高等职业技术教育“十二五”规划教材

Java程序设计教程

Java CHENGXU SHEJI JIAOCHENG

主　编／张　望　梅青平　柯　灵
主　审／乐明于
副主编／李咏霞　周　敏

西南交通大學出版社
Http://press.swjtu.edu.cn

高等职业技术教育“十二五”规划教材

Java 程序设计教程

主　编　张　望　梅青平　柯　灵

主　审　乐明于

副主编　李咏霞　周　敏

参　编　罗　粮　刘旭东　谷丽华　丁丽丽

西南交通大学出版社

图书在版编目（CIP）数据

Java 程序设计教程 / 张望，梅青平，柯灵主编. —
成都：西南交通大学出版社，2014.2
高等职业技术教育“十二五”规划教材
ISBN 978-7-5643-2780-4

Ⅰ. ①J… Ⅱ. ①张… ②梅… ③柯… Ⅲ. ①JAVA 语言－程序设计－高等职业教育－教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2013）第 295209 号

高等职业技术教育“十二五”规划教材

Java 程序设计教程

主编　张　望　梅青平　柯　灵

*

责任编辑　张华敏
助理编辑　宋彦博
特邀编辑　黄庆斌
封面设计　墨创文化

西南交通大学出版社出版发行
（四川省成都市金牛区交大路 146 号　邮政编码：610031　发行部电话：028-87600564）
http: //press.swjtu.edu.cn

四川森林印务有限责任公司印刷

*

成品尺寸：185 mm × 260 mm　　印张：13
字数：325 千字
2014 年 2 月第 1 版　　2014 年 2 月第 1 次印刷
ISBN 978-7-5643-2780-4
定价：28.50 元

前　言

与其他课程相比，程序设计课程要求学生从例子中学习、从实践中学习、从错误中学习，因此需要花费大量的时间来编写程序、调试程序并修改错误。

对刚刚接触程序设计的新手来说，学习 Java 与学习其他高级程序设计一样。学习程序设计的基本目的就是要培养大家描述实际问题的程序化解决方案的关键技能，并通过语句、循环、方法和数组将解决方案变成程序。

读者一旦掌握了使用循环、方法和数组编写程序的基本技能，就可以开始学习如何使用面向对象的方法开发大型程序了。

本书提供了以下章节顺序：

* 第 1 章“面向对象概述”，介绍了类与对象的基本概念，并介绍了 UML。

* 第 2 章“第一个 Java 程序”，通过一个实际的例子，向读者展示了一个 Java 程序所需要的基本元素，让读者从感性上对 Java 有一个初步认识。

* 第 3 章“数据类型、常量、变量”，介绍了 Java 支持的基本数据类型，static 关键字和 final 关键字的含义，以及变量的初始化。

* 第 4 章“操作符”，介绍了 Java 支持的操作符以及对这些操作符的分类，特别介绍了 Java 中对字符串所支持的操作。

* 第 5 章“流程控制”，介绍了 Java 所支持的流程控制语句，由于本章内容和其他语言有很多的共性，因此描述相对较少。

* 第 6 章“类和对象”，通过一个实际的例子，向读者展示了类和对象的语义，重点是通过该例子使读者理解到面向对象的基本思想。

* 第 7 章“继承”，本章是面向对象编程思想的重点内容，重点要理解继承所描述的语义，而不仅仅是语法，在此基础上，介绍了多态的语法和语义。

* 第 8 章“抽象类、接口”，描述了抽象类和接口的基本概念，以及抽象类和接口的区别。在本章最后，给出了一个实际的例子，通过该例子读者可以更加深刻地体会到面向对象编程思想。

* 第 9 章“内部类”，内部类是 Java 的特性之一，本章介绍了内部类的概念和基本语法，并深刻讨论了内部类背后的编程思想。

* 第 10 章“多线程”，介绍了 Java 的多线程机制，并重点讨论了 Java 多线程的同步问题。

* 第 11 章“数组和字符串”，介绍了一维、多维数组，以及 Java 对字符串的支持。

* 第 12 章“Java 集合”，介绍了 Java 开发中常用的集合。

* 第 13 章“异常”，介绍了 Java 的异常处理机制。

* 第 14 章“Java I/O 系统”，本章是本书的重点内容，详细讨论了 Java 对 I/O 的支持。

* 第 15 章“Java 网络编程”，介绍了 Java Socket 编程的基本知识。

本书由重庆城市管理职业学院张望、梅青平、柯灵担任主编，重庆城市管理职业学院乐明于担任主审，重庆城市管理职业学院李咏霞和重庆祖卡科技有限公司周敏担任副主编，重庆城市管理职业学院罗粮、刘旭东、谷丽华、重庆机械电子技师学院丁丽丽参编。

为方便老师教学，本书配有电子教案及源代码，有需要的老师请到西南交通大学出版社网站下载或 E-mail：420930692@qq.com。

由于作者水平有限，书中难免存在不当之处，恳请广大读者批评指正。如有批评和建议请发至：zw2chm@163.com。

编　者

2013.12

目　录

第 1 章　面向对象概述

1.1　结构化的软件开发方法简介

在 20 世纪 60 年代中期爆发了众所周知的软件危机。为了克服这一危机，在 1968、1969 年连续召开的两届著名的 NATO 会议上提出了软件工程这一概念，并在以后不断发展、完善。与此同时，软件研究人员也在不断探索新的软件开发方法。至今已形成了八类软件开发方法。

1978 年，E. Yourdon 和 L. L. Constantine 提出的结构化方法（SASD 方法）是八类方法中的一类，也可称为面向功能的软件开发方法或面向数据流的软件开发方法。1979 年 TomDeMarco 对此方法作了进一步完善。SASD 方法是 20 世纪 80 年代使用得最广泛的软件开发方法。它首先用结构化分析（SA）方法对软件进行需求分析，然后用结构化设计（SD）方法进行总体设计，最后是结构化编程（SP）。这一方法不仅开发步骤明确，而且给出了两类典型的软件结构（变换型和事务型），使软件开发的成功率大大提高，从而深受软件开发人员的青睐。

在进行结构化设计时，首先要考虑整个软件系统的功能，然后按模块划分的基本原则对功能进行分解，把整个软件系统划分为多个模块，每个模块实现了特定的子功能。在完成了所有的模块设计以后，把这些模块拼装起来，就构成了整个软件系统。软件系统可以看做是多个子系统的集合，每个子系统都具有输入输出功能模块。

结构化程序设计属于自顶向下的设计，在设计阶段就必须考虑如何实现系统的功能，在功能分解的过程中不断实现系统的功能。当用户需求发生变化时，就需要修改模块的结构，有时甚至将整个设计全部推翻。在进行结构化编程时，程序的主体是方法，方法是组成程序的基本单位，每个方法都是具有输入输出的子系统。方法的输入数据主要来自于方法的参数，即全局变量和常量。方法的输出数据主要包括方法的返回值以及指针类型的方法参数。一组相关的方法组成大的功能模块。

下面举例来说明结构化程序设计的方法，假设程序的功能是计算圆、矩形、三角形的面积。代码如下：

```
#include<stdio.h>
#define CIRCLE    1
#define RECTANGLE 2
#define TRIANGLE  3
double getCircleArea(){}        //实现细节略
double getRectangleArea(){}     //实现细节略
```

```
double getTriangleArea(){}     //实现细节略
double selectShape()
{
    int shape;
    scanf("%d",&shape);
    switch(shape)
    {
    case CIRCLE:
        return getCircleArea();
    case RECTANGLE:
        return getRectangleArea();
    case TRIANGLE:
        return getTriangleArea();
    }
}
void main()
{
    selectShape();

}
```

例程 1.1　Shpae.c

如果此时需求发生变化，需要增加一个求正六边形面积的功能，那么需要对已有的代码做多处改动。

（1）在整个系统的范围内增加一个常量，如：

```
#define REGULARHEXAGON  4
```

（2）在整个系统的范围内增加一个求正六边形面积的方法，如：

```
double getRegularhexagonArea ( )
```

（3）在选择形状模块 selectShape（ ）内增加以下逻辑：

```
case  REGULARHEXAGON:
return getRegularhexagonArea ( ) ;
```

由此可见，结构化程序开发方法制约了软件的可扩展性，模块之间的耦合度高，修改其中一个模块将会影响到其他模块。导致这种缺陷的根本原因在于：

- 自顶向下地按照功能来划分软件模块。软件功能不是一成不变的，会随着用户需求的变化而不断发生变化，这就使得软件在设计阶段就难以设计出稳定的系统结构。
- 软件系统中最小的子系统是方法。方法与一部分相关的数据分离，全局变量和常量分布在系统的各个角落，所有方法均可访问，这就使得各个系统之间的独立性降低，从而使得软件不可扩展。

1.2　面向对象的软件开发方法简介

面向对象的软件开发方法把软件系统看成是各种对象的集合，对象就是最小的子系统，一组相关的对象能够组合成更复杂的子系统。面向对象的软件开发方法具有以下优点：

（1）把软件系统看成是各种对象的集合，这更接近人类的思维方式。

（2）软件需求的变动往往是功能的变动，而功能的执行者——对象一般不会有大的变化。这使得按照对象设计出来的系统结构比较稳定。

（3）对象包括属性（数据）和行为（方法），对象把数据和方法的具体实现方式一起封装起来，这就使得方法和与之相关的数据不再分离，提高了每个子系统的相对独立性，从而提高了软件的可维护性。

（4）支持封装、抽象、继承和多态，提高了软件的可重用性、可维护性和可扩展性。

1.2.1　对象模型

在面向对象的分析和设计阶段，主要工作是建立对象模型。建立对象模型既包括自底向上的抽象过程，也包括自顶向下的分解过程。

（1）自底向上的抽象。

建立对象模型的第一步是从问题领域的陈述入手。分析需求的过程与对象模型的形成过程一致，开发人员与用户交谈是从用户熟悉的问题领域中的事物（具体实例）开始的，这就使得开发人员能够更容易搞清用户需求，然后再建立正确的对象模型。开发人员需要进行以下自底向上的抽象思维。

- 把问题领域中的事物抽象为具有特定属性和行为的对象。
- 把具有相同属性和行为的对象抽象为类。
- 若多个类之间存在一些共性（具有相同属性和行为），把这些共性抽象到父类中。

在自底向上的抽象过程中，为了使子类能更好地继承父类的属性和行为，可能需要自顶向下的修改，从而使整个类体系更加合理。由于类体系的构造是从具体到抽象，再从抽象到具体，符合人们的思维规律，因此能够更快、更方便地完成任务。

（2）自顶向下的分解。

在建立对象模型的过程中，也包括自顶向下的分解。例如对于计算机系统，首先识别出主机对象、显示器对象、键盘对象和打印机对象等。接着对这些对象再进一步分解，例如主机对象由处理器对象、内存对象、硬盘对象和主板对象组成。系统的进一步分解因有具体的对象为依据，因此分解过程比较明确，而且也相对容易。而面向对象建模也具有自顶向下开发方法的优点，既能有效控制系统的复杂性，又能同时避免结构化开发方法中功能分解的困难和不确定性。

1.2.2　UML 简介

1997 年，OMG（Object Management Group），即对象管理组织发布了统一建模语言

（Unified Modeling Language，UML）。UML 的目标之一就是为开发团队提供标准通用的设计语言来开发和构建计算机应用。UML 提出了一套 IT 专业人员期待多年的统一标准建模符号。通过使用 UML，这些人员能够阅读和交流系统架构和设计规划——就像建筑工人多年来所使用的建筑设计图一样。

UML 提供了多种类型的模型描述图，使用这些图时，它使得开发中的应用程序更易理解。UML 的内涵远不只是这些模型描述图，但是对于入门来说，这些图对于这门语言及其用法的基本原理提供了很好介绍。通过把标准的 UML 图放进工作产品中，精通 UML 的人员就更加容易加入项目并迅速进入角色。最常用的 UML 图包括用例图、类图、序列图、状态图、活动图、组件图和部署图。

（1）用例图。

用例图描述了系统提供的一个功能单元。用例图的主要目的是帮助开发团队以一种可视化的方式理解系统的功能需求，包括基于基本流程的“角色”（actors，也就是与系统交互的其他实体）关系，以及系统内用例之间的关系。用例图一般表示出用例的组织关系，即要么是整个系统的全部用例，要么是完成具体功能（例如，所有安全管理相关的用例）的一组用例。要在用例图上显示某个用例，可绘制一个椭圆，然后将用例的名称放在椭圆的中心或椭圆下面的中间位置。要在用例图上绘制一个角色（表示一个系统用户），可绘制一个人形符号。角色和用例之间的关系使用简单的线段来描述，如图 1.1 所示。

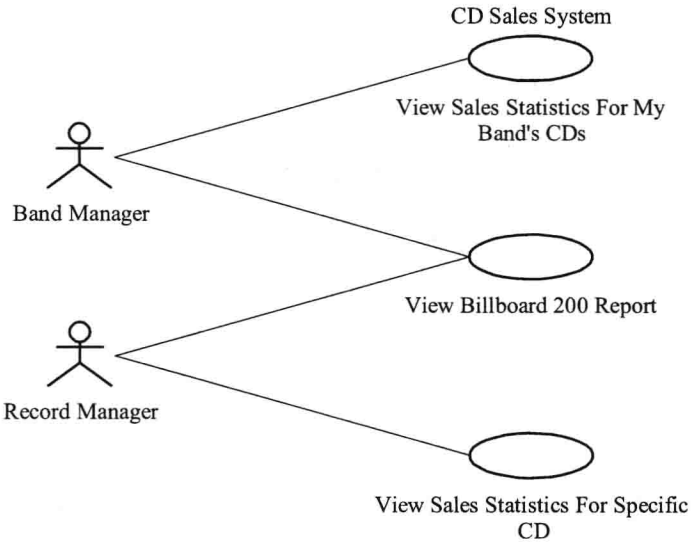

图 1.1　用例图

（2）类图。

类图表示不同的实体（人、事物和数据）如何彼此相关，它显示了系统的静态结构。类在类图上用包含三个部分的矩形来描述，如图 1.2 所示。最上面的部分显示类的名称，中间部分包含类的属性，最下面的部分包含类的操作（方法）。根据经验，几乎每个人都知道这个类图是什么，但是大多数人都不能正确地描述类之间的关系。对于如图 1.2 所示的类图，应该使用带有顶点指向父类的箭头的线段来表示继承关系，并且箭头应该是一个完全的三角形。

如果两个类都彼此知道对方，则应该使用实线来表示它们之间的关联关系；如果只有其中一个类知道该关联关系，则使用开箭头表示。

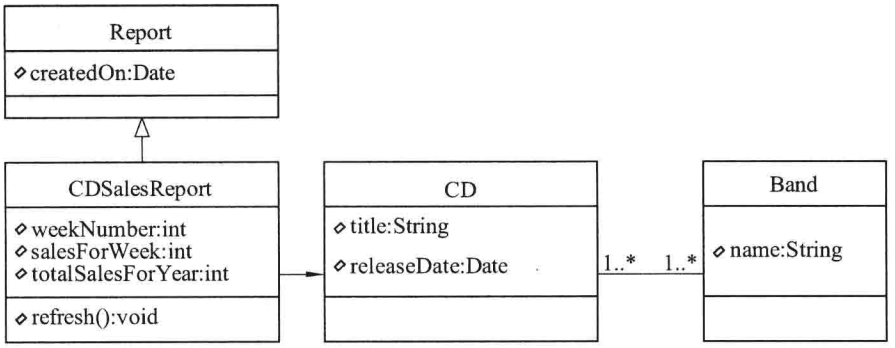

图 1.2　类图

（3）序列图。

序列图用于显示具体用例（或者是用例的一部分）的详细流程。它几乎是自描述的，并且显示了流程中不同对象之间的调用关系，同时还可以很详细地显示对不同对象的不同调用。

序列图有两个维度：其中垂直维度以发生的时间顺序显示消息调用的序列；水平维度显示消息被发送到的对象实例。

序列图的绘制如图 1.3 所示。横跨图的顶部，每个框表示每个类的实例（对象）。在框中，类实例名称与类名称之间用空格、冒号、空格来分隔，如 myReportGeneratorv：ReportGenerator。如果某个类实例向另一个类实例发送一条消息，则绘制一条具有指向接收类实例的开箭头的连线，并把消息（方法）的名称放在连线上面。对于某些特别重要的消息，可以绘制一条具有指向发起类实例的开箭头的虚线，将返回值标注在虚线上。

在阅读序列图时，应从左上角启动序列的“驱动”类实例开始，然后顺着每条消息往下阅读。虽然如图 1.3 所示的序列图显示了每条被发送消息的返回消息，但这是可选的。

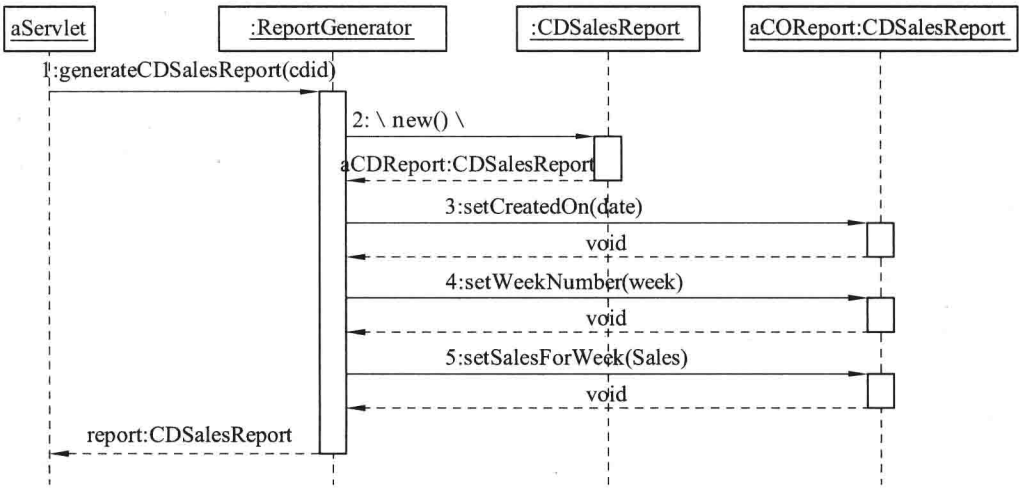

图 1.3　序列图

（4）状态图。

状态图表示某个类所处的不同状态和该类的状态转换信息。如图 1.4 所示，状态图的符号集包括 5 个基本元素：初始起点使用实心圆来绘制；状态之间的转换使用具有开箭头的线段来表示；状态使用圆角矩形来表示；判断点使用空心圆来表示；一个或者多个终止点使用内部包含实心圆的圆来表示。要绘制状态图，首先绘制起点和一条指向该类的初始状态的转换线段。状态本身可以在图上的任意位置绘制，然后只需使用状态转换线条将它们连接起来。

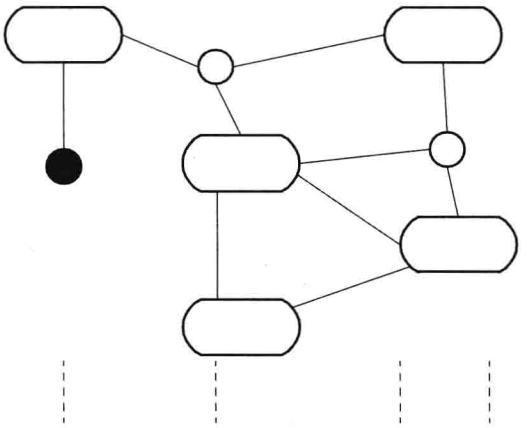

图 1.4 状态图

（5）活动图。

活动图表示在处理某个活动时，两个或者更多类对象之间的过程控制流。活动图可用于在业务单元的级别上对更高级别的业务过程进行建模，或者对低级别的内部类操作进行建模。活动图适合用于对较高级别的过程建模，比如公司当前在如何运作业务等。与序列图相比，虽然活动图在表示上的技术性要求不那么高，但有业务头脑的人们往往能够更快速地理解它们。

活动图的符号集与状态图中使用的符号集类似。像状态图一样，活动图也从一个连接到初始活动的实心圆开始。活动是通过一个将活动的名称包含其内的圆角矩形来表示的。活动可以通过转换线段连接到其他活动或者判断点，这些判断点连接到由判断点的条件所保护的不同活动。结束过程的活动连接到一个终止点。作为一种选择，活动可以分组为泳道（swimlane），泳道用于表示实际执行活动的对象，如图 1.5 所示。

（6）组件图。

组件图提供系统的物理视图，其用途是显示系统中的软件对其他软件组件（例如，库函数）的依赖关系。组件图可以在一个非常高的层次上显示。

（7）部署图。

部署图表示软件系统如何部署到硬件环境中，其用途是显示系统不同的组件将在何处物理地运行，以及它们之间如何通信。由于部署图是对物理运行情况进行建模，因此系统的生产人员可以很好地利用这种图。

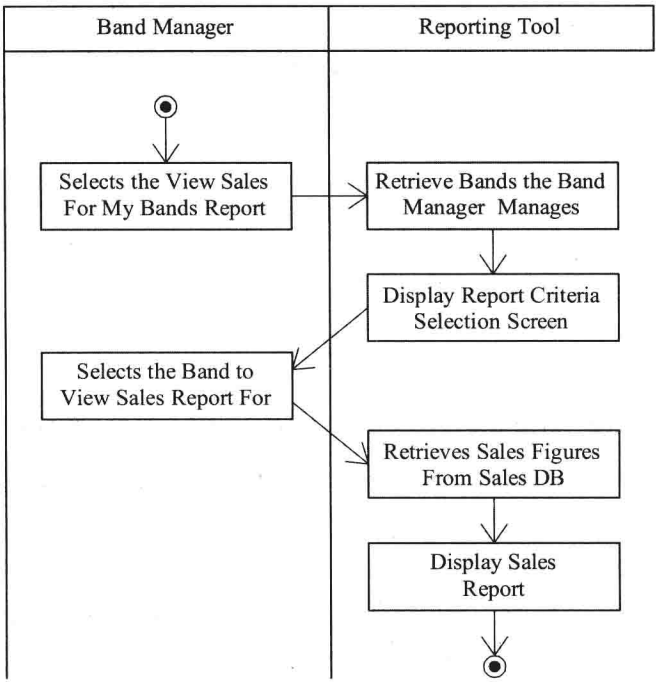

图 1.5　活动图

1.2.3　建模工具

Rational Rose 是 Rational 公司开发的一种可视化建模工具，它采用 UML 语言来构建对象模型，是分析和设计面向对象软件系统的强有力工具。相应用法可参见相关书籍。

1.3　面向对象开发中的核心思想和概念

在面向对象的软件开发过程中，开发者的主要任务就是先建立模拟问题领域的对象模型，然后通过程序代码来实现该对象模型。如何用程序代码来实现对象模型，并且保证软件系统的可重用性，可扩展性和可维护性呢？本节主要阐述面向对象开发的核心思想和概念，这些核心思想为从事面向对象的软件开发实践提供了理论武器。

1.3.1　类、类型

类是一组具有相同属性和行为的对象的抽象。类及类的关系构成了对象模型的主要内容。对象模型用来模拟问题领域，Java 程序实现对象模型，Java 程序运行在 Java 虚拟机提供的运行时环境中，Java 虚拟机运行在计算机上，如图 1.6 所示。

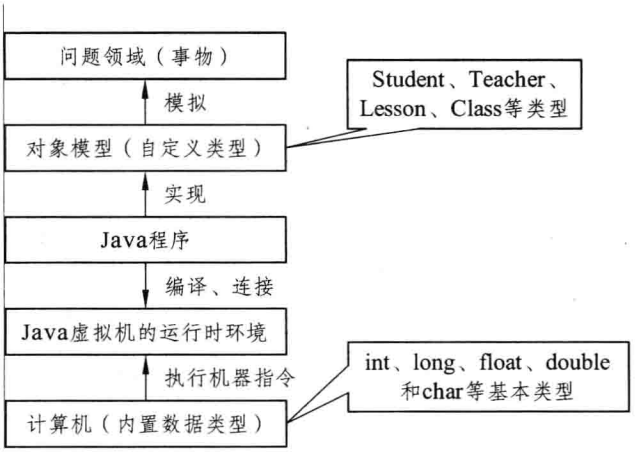

图 1.6 对象模型中的类型到计算机的内置数据类型的映射

由于计算机受存储单元的限制，只能表示和操作一些基本的数据类型，如整数、字符和浮点数。对象模型中的类可以看做是开发人员自定义的数据类型，Java 虚拟机的运行时环境封装了把自定义的数据类型映射到计算机的内置数据类型的过程，使得开发人员不受计算机的内置数据类型的限制，对任意一种问题领域，都可以方便地根据先识别对象，再进行分类（创建任意的数据类型）的思路来建立对象模型。

面向对象编程的主要任务就是定义对象模型中的各个类。例如例程 1.2 是矩形类的定义。

```
public class Rectangle{

    private double width;
    private double high;
    public Rectangle(double width, double high){
        this.width = width;
        this.high = high;
    }
    public double getArea(){
        return width*high;
    }
}
```

例程 1.2 Rectangle.java

如何创建矩形对象呢？Java 语言采用 new 语句创建对象，new 语句会调用对象的构造方法。以下程序代码创建了一个宽为 5、高为 3 的矩形对象 rect。

```
Rectangle rect = new Rectangle(3,5);
```

在运行时环境中，Java 虚拟机首先把 Rectangle 类的代码加载到内存中，然后根据这个模板来创建 Rectangle 对象；也就是说，对象是类的实例，类是对象的模板。