。要求每次修改 A.java 源文件以后，单独编译 A.java，然后直接运行应用程序 MainClass。

2.5.2 对象赋值实验

按照以下要求编写 Java 应用程序：定义一个类 Point，用来描述平面直角坐标系中点的坐标，该类应该能描述点的横、纵坐标信息及一些相关操作，包括获取点的横、纵坐标，修改点的坐标，显示点的当前位置等；定义一个主类 PointTest，创建 Point 类的对象并对其进行有关的操作。理解 Java 对象的理解 “值传递” 的过程。

PointTest.java

```
class Point{
    double x,y;
    public void setXY(double a, double b){
        x=a;
        y=b;
    }
    public double getX(){
        return x;
    }
    public double getY(){
        return y;
    }
    public void disp(){
        System.out.println("点的当前坐标为：("+x+", "+y+")");
    }
}
public class PointTest{
    public static void main(String[] args){
        Point p1=new Point();
        p1.disp();
        p1.setXY(3.2,5.6);
        p1.disp();
    }
}
```

运行结果：

- 点的当前坐标为：(0.0, 0.0)
- 点的当前坐标为：(3.2, 5.6)

2.5.3 JAVADOC 文档化工具的使用

Java 2 SDK 1.4.1 中提供了一个文档自动生成工具，可以简化程序员编写文档的工作。可以使用 javadoc.exe 命令启动 Java 文档化工具，自动生成 Java 程序文档。

输入下面给出的 Java 应用程序，利用 javadoc 命令生成该 Java 应用程序的文档，并使用浏览器 IE 显示生成的文档页面内容。

CommentToDoc.java

```
/* Java语言实验
   课程：Java语言
   时间：2019-2020学年第一学期
*/
/**
  这是一个 Java语言实验程序，定义类 CommentToDoc。其中含有 main()方法，故可以作为一
  个应用程序单独运行。其功能是在默认的输出设备上输出字符串"我在学习 Java语言"。
*/
public class CommentToDoc {
    //主方法，作为 Java应用程序的默认入口。
    public static void main(String args[ ]) {
        System.out.println("我在学习 Java语言"); //输出"我在学习 Java语言"
    }
}
```

执行指令：

```
javadoc CommentToDoc.java
```

生成文件：

- CommentTest.html
- package-frame.html
- package-summary.html
- package-tree.html
- constant-values.html
- overview-tree.html
- index-all.html
- deprecated-list.html
- allclasses-frame.html
- allclasses-noframe.html
- index.html
- help-doc.html

3 实验 3：简单型变量输入输出

3.1 实验目的

掌握从键盘为简单型变量输入数据。掌握使用 Scanner 类创建一个对象，例如：

```
Scanner reader = new Scanner (System.in);
```

练习让 reader 对象调用下列方法读取用户在命令行（例如，MS-DOS 窗口）输入的各种简单类型数据：

- nextBoolecm()
- nextByte()
- nextShort()

- nextInt()
- nextLong()
- nextFloat()
- nextDouble()

在调试程序时，体会上述方法都会堵塞，即程序等待用户在命令行输人数据回车确认。

3.2 实验要求

编写一个 Java 应用程序，在主类的 main 方法中声明用于存放产品数量的 int 型变量 amount 和产品单价的 float 型变量 price，以及存放全部产品总价值 float 型变量 sum。

使用 Scanner 对象调用方法让用户从键盘为 amount，price 变量输入数值，然后程序计算出全部产品总价值，并输出 amount，price 和 sum 的值。

3.3 程序模版

请按模板要求，将【代码】替换为 Java 程序代码。

InputData.java

```
import java.util.Scanner;
public class InputData {
   public static void main(String args[]) {
      Scanner reader=new Scanner(System.in);
      int amount=0;
      float price=0,sum=0;
      System.out.println("输入产品数量(回车确认):");
      【代码 1】     //从键盘为 amount赋值
      System.out.println("输入产品单价(回车确认):");
      【代码 2】     //从键盘为 price赋值
      sum = price*amount;
      System.out.printf("数量:%d,单价:%5.2f,总价值:%5.2f",amount,price,sum);
   }
}
```

3.4 实验指导

由于 amount 是 int 型，因此【代码 1】应该是 amount = reader.nextInt()，而 price 是 float 型，因此代码 2 应该是 price = reader.nextFloat()，不可以是 price = reader.nextDouble()。

另外，Scanner 对象可以调用 hasNextXXX()方法判断用户输入的数据的类型，例如，如果用户在键盘输入带小数点的数字：12.34（回车），那么 reader 对象调用 hasNextDouble()返回的值是 true，而调用 hasNextByte()、hasNextInt()以及 hasNextLong()返回的值都是 false；如果用户在键盘输人一个 byte 取值范围内的整数：89（回车），那么 reader 对象调用 hasNextByte()、hasNextInt()、hasNextLong( )以及 hasNextDoublc()返回的值都是 true。nextLine()等待用户在命令行输人一行文本回车，该方法得到一个 String 类型的数据，String 类型将在后续课程中讲述。

在从键盘输人数据时，我们经常让 reader 对象先调用 hasNextXXX()方法等待用户在键盘输入数据，然后再调用 nextXXX()方法读取数据。

3.5 扩展练习

3.5.1 反复输入实验

上机调试下列程序。该程序可以让用户在键盘依次输人若干个数字，每输入一个数字都需要按回车键确认，最后用户在键盘输入一个非数字字符串结束整个输入操作过程（用户输入非数字数字后 reader.hasNextDouble()的值将是 false）。程序将计算出这些数的和以及平均值。

MultipleInput.java

```
import java.util.*;
public class MultipleInput{
    public static void main (String args[ ]){
        Scanner reader = new Scanner(System.in);
        double sum = 0;
        int m = 0;
        while(reader.hasNextDouble()){
            double x = reader.nextDouble();
            m = m + 1;
            sum = sum + x;
        }
        System.out.printf("%d个数的和为%f\n",m,sum);
        System.out.printf("%d个数的平均值是%f\n",m,sum/m);
    }
}
```

3.5.2 转义字符实验

输入、运行以下 Java 应用程序，写出运行结果，以验证 Java 语言转义字符的功能。

TransChar.java

```
public class TransChar{
    public static void main(String args[ ]) {
        char ch1 = '\b';
        char ch2 = '\t';
        char ch3 = '\n';
        char ch4 = '\r';
        char ch5 = '\"';
        char ch6 = '\'';
        char ch7 = '\\';
        System.out.println("广州大学"+ch1+"计算机学院");
        System.out.println("广州大学"+ch2+"计算机学院");
        System.out.println("广州大学"+ch3+"计算机学院");
        System.out.println("广州大学"+ch4+ch3+"计算机学院");
        System.out.println(ch5+"广州大学"+"计算机学院"+ch5);
        System.out.println(ch6+"广州大学"+"计算机学院"+ch6);
        System.out.println(ch7+"广州大学"+"计算机学院"+ch7);
    }
}
```

运行结果：

- 广州大 计算机学院
- 广州大学 计算机学院
- 广州大学
- 计算机学院
- 广州大学

- 计算机学院
- "广州大学计算机学院"
- '广州大学计算机学院'
- \广州大学计算机学院\

3.5.3 输入数据处理实验

输入、运行以下 Java 应用程序，写出运行结果，并说明程序所完成的功能。

ParseDouble.java

```
import java.io.*;
public class ParseDouble {
  public static void main(String args[]) {
    String s;
    double d;
    int i;
    boolean b = false;
    do {
      try {
        System.out.println("请输入一个浮点数: ");
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        s = br.readLine();          // 以字符串方式读入
        i = s.indexOf('.');         // 找到小数点的位置
        d = Double.parseDouble(s);  // 将字符串转换成浮点数
        System.out.println(d + " 整数部分为: " + (long)d);
        if(i == -1)                 // 若没有小数点，则没有小数部分
          System.out.println(d + " 小数部分为: 0.0");
        else                        // 若有小数点，则截取小数点后的字符串合成浮点数
          System.out.println(d +
                     " 小数部分为: " +
                     Double.parseDouble(((s.charAt(0)=='-') ? "-" : "") +
                     "0." +
                     s.substring(i+1,s.length())));
        b = false;
      }
      catch(NumberFormatException nfe) {
        System.out.println("输入浮点数格式有误。\n");
        b = true;
      }
      catch(IOException ioe) {
        b = false;
      }
    }
    while(b);   // 浮点格式错误时重新输入
  }             // end of main
}               // end of class
```

运行结果：

- 请输入一个浮点数：
- abc
- 输入浮点数格式有误
- 请输入一个浮点数：
- 3.1415926
- 3.1415926 的整数部分为：3
- 3.1415926 的小数部分为：0.1415926

4 独立实验任务 1-1（需要提交实验报告）

4.1 实验目的

- 掌握开发 Java 应用程序的步骤，掌握 Java 应用程序的基本结构。
- 掌握 Java 基本数据类型在命令行的输入和输出方法。
- 熟悉如何使用 Java 分支和循环语句解决问题。

4.2 实验任务

编写一个 Java 应用程序，在主类的 main 方法中实现下列功能。

- 程序随机分配给用户一个 1 至 100 之间的整数
- 用户通过键盘输人自己的猜测
- 程序返回提示信息，提示信息分别是：“猜大了”、“猜小了”和“猜对了”
- 用户可根据提示信息再次输入猜测，直到提示信息是“猜对了”
- 用户猜对以后，显示猜测次数，并提供“重新开始”和“退出”功能

注：这是第一部分实验需要大家独立完成，并写入实验报告的两个独立实验任务之一。

5 实验 4：分支和循环语句

5.1 实验目的

本实验的目的是熟悉如何使用循环语句解决问题。

5.2 实验要求

编写一个 Java 应用程序，输出区间[200，300]上的所有素数，要求写出程序的运行结果。

5.3 程序模版

请按模板要求，将【代码】替换为 Java 程序代码。

FindPrime.java

```
public class FindPrime{
    public static void main(String[] args){
        int i,j;
        boolean isPrime;
        for(i=200;i<=300;i++){
            isPrime=true;
            for(j=2;j<i-1;j++){
                if(i%j==0){
                    【代码】
                }
            }
            if(isPrime){
                System.out.print(i+" ");
            }
        }
        System.out.println();
    }
```

```
}
```

运行结果：

- 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293

5.4 实验指导

使用 break 语句可以跳出当前循环。

5.5 扩展练习

5.5.1 循环标签

continue 语句默认是结束当前循环的单次循环。有时我们在使用循环时，想通过内层循环里的语句直接结束外部循环的当前单次循环。Java 提供了相关的功能，只需要在 continue 后通过标签指定外层循环。Java 中的标签是一个紧跟着英文冒号的标识符，该标签只有放在循环语句之前才有作用。类似的，break 语句默认是结束当前循环，在使用循环时，也可以通过内层循环里中的 break 后指定外层循环标签，从而直接跳出外层循环。

FindPrime2.java

```
public class FindPrime2{
    public static void main(String[] args){
        int i,j;
        outer:
        for(i=200;i<=300;i++){
            for(j=2;j<i-1;j++){
                if(i%j==0)
                  continue outer;
            }
            System.out.print(i+" ");
        }
    }
}
```

6 实验 5：类与对象

6.1 实验目的

- 掌握 Java 类的定义和使用，以及对象的声明和使用。
- 理解构造函数的概念和使用方法。
- 掌握类及其成员的访问控制符的使用。

6.2 实验要求

按要求编写一个 Java 应用程序：第一，定义一个表示学生的类 Student。这个类的属性有“学号”、“班号”、“姓名”、“性别”、“年龄”，方法有“获得学号”、“获得班号”、“获得性别”、“获得姓名”、“获得年龄”、“获得年龄”；第二，为类 Student 增加一个方法 public String toString( )，该方法把 Student 类的对象的所有属性信息组合成一个字符串以便输出显示。

6.3 程序模版

请按模版要求，将【代码】替换为 Java 程序代码。

StudentDemo.java

```
class Student{
    private long studentID;
    private int classID;
    private String name;
    private String sex;
    private int age;
    public Student(long studentID, int classID, String name,
                   String sex, int age){
        【代码】
    }
    public long getStudentID(){
        return studentID;
    }
    public int getClassID(){
        return classID;
    }
    public String getName(){
        return name;
    }
    public String getSex(){
        return sex;
    }
    public int getAge(){
        return age;
    }
    public String toString(){
        return "学号："+getStudentID()+
               "\n班号："+getClassID()+
               "\n姓名："+getName()+
               "\n性别："+getSex()+
               "\n年龄："+getAge();
    }
}
public class StudentDemo{
    public static void main(String[] args){
        Student s1=new Student(90221,2,"Tom","male",20);
        System.out.println(s1.toString());
    }
}
```

运行结果是：

学号：90221
班号：2
姓名：Tom
性别：male
年龄：20

6.4 实验指导

在 Student 类的构造方法中，参数中的局部变量的名字与成员变量的名字相同，于是 Student 类的成员变量被隐藏，此时若想使用被隐藏的成员变量，必须使用关键字 this。

6.5 扩展练习

6.5.1 思考

对于上述实验中的 Student 类，尝试能否使用以下语句创建一个新的对象。如果不能，为什么？

```
Student s1=new Student();
```

6.5.2 拓展 STUDENT 类

拓展上述实验中的 Student 类，添加以下方法：“设置学号”、“设置班号”、“设置性别”、“设置姓名”、“设置年龄”、“设置年龄”，以及没有参数的构造方法。

StudentExtendedDemo.java

```
class StudentExtended{
    private long studentID;
    private int classID;
    private String name;
    private String sex;
    private int age;
    public Student(long studentID, int classID, String name,
                   String sex, int age){
        【代码 1】
    }
    public long getStudentID(){
        return studentID;
    }
    public int getClassID(){
        return classID;
    }
    public String getName(){
        return name;
    }
    public String getSex(){
        return sex;
    }
    public int getAge(){
        return age;
    }
    public void setStudentID(long studentID){
        【代码 2】
    }
    public void setClassID(int classID){
        【代码 3】
    }
    public void setName(String name){
        【代码 4】
    }
    public void setSex(String sex){
        【代码 5】
    }
    public void setAge(int age){
        【代码 6】
    }
    public Student(){
    }
    public String toString(){
        return "学号: "+getStudentID()+
```

```
                "\n班号："+getClassID()+
                "\n姓名："+getName()+
                "\n性别："+getSex()+
                "\n年龄："+getAge();
    }
}
public class StudentExtendedDemo{
    public static void main(String[] args){
        Student s1=new Student(90221,2,"Tom","male",20);
        Student s2=new Student();
        【代码 7】
        System.out.println(s1.toString());
        System.out.println(s2.toString());
    }
}
```

要求运行结果是：

学号：90221
班号：2
姓名：Tom
性别：male
年龄：20
学号：90222
班号：1
姓名：Jerry
性别：female
年龄：21

7 实验 6：类变量与实例变量

7.1 实验目的

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。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，方法中出现的类变量也是该对象的变量，只不过这个变量和所有的其他对象共享而已。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。本实验的目的是掌握类变量与实例变量，以及类方法与实例方法的区别。

7.2 实验要求

编写程序模拟两个村庄共同拥有一片森林。编写一个 Village 类，该类有一个静态的 int 型成员变量 treeAmount 用于模拟森林中树木的数量。在主类 MainClass 的 main 方法中创建两个村庄，一个村庄改变了 treeAmount 的值，另一个村庄查看 treeAmount 的值。

7.3 程序模版

请按模版要求，将【代码】替换为 Java 程序代码。

Village.java

```
class Village {
    static int treeAmount;  //模拟森林中树木的数量
    int peopleNumber;       //村庄的人数
    String name;            //村庄的名字
    Village(String s) {
        name = s;
    }
    void treePlanting(int n){
        treeAmount = treeAmount+n;
        System.out.println(name+"植树"+n+"颗");
    }
    void  fellTree(int n){
        if(treeAmount-n>=0){
           treeAmount = treeAmount-n;
           System.out.println(name+"伐树"+n+"颗");
        }
        else {
           System.out.println("无树木可伐");
        }
    }
    static int lookTreeAmount() {
       return treeAmount;
    }
    void addPeopleNumber(int n) {
       peopleNumber = peopleNumber+n;
       System.out.println(name+"增加了"+n+"人");
   }
}
```

MainClass.java

```
public class MainClass {
    public static void main(String args[]) {
        Village zhaoZhuang,maJiaHeZhi;
        zhaoZhuang = new Village("赵庄");
        maJiaHeZhi = new Village("马家河子");
        zhaoZhuang.peopleNumber=100;
        maJiaHeZhi.peopleNumber=150;
        【代码 1】 //用类名 Village访问 treeAmount,并赋值 200
        int leftTree =Village.treeAmount;
        System.out.println("森林中有 "+leftTree+" 颗树");
        【代码 2】 //zhaoZhuang调用 treePlanting(int n),并向参数传值 50
        leftTree =【代码 3】 //maJiaHeZhi调用 lookTreeAmount()方法得到树木的数量
        System.out.println("森林中有 "+leftTree+" 颗树");
        【代码 4】 //maJiaHeZhi调用 fellTree(int n),并向参数传值 70
        leftTree = Village.lookTreeAmount();
        System.out.println("森林中有 "+leftTree+" 颗树");
        System.out.println("赵庄的人口:"+zhaoZhuang.peopleNumber);
        zhaoZhuang.addPeopleNumber(12);
        System.out.println("赵庄的人口:"+zhaoZhuang.peopleNumber);
        System.out.println("马家河子的人口:"+maJiaHeZhi.peopleNumber);
        maJiaHeZhi.addPeopleNumber(10);
        System.out.println("马家河子的人口:"+maJiaHeZhi.peopleNumber);
    }
}
```

7.4 实验指导

对象共享类变量，在代码 1 之前已经有了 zhaoZhuang 对象，这个时候，代码 1 用 Village.treeAmount=200 或 zhaoZhuang.treeAmount=200 替换都是正确的。

7.5 扩展练习

代码 2 是否可以写成 “Village.treePlanting(50);” ？

8 实验 7：子类与继承

8.1 实验目的

上转型对象可以访问子类继承或隐藏的成员变量，也可以调用子类继承的方法或子类重写的实例方法。上转型对象操作子类继承的方法或子类重写的实例方法，其作用等价于子类对象去调用这些方法。因此，如果子类重写了父类的某个实例方法后，当对象的上转型对象调用这个实例方法时一定是调用子类重写的实例方法。本实验的目的是掌握上转型对象的使用，理解不同对象的上转型对象调用同一方法可能产生不同的行为，即理解上转型对象在调用方法时可能具有多种形态。

8.2 实验要求

1、编写一个 abstract 类，类名为 Geometry，该类有一个 abstract 方法。

2、编写 Geometry 的若干个子类，比如 Circle 子类和 Rect 子类。

3、编写 Student 类，该类定义一个 public double area(Geometry...p)方法，该方法的参数是可变参数，即参数的个数不确定，但类型都是 Geometry。该方法返回参数计算的面积之和。

4、在主类 MainClass 的 main 方法中创建一个 Student 对象，让该对象调用 public double area(Geometry...p)计算若干个矩形和若干个圆的面积之和。

8.3 程序模版

请按模板要求，将【代码】替换为 Java 程序代码。

Geometry.java

```
public abstract class Geometry  {
    public abstract double getArea();
}
```

Rect.java

```
public class Rect extends Geometry {
    double a, b;
    Rect(double a, double b) {
        this.a = a;
        this.b = b;
    }
   【代码 1】 //重写 getArea()方法,返回矩形面积
}
```

Circle.java

```
public class Circle extends Geometry {
    double r;
```

```
    Circle(double r) {
        this.r = r;
    }
    【代码 2】 //重写 getArea()方法,返回圆面积
}
```

Student.java

```
public class Student {
    public double area(Geometry ...p) {
       double sum=0;
       for(int i=0; i<p.length; i++) {
           sum = sum + p[i].getArea();
       }
       return sum;
    }
}
```

MainClass.java

```
public class MainClass {
    public static void main(String args[]) {
        Student zhang = new Student();
        double area =
        zhang.area(new Rect(2,3), new Circle(5.2), new Circle(12));
        System.out.printf("2个圆和 1个矩形图形的面积和: \n%10.3f",area);
    }
}
```

8.4 实验指导

尽管 abstract 类不能创建对象，但 abstract 类声明的对象可以存放子类对象的饮用，即成为子类对象的上转型对象。由于 abstract 类可以有 abstract 方法，这样就保证子类必须要重写这些 abstract 方法。由于 Student 类的 area 方法的参数类型是 Geometry，因此可以将其子类的对象的引用传递给方法 area 的参数。因此 area 方法可以通过循环语句让每个参数调用 getArea 方法，计算各自的面积。

可变参数是 JDK1.5 版本后新增的功能。可变参数声明方法时不给出参数列表中从某项直至最后一项参数的名字和个数，但这些参数的类型必须相同。可变参数使用"..."表示若干个参数，这些参数的类型必须相同，最后一个参数必须是参数列表中的最后一个参数。例如：

```
public void f(int ... x)
```

那么，方法 f 的参数列表中，从第一个至最后一个参数都是 int 型，但连续出现的 int 型参数的个数不确定。称 x 是方法 f 的参数列表中的可变参数的 “参数代表” 。参数代表可以痛殴下标运算来表示参数列表中的具体参数，即 x[0], x[1], ..., x[m]分别表示 x 代表的第 1 个至 m+1 个参数。

对于 Student 类中的 area(Geometry...p)方法中的可变参数 p，那么 p[i]就是第 i+1 个参数。

8.5 扩展练习

8.5.1 编写 TRIANGLE 类

编写一个 Geometry 的子类 Triangle，可以计算三角形的面积，在主类中让 Student 类的对象 zhang 调用 area 方法计算 1 个三角形和 2 个圆的面积之和。

8.5.2 分析类的关系

阅读如下所示的 3 个 Java 类的定义，分析它们之间的关系，写出运行结果

```
class SuperClass {
    int x;
    SuperClass() {
        x=3;
        System.out.println("in SuperClass : x=" +x);
    }
    void doSomething() {
        System.out.println("in SuperClass.doSomething()");
    }
}
class SubClass extends SuperClass {
    int x;
    SubClass() {
       super();      //调用父类的构造方法
        x=5;         //super( ) 要放在方法中的第一句
        System.out.println("in SubClass :x="+x);
    }
    void doSomething( ) {
        super.doSomething( ); //调用父类的方法
    System.out.println("in SubClass.doSomething()");
    System.out.println("super.x="+super.x+" sub.x="+x);
}
}
public class Inheritance {
    public static void main(String args[]) {
        SubClass subC=new SubClass();
        subC.doSomething();
    }
}
```

运行结果：

in SuperClass: x=3

in SubClass: x=5

in SuperClass.doSomething()

in SubClass.doSomething()

super.x=3 sub.x=5

8.5.3 类的设计

请按以下要求，完成类的设计，并编写 Java 应用程序：假定根据学生的 3 门学位课程的分数决定其是否可以拿到学位。对于本科生，如果 3 门课程的平均分数超过 60 分即表示通过；而对于研究生，则需要平均超过 80 分才能够通过。

- 设计一个基类 Student 描述学生的共同特征。
- 设计一个描述本科生的类 Undergraduate，该类继承并扩展 Student 类。
- 设计一个描述研究生的类 Graduate，该类继承并扩展 Student 类。
- 设计一个测试类 StudentDemo，分别创建本科生和研究生这两个类的对象，并输出相关信息。

StudentDemo2.java

```
class Student{
    private String name;
```

```
    private int classA,classB,classC;
    public Student(String name, int classA, int classB, int classC){
        this.name=name;
        this.classA=classA;
        this.classB=classB;
        this.classC=classC;
    }
    public String getName(){
        return name;
    }
    public int getAverage(){
        return (classA+classB+classC)/3;
    }
}
class UnderGraduate extends Student{
    public UnderGraduate(String name,int classA,int classB,int classC){
        super(name,classA,classB,classC);
    }
    public void isPass(){
        if(getAverage()>=60)
            System.out.println("本科生"+getName()+
                               "的三科平均分为："+getAverage()+
                               ",可以拿到学士学位。");
        else
            System.out.println("本科生"+getName()+
                               "的三科平均分为："+getAverage()+
                               ",不能拿到学士学位。");
    }
}
class Graduate extends Student{
    public Graduate(String name,int classA,int classB,int classC){
        super(name,classA,classB,classC);
    }
    public void isPass(){
        if(getAverage()>=80)
            System.out.println("研究生"+getName()+
                               "的三科平均分为："+getAverage()+
                               ",可以拿到硕士学位。");
        else
            System.out.println("研究生"+getName()+
                               "的三科平均分为："+getAverage()+
                               ",不能拿到硕士学位。");
    }
}
public class StudentDemo2{
    public static void main(String[] args){
        UnderGraduate s1=new UnderGraduate("Tom",55,75,81);
        Graduate s2=new Graduate("Mary",72,81,68);
        s1.isPass();
        s2.isPass();
    }
}
```

运行结果：

本科生 Tom 的三科平均分为：70，可以拿到学士学位。

研究生 Mary 的三科平均分为：73，不能拿到硕士学位。

9 实验 8：文件读写

9.1 实验目的

- 掌握 Java I/O 基本原理。
- 掌握 InputStream、OutputStream 抽象类的基本使用方法。
- 掌握 FileInputStream、FileOutputStream 抽象类的基本使用方法。

9.2 实验要求

输入、运行以下 Java 应用程序，写出运行结果，理解读取文件的基本方法。

9.3 程序模版

ReadFile.java

```
import java.io.*;
public class ReadFile{
    public static void main(String[] args) throws IOException{
        BufferedReader br=new BufferedReader(new FileReader(
                          "ReadFile.java"));
        String s=br.readLine();
        while(s!=null){
            System.out.println(s);
            s=br.readLine();
        }
        br.close();
    }
}
```

输入、运行以下 Java 应用程序，写出运行结果，理解读写文件的基本方法。

CopyFile.java

```
import java.io.*;
public class CopyFile {
    public static void main(String[] args) {
        try {
            FileInputStream fis = new FileInputStream("CopyFile.java");
            FileOutputStream fos = new FileOutputStream("temp.txt");
            int read = fis.read();
            while ( read != -1 ) {
                fos.write(read);
            read = fis.read();
            }
            fis.close();
            fos.close();
        }
        catch (IOException e) {
            System.out.println(e);
        }
    }
}
```

9.4 实验指导

CopyFile 类的功能是完成文件的复制：通过字节流从 “CopyFile.java” 文件中读取数据并写入到 “temp.txt” 文件中去，实现文件复制功能。

9.5 扩展练习

编写一个 Java 应用程序，实现如下的设计功能：运行该程序可以列出当前目录下的文件。

```
import java.io.*;
public class FileList2{
    public static void main(String[] args){
        File dir=new File(".");
        File files[]=dir.listFiles();
        System.out.println(dir);
        for(int i=0;i<files.length;i++){
            if(files[i].isFile())
                System.out.println("\t"+files[i].getName());
            else
                System.out.println("<DIR>\t"+files[i].getName());
        }
    }
}
```

10 独立实验任务 1-2（需要提交实验报告）

10.1 实验目的

熟悉类的基本设计方法，根据 Java 类的继承机制有效解决问题。

10.2 实验任务

假定要为某个公司编写雇员工资支付程序，这个公司有各种类型的雇员（Employee），不同类型的雇员按不同的方式支付工资：

- 经理（Manager）：每月获得一份固定的工资
- 销售人员（Salesman）：在基本工资的基础上每月还有销售提成。
- 工人（Worker）：按照每月工作的天数计算工资。

根据上述要求试用类的继承和相关机制描述这些功能，并编写一个 Java 应用程序，演示这些类的用法。

提示：应设计一个雇员类（Employee）描述所有雇员的共图特性，这个类应该提供一个计算工资的抽象方法 ComputeSalary( )，使得可以通过这个类计算所有雇员的工资。经理、销售人员和一般工人对应的类都应该继承这个类，并重新定义计算工资的方法，进而给出它的具体实现。

11 实验 9：面向接口编程

11.1 实验目的

接口回调是多态的另一种体现，接口回调是指可以把使用某一接口的类创建的对象的引用赋值给该接口声明的接口变量中，那么该接口变量就可以调用被类实现的接口中的方法，当接口变量调用被类实现的接口中的方法时，就是通知相应的对象调用接口的方法，这一过程成为对象功能的接口回调。所谓面向接口编程，是指当设计某种重要的类时，不让该类面向具体的类，而是面向接口，即设计类中的重要数据是接口声明的变量，而不是具体类声明的对象。本实验的目的是掌握接口回调和面向接口编程思想。

11.2 实验要求

小狗在不同环境条件下可能呈现不同的状态，小狗通过调用 cry()方法体现自己的当前的状态。要求用接口封装小狗的状态。具体要求如下。

- 编写一个接口 DogState，该接口有一个名字为 void showState()的方法。
- 编写 Dog 类，该类中有一个 DogState 接口声明的变量 state。另外，该类有一个 cry()方法，在该方法中让接口 state 回调 showState()方法。即 Dog 对象通过 cry()方法来体现自己目前的状态。
- 编写若干个实现 DogState 接口的类，负责刻画小狗的各种状态。
- 编写主类，在主类中用 Dog 创建小狗，并让小狗调用 cry 方法体现自己的状态。

11.3 程序模板

请按模板要求，将【代码】替换为 Java 程序代码。

E.java

```
interface DogState {
  public void showState();
}
class SoftlyState implements DogState {
  【代码 1】 //重写 public void showState()
}
class MeetEnemyState implements DogState {
  【代码 2】 //重写 public void showState()
}
class MeetFriendState implements DogState {
  【代码 3】 //重写 public void showState()
}
class Dog {
  DogState  state;
  public void cry() {
    state.showState();
  }
  public void setState(DogState s) {
    state = s;
  }
}
public class E {
  public static void main(String args[]) {
    Dog yellowDog =new Dog();
    yellowDog.setState(new SoftlyState());
```

```
    yellowDog.cry();
    yellowDog.setState(new MeetEnemyState());
    yellowDog.cry();
    yellowDog.setState(new MeetFriendState());
    yellowDog.cry();
  }
}
```

11.4 实验指导

接口 DogState 规定了状态的方法名称，因此，实现该接口的类，例如 MeetEnemyState 类，必须具体实现接口中的方法 public void showState()，以便体现小狗遇到敌人时是怎样的状态，例如，程序中的代码 2 可以是:

```
public void showState() {
  System. out. println ("遇到敌人狂叫，并冲向去很咬敌人"〉;
}
```

11.5 扩展练习

由于 Dog 类是面向 DogState 接口，并让小狗通过 cry 方法来体现自己的状态，因此当再增加一个实现 DogState 接口的类后(即给小狗增加一种状态)，Dog 类不需要进行修改。

请增加如下的类。

```
class MeetAnotherDog implements DogState {
  public void showState() {
    System.out.println("嬉戏");
  }
}
```

并在主类中测试小狗遇到同类时调用 cry()的效果。