第 5 章　类与对象

本章知识点：面向对象的编程思想是力图使在计算机语言中对事物的描述与现实世界中该事物的本来面目尽可能地一致。所以在面向对象的程序设计中，类(class)和对象(object)是面向对象程序设计方法中最核心的概念。类的概念是为了让程序设计语言能更清楚地描述日常生活中的事物。类是对某一类事物的描述，是抽象的、概念上的定义，而对象则是实际存在的属该类事物的具体的个体，因而也称为实例(instance)。

本章将指导读者掌握定义类、创建对象以及方法间参数传递的语法。

实验 5.1　类的定义

1. 实验目的

(1) 学习类的一般结构与类的定义。

(2) 学习类的成员变量和成员方法的声明格式。

2. 实验要求

编写一个 Java 程序，在程序中定义一个 Student 类，并且完善 Student 类的结构。

3. 程序模板

按模板要求，将【代码 1】～【代码 5】替换为相应的 Java 程序代码。

```
//文件名:Student.java
class Student
{
  String name;
  int age;
  【代码 1】              //定义一个整型属性 chinese
  【代码 2】              //定义一个整型属性 math
  【代码 3】              //定义一个整型属性 english
  int total( )
  {
    【代码 4】            //返回 chinese、math 以及 english 三个整型属性的总和
  }
  int average( )
  {
    【代码 5】            //返回 chinese、math 以及 english 三个整型属性的平均值
  }
}
```

4. 实验指导

类是对某一类事物的描述，是抽象的、概念上的定义。类是将数据和方法封装在一起的一种数据结构，其中数据表示类的属性(类的属性也称为类的数据成员或成员变量)，方法表示类的行为，所以定义类实际上就是定义类的属性与方法。在使用类之前，必须先定义它，然后才可利用所定义的类来声明相应的变量，并创建对象。

实验 5.2 对象的创建与使用

1. 实验目的

(1) 学习 Java 程序中对象的创建。

(2) 学习 Java 程序中调用对象的成员变量与成员方法。

2. 实验要求

编写一个 Java 程序，在程序中创建 Student 类的两个实例，并访问它们的属性和方法。此实验要用到 Student 类，所以必须保证实验 5.1 的 Student 类能编译通过，并且与当前实验的文件在同一个文件夹下，使程序运行结果如图 5.1 所示。

图 5.1 程序 MyObject 运行结果

3. 程序模板

按模板要求，将【代码 1】～【代码 11】替换为相应的 Java 程序代码，使之输出如图 5.1 所示的结果。

```
//文件名:MyObject.java
class MyObject
{
  public static void main(String[ ] args)
  {
    Student s1 = new Student();
    s1.name = "张三";
    s1.age = 13;
    s1.chinese = 80;
    s1.math = 90;
    s1.english = 85;
    System.out.println("你好,我叫" + s1.name + "," + "我今年" + s1.age + "岁");
    System.out.println("我的总分是" + s1.total());
```

```
        System.out.println("我的平均分是" + s1.average());
        【代码 1】              //创建 Student 类的一个实例 s2
        【代码 2】              //s2 的 name 属性值为"李四"
        【代码 3】              // s2 的 age 属性值为 12
        【代码 4】              // s2 的 chinese 属性值为 80
        【代码 5】              // s2 的 math 属性值为 90
        【代码 6】              // s2 的 english 属性值为 90
        【代码 7】              //显示 s2 的名字以及年龄
        【代码 8】              //显示 s2 的总分
        【代码 9】              //显示 s2 的平均分
        if (【代码 10】)        //如果 s1 的总分比 s2 的总分高
           System.out.println(s1.name + "的成绩好");
        else if (【代码 11】) //如果 s1 的总分比 s2 的总分低
          System.out.println(s2.name + "的成绩好");
        else
          System.out.println(s1.name + "和" + s2.name + "的成绩一样);
        }
    }
```

4. 实验指导

要创建属于某类的对象,首先声明指向“由类所创建的对象”的变量,然后利用 new 运算符创建新的对象,并指派给前面所创建的变量。对象名和对象成员之间用“.”相连,通过这种引用可以访问对象的成员。如果对象成员是成员变量,则通过这种引用方式可以获取或修改类中成员变量的值。

实验 5.3 参数传递

1. 实验目的

学习方法调用时参数的传递。

2. 实验要求

编写一个 Java 程序,在实验 5.1 的基础上为 Student 类增加一个方法,该方法能够接收参数并对 Student 类中的各个属性进行赋值,使程序运行结果如图 5.2 所示。

图 5.2 程序 MyObjectA 运行结果

3. 程序模板

按模板要求,将【代码 1】~【代码 6】替换为相应的 Java 程序代码,使之输出如图 5.2 所示的结果。

为 Student 类增加一个 setStudent()方法，该方法能接收 5 个参数并对该类的各属性进行赋值。

```
void setStudent(String n,int a,int c,int m,int e)
{
  【代码 1】              //将参数 n 赋值给属性 name
  【代码 2】              //将参数 a 赋值给属性 age
  【代码 3】              //将参数 c 赋值给属性 chinese
  【代码 4】              //将参数 m 赋值给属性 math
  【代码 5】              //将参数 e 赋值给属性 english
}
```

新建一个 MyObjectA 类如下：

```
//文件名:MyObjectA.java
class MyObjectA
{
  public static void main(String[] args)
  {
    Student s1 = new Student( );
    【代码 6】            //调用 s1 的 setStudent( )方法,参数为:"张三",13,80,90,85
    System.out.println("你好,我叫" + s1.name + "," + "我今年" + s1.age + "岁");
    System.out.println("我的总分是" + s1.total());
    System.out.println("我的平均分是" + s1.average());
  }
}
```

4. 实验指导

修改 Student 类，并新建 MyObjectA 类，编译运行。

将外部传入的参数赋值给类的成员变量，方法的形式参数与类的成员变量同名时，则需用 this 关键字来标识成员变量，如 this.age＝age。

实验 5.4 调试 Java 程序

1. 实验目的

学习使用 JDB 调试 Java 应用程序。

2. 实验要求

调试运行 DebugApp.java 程序，在调试器中使用命令查看程序运行过程中各变量的值。

3. 程序模板

```
//文件名:DebugApp.java
class Person
{
  String name;
  int age;
  public int biJiao(int a1,int a2)
  {
    if (a1 > a2)
      return a1;
```

```
    else
      return a2;
  }
}
public class DebugApp
{
  public static void main(String[] args)
  {
    int x = 1;
    int y = 2;
    Person p = new Person();
    p.name = "张三";
    p.age = 20;
    int s = p.biJiao(x,y);
    System.out.println("较大的数是" + s);
  }
}
```

4. 实验指导

许多 Java 开发环境都提供了便捷的调试器,各种调试器原理基本都一样。这里介绍 JDK 包含的 JDB 调试器,JDB 调试器用户界面非常有限,用户可以尽量选择其他便捷调试器。

要想使用 JDB,首先必须用-g 选项编译程序,编译器会在文件中添加一些局部变量的名字和其他调试信息。

```
javac -g DebugApp.java
```

输入命令运行调试器:

```
jdb DebugApp
```

调试器启动后,命令提示符界面如图 5.3 所示。在调试器运行状态下,可以输入“?”查看所有的调试器命令,如图 5.4 所示。

图 5.3 启动调试器

要利用调试器调试程序,必须设置一个或多个断点,然后运行程序,当程序执行到设置的断点处时会停止运行,此时可以查看变量的值。

设置、清除断点,可以使用如下方法。

- stop in <类 ID>.<方法>[(参数类型,…)]: 为类的方法设置断点。
- stop at <类 ID>:<行>: 为源程序的行设置断点。

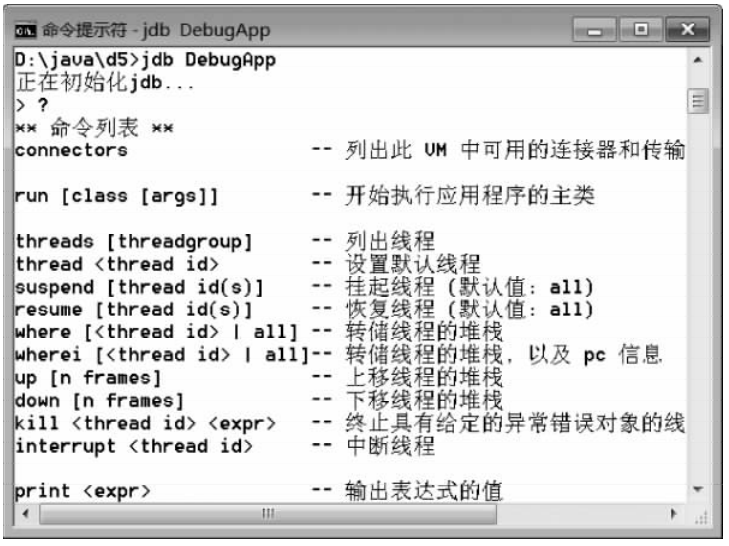

图 5.4 调试器命令

- clear <类 ID>.<方法>[(参数类型,…)]：清除方法中的断点。
- clear <类 ID>:<行>：清除行中的断点。
- clear：列出断点。

例如,为程序模板中的 DebugApp 程序第 22 行设置断点：

```
stop at DebugApp:22
```

现在运行程序,直到遇到断点,输入命令：

```
run
```

调试器将在程序的第 22 行开始位置中断,如图 5.5 所示。可以使用 list 命令显示当前行,以及之前或之后的若干行内容,如图 5.6 所示。

图 5.5 调试器的中断命令

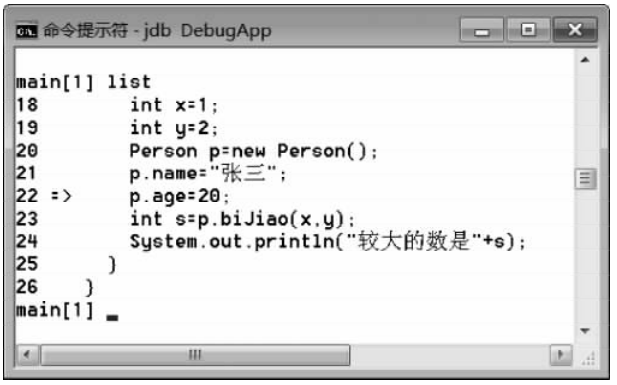

图 5.6 调试器的 list 命令

此时注意,箭头“=>”所指的语句“p.age=20”并未执行。

利用 print 命令可以查看此时变量的值,利用 dump 命令可以查看更加详细的信息,如图 5.7 所示。若让程序继承执行下去,则输入 next 命令,执行第 22 行语句,如图 5.8 所示。

图 5.7 查看变量的 print 命令

图 5.8 调试器的 next 命令

第 23 行语句调用了 biJiao()方法，如果不希望调试进入方法内部，可以输入 next 命令；如果希望调试进入方法内部，可以输入 step 命令，如图 5.9 所示。退出调试状态，可以使用 quit 命令。

图 5.9 调试器的 step 命令

next、step 命令常用操作如下。

- step：执行当前行。
- step up：执行到当前方法返回其调用方法。
- stepi：执行当前指令。
- next：跳过一行(跨过调用)。
- cont：从断点处继续执行。

第 5 章实验参考答案

实验 5.1：

```
【代码 1】:int chinese;
【代码 2】:int math;
【代码 3】:int english;
【代码 4】:return chinese + math + english;
【代码 5】:return (int)total()/3;
```

实验 5.2：

```
【代码 1】:Student s2 = new Student();
【代码 2】:s2.name = "李四";
【代码 3】:s2.age = 12;
【代码 4】:s2.chinese = 80;
【代码 5】:s2.math = 90;
【代码 6】:s2.english = 90;
【代码 7】:System.out.println("你好,我叫" + s2.name + "," + "我今年" + s2.age + "岁");
【代码 8】:System.out.println("我的总分是" + s2.total());
【代码 9】:System.out.println("我的平均分是" + s2.average());
【代码 10】:s1.total()> s2.total()
【代码 11】:s1.total()< s2.total()
```

实验 5.3：

```
【代码 1】:name = n;
【代码 2】:age = a;
【代码 3】:chinese = c;
【代码 4】:math = m;
【代码 5】:english = e;
【代码 6】:s1.setStudent("张三",13,80,90,85);
```