高等学校Java课程系列教材

Java课程设计

（第3版）

◎ 耿祥义 张跃平 编著

动物换位　单词簿　广告墙　走迷宫　魔板游戏　扫雷游戏

保存计算过程的计算器　标准化试题训练系统

清华大学出版社

高等学校Java课程系列教材

Java课程设计

（第3版）

◎ 耿祥义 张跃平 编著

清華大學出版社

北京

内 容 简 介

在掌握了 Java 基本知识后，可以通过课程设计来巩固和提高 Java 编程技术，本书就是针对这一目的编写的。

本书以 8 个具有一定代表性的课程设计题目为框架，体现 MVC 模式和面向对象的设计思想，强化内置 Derby 数据库、网络 MySQL 数据库以及 Excel 工作簿在应用中的作用；设计思路清晰，便于理解，可帮助读者提高设计能力以及面向对象的编程能力；每个课程设计都按照 MVC 模式展开，每章内容都由设计要求、数据模型、简单测试、视图设计、GUI 程序、程序发布和课设题目 7 个部分构成；各个课程设计题目相互独立，读者可以从任何一个课程设计题目开始阅读本书，可以按照本书布置的课程设计作业来开发一个软件，也可以参考这些课程设计题目设计类似的软件。

本书不仅可以作为理工科各个专业 Java 课程设计的教材，也可作为自学者提高编程能力的参考书。

图书在版编目（CIP）数据

Java 课程设计 / 耿祥义，张跃平编著. —3 版. —北京：清华大学出版社，2018（2018.7重印）
（高等学校 Java 课程系列教材）
ISBN 978-7-302-48864-4

Ⅰ. ①J…　Ⅱ. ①耿…　②张…　Ⅲ. ①JAVA 语言 -程序设计　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2017）第 287202 号

责任编辑：魏江江　王冰飞
封面设计：刘　键
责任校对：徐俊伟
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张：**13　　**字　　数：**325 千字
版　　次：2004 年 1 月第 1 版　2018 年 1 月第 3 版　　**印　　次：**2018 年 7 月第 2 次印刷
定　　价：34.50 元

产品编号：077170-01

第3版前言

本书以8个具有一定代表性的课程设计题目为框架，从各个方面展示了Java在应用系统开发中的实用技术。在第3版特别体现了MVC模式，对代码全部进行了新的设计和编写，充分体现面向对象的设计思想。另外，本书特别增加了使用数据库的训练，如内置Derby数据库、网络MySQL数据库以及操作Excel工作簿的新题目，并舍弃了第2版的一些题目。

本书中的课程设计题目互相独立，读者可以从任何一个课程设计题目开始阅读本书，每个课程设计都按照MVC模式展开，设计思路清晰，便于理解，可帮助读者提高设计能力以及面向对象的编程能力。本书每章内容都由设计要求、数据模型、简单测试、视图设计、GUI程序、程序发布和课设题目7个部分构成。读者可以按照本书布置的课程设计作业来开发一个软件，也可以参考这些课程设计题目设计类似的软件。读者阅读调试完8个课程设计后（建议至少阅读调试完前5个课程设计），在设计能力和编程技术能力方面一定会有收获，在此基础上再完成一个教材建议的课设题目或自己构思一个难度相当的课设题目。

虽然本书是《Java 2实用教程（第5版）》的配套教材，但也可以独立使用。

本书的全部代码都是作者亲自编写并且在JDK1.8运行环境下调试通过。本书代码仅供读者学习Java使用，不得以任何方式抄袭出版。大家也可关注作者微信公众号java-violin或访问作者个人网站http://gengxiangyi.lingw.net获得有关资料。

希望本书能对读者学习Java有所帮助，并请读者批评指正。

作　者

2017年10月

目录

第 8 章 扫雷游戏

第 1 章　动物换位

1.1　设计要求

设计 GUI 界面的动物换位游戏，游戏结果是让左、右两组动物交换位置。具体要求如下：

① 在水平排列的 7 个位置上左、右各有 3 个类型相同的动物，中间的位置上没有动物。左边动物将其右侧视为自己的前进方向，右边动物将其左侧视为自己的前进方向。

② 单击一个动物，如果该动物的前方位置上没有动物，该动物就跳跃到该位置上，如果该动物的前方位置上有其他动物，但相隔一位的位置上没有其他动物，该动物就越过自己前面的动物跳跃至该隔位上，其他情形下该动物不能跳跃（跳跃时不能越过两个位置）。

③ 左面的动物只能向右方跳跃，右面的动物只能向左方跳跃。

④ 单击“撤销”按钮撤销上一次移动的动物，单击“重新开始”按钮可重新开始游戏。

程序运行的参考效果图如图 1.1 所示。

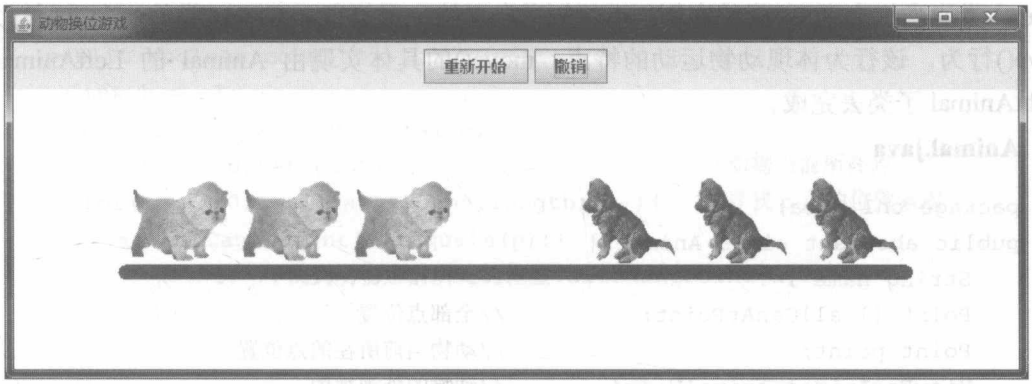

图 1.1　动物换位游戏

注意　按照 MVC-Model View Control（模型，视图，控制器）的设计思想展开程序的设计和代码的编写。数据模型部分相当于 MVC 中的 Model 角色，视图设计部分给出的界面部分相当于 MVC 中的 View，视图设计部分给出的事件监视器相当于 MVC 中的 Control。

1.2　数据模型

根据系统设计要求在数据模型部分编写了以下类。

- Animal 类：封装了左、右动物共同的属性和行为。
- LeftAnimal 类：Animal 的子类，封装了左边动物的独特行为。

- RightAnimal 类：Animal 的子类，封装了右边动物的独特行为。
- Point 类：刻画动物可以到达的位置。
- ViewForAnimal 类：封装制作动物视图的方法。

数据模型部分涉及的类的 UML 图如图 1.2 所示。

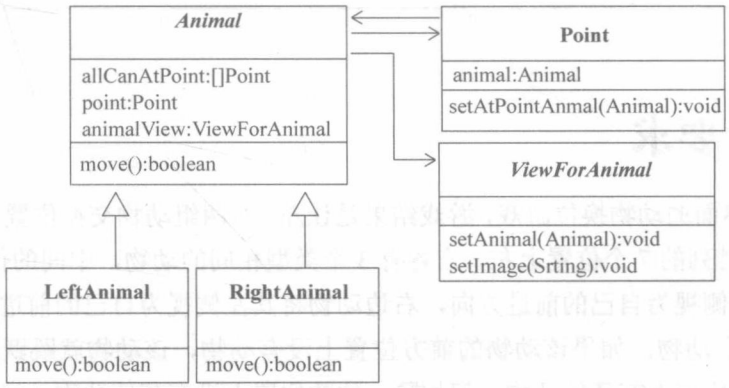

图 1.2　类的 UML 图

❶ 动物相关类

1）Animal 类

根据设计要求提出的动物的特点设计了以下 Animal 类用来刻画动物的数据和行为。Animal 类应当组合 Point 类的实例，以便知道自己的位置信息。Animal 类的一个重要行为是 move()行为，该行为体现动物运动的特点，move()的具体实现由 Animal 的 LeftAnimal 和 RightAnimal 子类去完成。

Animal.java

```
package ch1.data;
public abstract class Animal {
    String name ;
    Point [] allCanAtPoint;              //全部点位置
    Point point;                         //动物当前所在的点位置
    ViewForAnimal animalView;            //动物的外观视图
    public void setAtPoint(Point p) {
        if(p!=null){
            point = p;
            point.setIsHaveAnimal(true);
            point.setAtPointAnimal(this);
        }
    }
    public Point getAtPoint() {
        return point;
    }
    public void setAllCanAtPoint(Point [] point){
        allCanAtPoint = point;
    }
```

```
    public void setAnimalView(ViewForAnimal animalView) {
        this.animalView = animalView;
        animalView.setAnimal(this);
    }
    public ViewForAnimal getAnimalView() {
        return animalView;
    }
    public void setName(String s) {
        name = s;
    }
    public String getName() {
        return name;
    }
    public abstract boolean move();
}
```

2）LeftAnimal 和 RightAnimal 子类

Animal 类的 move()方法是 abstract 方法，move()的具体实现由 Animal 的 LeftAnimal 和 RightAnimal 子类按照设计要求去完成。代码分别如下。

LeftAnimal.java

```
package ch1.data;
public class LeftAnimal extends Animal{
   public boolean move(){
      int k = -1;
      boolean successMove = false;
      Point p = getAtPoint();                          //动物当前所在点
      for(int i=0;i<allCanAtPoint.length;i++){         //寻找 p 点的位置索引
         if(allCanAtPoint[i].equals(p)){
            k = i; //找到动物当前所处的位置:allCanAtPoint[k]
            break;
         }
      }
      if(k==allCanAtPoint.length-1){                   //已经在最右面的点位置
          return false;
      }
      if(allCanAtPoint[k+1].isHaveAnimal()==false) {//前面位置上没有动物
         this.setAtPoint(allCanAtPoint[k+1]); //动物到达 allCanAtPoint[k+1]点
         successMove = true;
         p.setAtPointAnimal(null);                     //p 点位置为无动物
         return successMove ;
      }
      if((k+1)==allCanAtPoint.length-1){               //前面位置上是已经到达终点的动物
          return false;
      }
      if(allCanAtPoint[k+2].isHaveAnimal()==false) {//前方隔位上没有动物
```

```
            this.setAtPoint(allCanAtPoint[k+2]);
            successMove = true;
            p.setAtPointAnimal(null);
            return successMove ;
        }
        return successMove ;
    }
}
```

RightAnimal.java

```
package ch1.data;
public class RightAnimal extends Animal{
    public boolean move(){
        int k = -1;
        boolean successMove = false;
        Point p = getAtPoint();                          //动物当前所在点
        for(int i=0;i<allCanAtPoint.length;i++){   //寻找 p 点的位置索引
            if(allCanAtPoint[i].equals(p)){
                k = i;
                break;
            }
        }
        if(k==0){  //已经在最左面的点位置
            return false;
        }
        if(allCanAtPoint[k-1].isHaveAnimal()==false) {//前面位置上没有动物
            this.setAtPoint(allCanAtPoint[k-1]);//动物到达 allCanAtPoint[k-1]点
            successMove = true;
            p.setAtPointAnimal(null);                    //p 点位置为无动物
            return successMove ;
        }
        if((k-1)==0){                                    //前面位置上是已经到达终点的动物
            return false;
        }
        if(allCanAtPoint[k-2].isHaveAnimal()==false) {//前方隔位上没有动物
            this.setAtPoint(allCanAtPoint[k-2]);//动物到达 allCanAtPoint[k-2]点
            successMove = true;
            p.setAtPointAnimal(null);                    //p 点位置为无动物
            return successMove ;
        }
        return successMove ;
    }
}
```

❷ 位置相关类

根据设计要求提出的动物运动位置的特点设计了以下 Point 类，用来刻画和位置相关的数据和行为，Point 类的实例称为一个点，是动物可以到达的点。Point 类的实例应当组合 Animal 的实例，以便告知在当前位置上的是 Animal 的哪个实例，即该点上是哪个动物。

Point.java

```
package ch1.data;
public class Point{
    int x,y;
    boolean haveAnimal;
    Animal animal=null;                    //在该点位置上的动物
    public void setX(int x){
        this.x=x;
    }
    public void setY(int y){
        this.y=y;
    }
    public boolean isHaveAnimal(){
        return haveAnimal;
    }
    public void setIsHaveAnimal(boolean boo){
        haveAnimal=boo;
    }
    public int getX(){
        return x;
    }
    public int getY(){
        return y;
    }
    public void setAtPointAnimal(Animal animal){
        this.animal=animal;
        if(animal!=null) {
            haveAnimal = true;
        }
        else {
            haveAnimal = false;
        }
    }
    public Animal getAtPointAnimal(){
        return animal;
    }
}
```

❸ 视图相关类

动物需要一个外观提供给游戏的玩家，以便玩家单击动物来运动当前动物。ViewForAnimal 类是 javax.swing.JComponent 的子类，以便其子类有具体的外观。另外 ViewForAnimal 类应当组合 Animal 类的实例，以便确定为哪个 Animal 实例提供视图，即该视图是哪个动物的视图。ViewForAnimal 的子类将在视图（View）设计部分给出，见 1.4 节中的 AnimalView 类。

ViewForAnimal.java

```
package ch1.data;
import javax.swing.JPanel;
public abstract class ViewForAnimal extends JPanel {
    public abstract void setAnimal(Animal animal);
    public abstract void setImage(String name);
    public abstract Animal getAnimal();
    public abstract void setAnimalViewLocation(int x,int y);
    public abstract void setAnimalViewSize(int w,int h);
}
```

1.3 简单测试

按照源文件中的包语句将相关的 Java 源文件保存到以下目录中：

```
D:\ ch1\data
```

编译各个源文件，例如：

```
D:\>javac/ch1/data/Animal.java
```

或一次编译全部源文件：

```
D:\>javac/ch1/data/*.java
```

把 1.2 节给出的类看作一个小框架，下面用框架中的类编写一个简单的应用程序，测试左、右动物运动换位，即在命令行表述对象的行为过程，如果表述成功（如果表述困难，说明数据模型不是很合理），那么就为以后的 GUI 程序设计提供了很好的对象功能测试，在后续的 GUI 设计中，重要的工作仅仅是为某些对象提供视图界面，并处理相应的界面事件而已。

将 AppTest.java 源文件按照包名保存到以下目录中：

```
D:\ch1\test
```

编译源文件：

```
D:\>javac ch1/test/AppTest.java
```

运行 AppTest 类（运行效果如图 1.3 所示）：

```
D:\>java ch1.test.AppTest
```

```
D:\>java ch1.test.AppTest
猫0猫1猫2    狗0狗1狗2
猫0猫1猫2狗0    狗1狗2
猫0猫1    狗0猫2狗1狗2
猫0    猫1狗0猫2狗1狗2
猫0狗0猫1    猫2狗1狗2
猫0狗0猫1狗1猫2    狗2
猫0狗0猫1狗1猫2狗2
猫0狗0猫1狗1    狗2猫2
猫0狗0    狗1猫1狗2猫2
    狗0猫0狗1猫1狗2猫2
狗0    猫0狗1猫1狗2猫2
狗0狗1猫0    猫1狗2猫2
狗0狗1猫0狗2猫1    猫2
狗0狗1猫0狗2    猫1猫2
狗0狗1    狗2猫0猫1猫2
狗0狗1狗2    猫0猫1猫2
```

图 1.3 简单测试

AppTest.java

```
package ch1.test;
import ch1.data.*;
public class AppTest {
   public static void main(String [] args) {
      Point [] point = new Point[7];          //创建 7 个点
      for(int i=0;i<point.length;i++) {
         point[i] = new Point();
         point[i].setX(i);
         point[i].setY(10);
      }
      Animal [] left = new Animal[3];         //3 个左边动物
      Animal [] right = new Animal[3];        //3 个右边动物
      for(int i =0;i<left.length;i++ ){       //把左边动物放在点上
        left[i] = new LeftAnimal();
        left[i].setName("猫"+i);
        left[i].setAtPoint(point[i]);
        left[i].setAllCanAtPoint(point);
      }
      for(int i =0;i<right.length;i++ ){      //把右边动物放在点上
        right[i] = new RightAnimal();
        right[i].setName("狗"+i);
        right[i].setAtPoint(point[4+i]);
        right[i].setAllCanAtPoint(point);
      }
      input(point);
      if(right[0].move())                     //让右边的第 1 个动物运动
        input(point);                         //输出各个点上有无动物的信息
      if(left[2].move())
        input(point);
      if(left[1].move())
        input(point);
      if(right[0].move())
        input(point);
```

```
        if(right[1].move())
          input(point);
        if(right[2].move())
          input(point);
        if(left[2].move())
          input(point);
        if(left[1].move())
          input(point);
        if(left[0].move())
          input(point);
        if(right[0].move())
          input(point);
        if(right[1].move())
          input(point);
        if(right[2].move())
          input(point);
        if(left[1].move())
          input(point);
        if(left[0].move())
          input(point);
        if(right[2].move())
          input(point);      //恭喜成功了
    }
    static void input(Point [] point){
      for(int i=0;i<point.length;i++){
         Animal animal=point[i].getAtPointAnimal();
         if(animal!=null)
            System.out.print(animal.getName());
         else
           System.out.print("   ");
      }
      System.out.println();
    }
}
```

1.4 视图设计

设计 GUI 程序除了使用 1.2 节给出的类以外，需要使用 javax.swing 包提供的视图（也称 Java Swing 框架）以及处理视图上触发的界面事件。与 1.3 节中简单的测试相比，GUI 程序可以提供更好的用户界面，完成 1.1 节提出的设计要求。

GUI 部分设计的类如下（主要类的 UML 图如图 1.4 所示）。

- AnimalView 类：其实例为动物提供外观显示。
- GamePanel 类：其实例用于放置动物的外观，并组合负责处理界面事件的监视器。

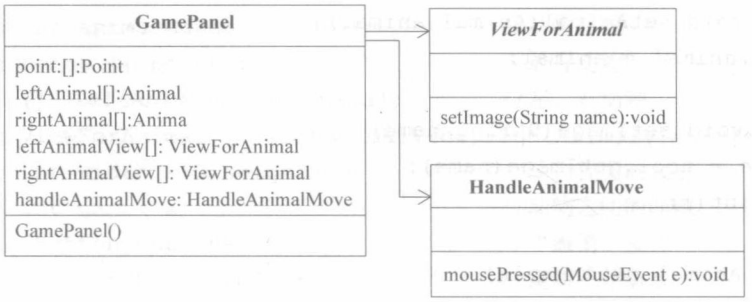

图 1.4　主要类的 UML 图

- HandleAnimalMove 类：其实例是一个监视器，该监视器负责处理 AnimalView 视图，即动物视图上触发的 MouseEvent 事件。当用户在动物视图上按下鼠标时该监视器让动物执行 move()方法，松开鼠标时该监视器检查用户是否成功地将左右动物互换完毕。
- HandleRedo 类：其实例是一个监视器，该监视器负责监视按钮上触发的 ActionEvent 事件，当用户单击按钮触发 ActionEvent 事件时，该监视器负责撤销用户移动动物的操作。
- HandleReStart 类：其实例是一个监视器，该监视器负责监视按钮上触发的 ActionEvent 事件，当用户单击按钮触发 ActionEvent 事件时，该监视器负责将游戏还原成最初始的样子。

❶ 视图相关类

1）AnimalView 类

AnimalView 类是 ViewForAnimal 类的子类，实现了 ViewForAnimal 类中定义的 abstract 方法，其实例通过绘制一幅图像提供动物的外观显示，例如绘制小狗或小猫的图像（如图 1.5 所示）。

图 1.5　AnimalView 类的两个实例

AnimalView.java

```
package ch1.view;
import java.awt.*;
import ch1.data.*;
public class AnimalView extends ViewForAnimal{
   Animal animal;              //要绘制图像的动物
   Image image;                //给动物绘制的图像
   Toolkit tool;               //负责绘制图像的 Toolkit 对象
   public AnimalView() {
      tool = getToolkit();
   }
```

```
    public void setAnimal(Animal animal){
        this.animal = animal;
    }
    public void setImage(String name){
        image = tool.getImage(name);
        repaint();
    }
    public Animal getAnimal() {
        return animal;
    }
    public void setAnimalViewLocation(int x,int y){
        setLocation(x,y);
    }
    public void setAnimalViewSize(int w,int h){
        setSize(w,h);
    }
    public void paintComponent(Graphics g){ //绘制图像的方法（自动执行）
        super.paintComponent(g);
        int w=getBounds().width;
        int h=getBounds().height;
        g.drawImage(image,0,0,w,h,this);
    }
}
```

2）GamePanel 类

GamePanel 类是 javax.swing.JPanel 类的子类，GamePanel 类的实例将 AnimalView 类的实例（动物视图）放置其中（如图 1.6 所示）。

图 1.6　GamePanel 类的实例

GamePanel.java

```
package ch1.view;
import javax.swing.*;
import java.awt.*;
import ch1.data.Animal;
import ch1.data.Point;
import ch1.data.ViewForAnimal;
import ch1.data.LeftAnimal;
import ch1.data.RightAnimal;
import java.util.*;
import java.awt.geom.*;
public class GamePanel extends JPanel {
```