普通高等教育“十三五”规划教材（软件工程专业）

Java程序设计实训教程

JAVA CHENGXU SHEJI SHIXUN JIAOCHENG

主　编　宁淑荣　杨国兴　副主编　金忠伟　李田英

Java

- 精编实例程序——着重培养面向对象的设计思想。
- 实践教学指导——跟随实例步骤轻松掌握开发技能。
- 培养扎实基础——内容由浅入深，循序渐进地引导教学。

中国水利水电出版社
www.waterpub.com.cn

普通高等教育“十三五”规划教材（软件工程专业）

Java 程序设计实训教程

主　编　宁淑荣　杨国兴

副主编　金忠伟　李田英

中国水利水电出版社
www.waterpub.com.cn

·北京·

内 容 提 要

本书精心设计了 4 个实训：扫雷游戏、网络五子棋、棋谱的保存与回放、学生成绩管理系统，介绍 Java 在应用软件开发中用到的主要技术，并体现面向对象的设计思想。对于 Java 中的输入输出、数据库、异常处理、网络编程、界面编程等都有比较深入的训练。

本书可作为计算机类专业 Java 实训、Java 课程设计等课程的教材，也可作为学生毕业设计以及 Java 程序设计爱好者的参考书。

本书配有电子教案和源代码，读者可以到中国水利水电出版社网站和万水书苑上免费下载，网址为 http://www.waterpub.com.cn/softdown/和 http://www.wsbookshow.com。

图书在版编目（CIP）数据

Java程序设计实训教程 / 宁淑荣，杨国兴主编. -- 北京 : 中国水利水电出版社，2018.1
普通高等教育“十三五”规划教材. 软件工程专业
ISBN 978-7-5170-6121-2

Ⅰ. ①J… Ⅱ. ①宁… ②杨… Ⅲ. ①JAVA语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2017)第302641号

策划编辑：周益丹　　责任编辑：封　裕　　封面设计：李　佳

书　名	普通高等教育“十三五”规划教材（软件工程专业） Java 程序设计实训教程 Java CHENGXU SHEJI SHIXUN JIAOCHENG
作　者	主　编　宁淑荣　杨国兴 副主编　金忠伟　李田英
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（营销中心）、82562819（万水）
经　售	全国各地新华书店和相关出版物销售网点
排　版	北京万水电子信息有限公司
印　刷	三河市铭浩彩色印装有限公司
规　格	184mm×260mm　16 开本　11.25 印张　271 千字
版　次	2018 年 1 月第 1 版　2018 年 1 月第 1 次印刷
印　数	0001—3000 册
定　价	26.00 元

凡购买我社图书，如有缺页、倒页、脱页的，本社营销中心负责调换

前　言

Java 是目前使用最广泛的语言之一。对于软件开发人员来说，掌握 Java 语言基础并拥有使用 Java 进行软件开发的能力是非常重要的，因此大多数与计算机相关的专业都开设了 Java 程序设计课程。

Java 程序设计是一门实践性很强的课程（任何一种计算机语言课程皆是），仅仅掌握 Java 的基本语法知识，与能利用 Java 进行软件开发还有很大的差距。掌握 Java 基本知识后，应该通过大量的编程实践来逐步提高利用 Java 进行软件开发的能力。本书精心设计了 4 个实训，详细介绍具体的开发过程，读者可以跟随书中介绍的步骤轻松完成实训程序的开发。

本书由 4 个实训组成：扫雷游戏、网络五子棋、棋谱的保存与回放、学生成绩管理系统，涉及的主要知识有异常处理、输入输出流、数据库编程、多线程和网络编程等。

本书中的所有程序都由作者亲自编写，并在 JDK1.6 环境下调试通过，实例中用到的数据库是 MySQL 数据库。

为了方便教师教学和学生学习，本书提供用 PowerPoint 制作的电子教案，教师可根据具体情况进行必要的修改后使用。

本书由宁淑荣、杨国兴任主编，金忠伟、李田英任副主编，其中实训 1 至实训 3 由宁淑荣编写，实训 4 由杨国兴、金忠伟和李田英共同编写。

本书的编写和出版得到了“北京联合大学规划教材建设项目”资助，特此感谢。

由于作者水平有限，书中不妥之处在所难免，恳请专家与读者批评指正。

编　者

2017 年 10 月

目　　录

实训 1　扫雷游戏

使用 Java 语言编写一个类似 Windows 系统中的扫雷游戏，游戏界面由若干行、若干列的小方块组成，在这些小方块中随机布置着若干个雷，游戏的任务就是将其中的雷找到，将不是雷的小方块翻开，若将所有不是雷的小方块成功翻开则游戏成功，若翻开是雷的小方块则游戏失败。运行界面如图 1.1 所示。

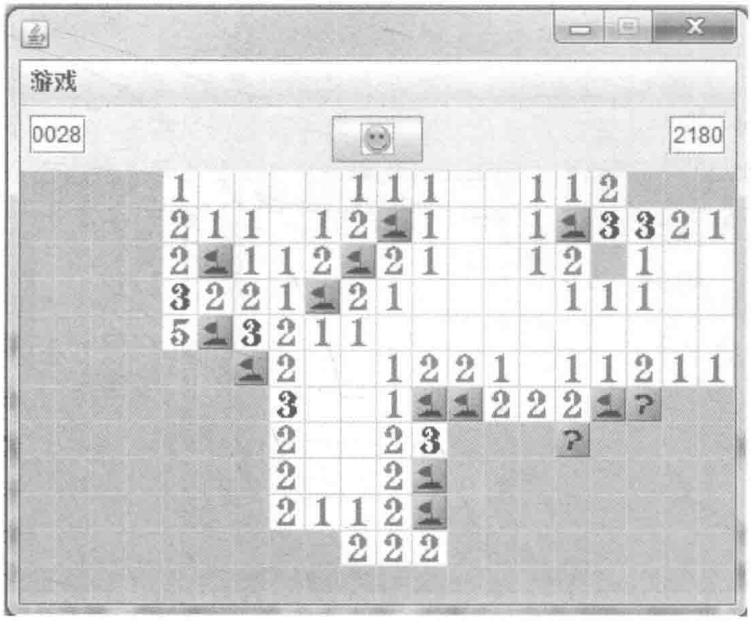

图 1.1　扫雷游戏界面

游戏窗口包含菜单“游戏”，可通过菜单选择游戏的难度级别；菜单的下一行包括三部分，单击中间的按钮可以重新开始游戏，左侧的文本框显示剩余的雷数（没有标记出来的雷数）；右侧的文本框显示游戏已经使用的时间（秒）。

本实训首先对扫雷游戏进行分析设计，然后逐步实现扫雷游戏的所有功能。

1.1　系统设计

1.1.1　需求分析

1. 随机布雷

游戏开始前，在雷区随机布雷，也就是将若干个雷随机分布在雷区的小方块下面。

2. 实现扫雷功能

在小方块上单击鼠标左键，如果该小方块下面是雷，则游戏失败，并将所有雷显示出来；如果该小方块下面不是雷，则将其翻开，翻开后显示的数字是其周围的雷数，每个小方块周围

有 8 个相邻的小方块，因此显示的数字应在 0～8 之间，如果该小方块周围的雷数是 0，则自动将其周围的小方块也翻开。

3. 标记小方块

如果游戏者能够确定某个小方块是雷，可以单击鼠标右键，将其标记为雷（标记为小红旗），在标记为小红旗的小方块上再次单击鼠标右键可将其标记为问号，表示怀疑该小方块是雷，在标记为问号的小方块上再次单击鼠标右键则恢复为原始状态。

4. 选择游戏的难度

扫雷程序分为三个级别：初级、中级和高级，级别越高雷区越大而且雷越多。通过菜单可以选择游戏的级别，如图 1.2 所示；还可以通过“自定义”菜单项自由设置雷区的行数、列数和雷数。

图 1.2 扫雷“游戏”菜单和“用户自定义”对话框

5. 显示剩余雷数和已经使用的时间

在窗口上方两侧的两个文本框中分别显示未标记的雷数和游戏已经使用的时间。

6. 最好成绩纪录

将扫雷成功的最少时间（每个级别分别记录）记录在文件中，每当扫雷成功后，将游戏用时与原来的纪录比较，如果当前使用的时间较少，则保存新的时间作为纪录。在保存成绩之前，要通过对话框输入游戏者的名字，如图 1.3 所示。

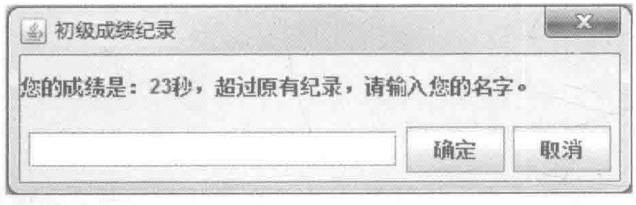

图 1.3 输入游戏者名字

另外，选择菜单项“排行榜”，可以显示各级别的最好成绩和创造该成绩纪录的游戏者名字，如图 1.4 所示。

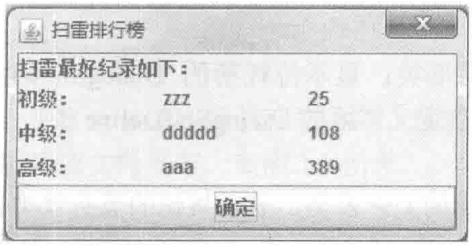

图 1.4　扫雷排行榜

7. 重新开始

在任何情况下，只要单击窗口上方中间的“重新开始”按钮，都可以重新开始扫雷游戏。

8. 加快扫雷进程

如果某个小方块周围的雷都已经标记出来（例如周围雷数是 3，在其相邻的 8 个小方块中已经有 3 个标记为雷），则在该小方块上单击鼠标右键，会将其相邻的没有标记为雷的小方块翻开。当然如果标记错误，在翻开的过程中就会踩到雷而导致扫雷失败。另外，在翻开小方块的过程中，如果某个小方块周围的雷数是 0，则也将其周围尚未翻开的小方块翻开，从而加快扫雷进程。

9. 附加功能

在扫雷的基本功能之外，还可以实现一些附加功能，如扫雷成功或失败时加入不同的声音；在一次单击翻开多个小方块时，也可以加入表示翻开的声音。

1.1.2　类的设计

根据以上的功能分析，设计出系统中需要的类。

1. MineFrame 类

MineFrame 类实现游戏的主窗口，包括菜单、窗口上方的面板（显示计时、雷数和重新开始的按钮），以及游戏的主要部分（雷区）。

2. MinePanel 类

MinePanel 类代表雷区，实现扫雷的主要功能，主要属性包括 Block 类型的二维对象数组。

3. Block 类

Block 类是雷区的小方块类，每个小方块有自己的类型、状态等属性，还有翻开、画出方块等方法。

4. BlockType 类和 BlockState 类

BlockType 类定义一组常量，表示小方块的类型（数值 0～8 分别表示小方块周围的雷数，数值 9 表示该小方块下面就是雷）。BlockState 类定义一组常量，表示小方块的状态（0～4 分别表示原始状态、翻开状态、标记为雷状态、标记为问号状态和雷爆炸状态）。

5. Record 类和 Grade 类

Record 类用于记录最好成绩，包括最好成绩创造者的名字和所用时间。Grade 类定义一组常量，表示游戏的难度级别（0、1、2、3 分别表示初级、中级、高级和自定义）。

6. RecordDao 类

RecordDao 类用于将 Record 对象输出到文件或从文件中读取 Record 对象。

7. 对话框类

扫雷游戏包括 3 个对话框类：显示排行榜的 DialogShowRecord 类、用户输入姓名的 DialogRecordName 类、用户自定义雷区的 DialogSelfDefine 类。

8. 监听器类

监听器类主要有监听菜单的监听器类、监听按钮的监听器类、监听鼠标的监听器类。

将扫雷游戏中几个主要类之间的关系用图 1.5 表示，其中 MineFrame、MinePanel、Block、Record 是本系统设计的主要类，JFrame、JPanel、Thread 是 JDK 提供的类，Serializable 是 JDK 提供的接口。

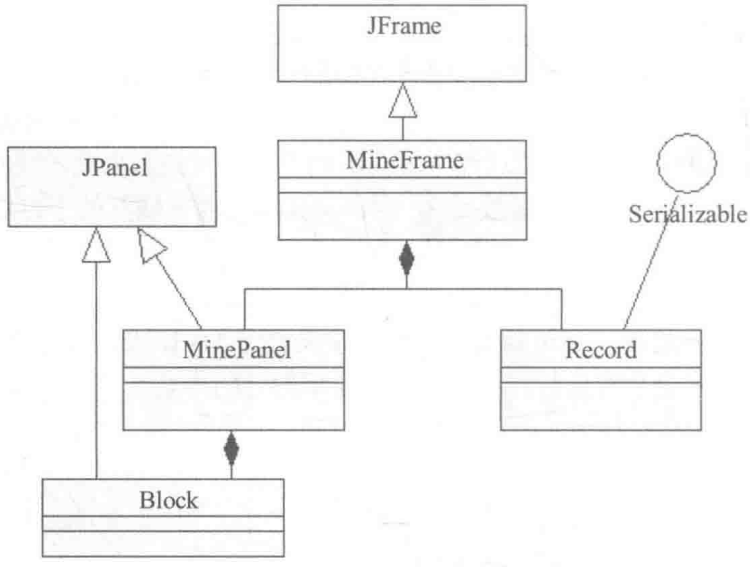

图 1.5 扫雷游戏类图

其他一些实现辅助功能的类并没有在图中绘出，在后面的系统实现中再详细介绍。

1.2 创建主窗口

扫雷游戏主窗口由菜单、窗口上方的计时区，以及中心部分的雷区组成，如图 1.6 所示。本节完成菜单和计时区的设计，雷区部分的设计将在 1.3 节实现。

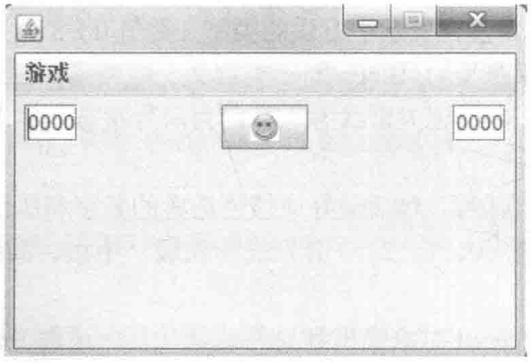

图 1.6 扫雷游戏主窗口

1.2.1　创建项目

启动 Eclipse 后，创建一个名为 Saolei 的 Java Project，然后在项目文件夹下创建子文件夹 image，将所需要的图标复制到该文件夹下，如图 1.7 所示。

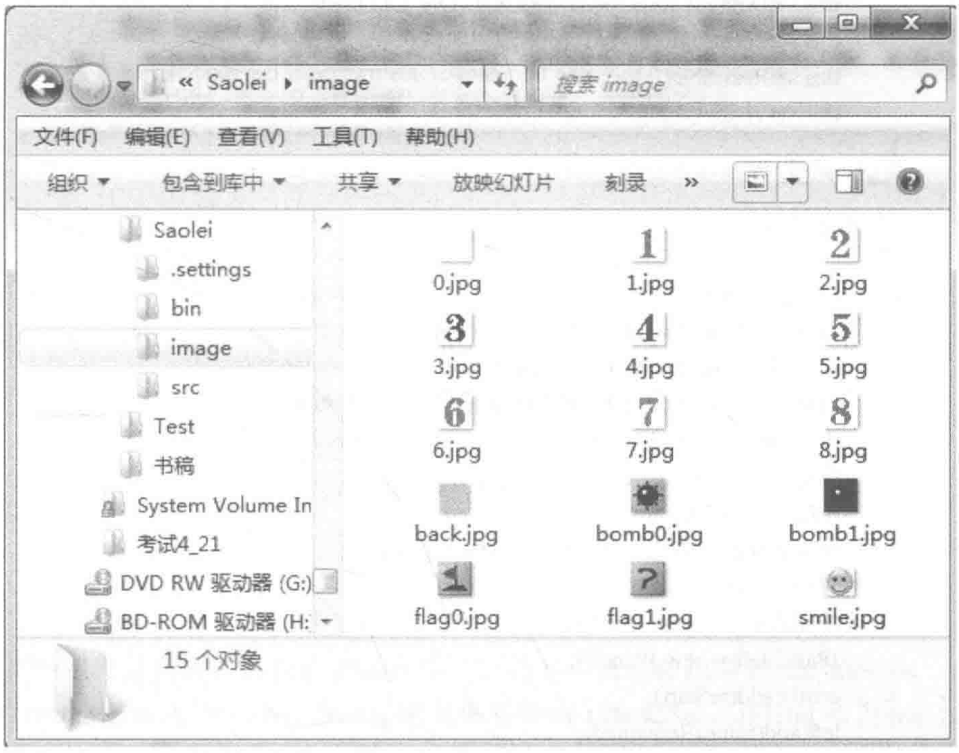

图 1.7　准备程序需要的图标

1.2.2　MineFrame 类

在 Saolei 项目中创建 MineFrame 类（从 JFrame 类继承），在类中添加菜单、按钮、文本框等组件，在构造方法中创建这些组件对象，并将这些组件组织到窗口中，代码如下：

```
/**
* 扫雷游戏主窗口类
 */
public class MineFrame extends JFrame {
    JMenuBar menuBar;
    JMenu menu;
    JMenuItem[] menuItems;
    String[] menuItemNames = {"初级","中级","高级","自定义","排行榜","退出"};
    JTextField minesRemained;          //显示剩余雷数的文本框
    JButton reStart;                   //“重新开始”按钮
    JTextField timeUsed;               //显示游戏使用时间的文本框
    Icon face;                         //按钮上的图标
    JPanel upPanel;                    //计时区域
    public MineFrame(){
```

```
        createMenu();
        createUpPanel();
        Container c =this.getContentPane();
        c.add(upPanel, BorderLayout.NORTH);
        this.setSize(300,200);
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
        this.setLocationRelativeTo(null);
        this.setResizable(false);
        this.setVisible(true);
    }
    /**
     * 创建计时区
     */
    private void createUpPanel(){
        minesRemained = new JTextField("0000");
        minesRemained.setEditable(false);
        timeUsed = new JTextField("0000");
        timeUsed.setEditable(false);
        face = new ImageIcon("image/smile.jpg");
        reStart = new JButton(face);
        reStart.addActionListener(new ButtonMonitor());
        JPanel center = new JPanel();
        JPanel right = new JPanel();
        JPanel left = new JPanel();
        center.add(reStart);
        left.add(minesRemained);
        right.add(timeUsed);
        upPanel = new JPanel(new BorderLayout());
        upPanel.add(left,BorderLayout.WEST);
        upPanel.add(center,BorderLayout.CENTER);
        upPanel.add(right,BorderLayout.EAST);
    }
    /**
     * 创建菜单并注册监听器
     */
    private void createMenu(){
        menuBar = new JMenuBar();
        menu = new JMenu("游戏");
        menuItems = new JMenuItem[menuItemNames.length];
        for(int i =0; i<menuItemNames.length; i++){
            menuItems[i] = new JMenuItem(menuItemNames[i]);
            menu.add(menuItems[i]);
        }
        menuBar.add(menu);
        this.setJMenuBar(menuBar);
        MenuMonitor mm = new MenuMonitor();
        for(int i=0; i<menuItems.length; i++){
```

```
                menuItems[i].addActionListener(mm);
            }
        }
        /**
         * 按钮监听器类
         */
        class ButtonMonitor implements ActionListener{
            public void actionPerformed(ActionEvent e) {
            }
        }
        /**
         * 菜单监听器类
         */
        class MenuMonitor implements ActionListener{
            public void actionPerformed(ActionEvent e) {
                JMenuItem mi = (JMenuItem) e.getSource();
                if(mi.equals(menuItems[5])){
                    System.exit(0);
                }
            }
        }
    }
```

方法 createUpPanel()创建窗口上方的计时区 upPanel。首先创建两个文本框和一个按钮，然后将这 3 个组件分别放在各自的 JPanel 中，再将 3 个 Jpanel 组装到 upPanel 中。

方法 createMenu()创建菜单。Swing 的菜单系统由 JMenuBar、JMenu 和 JMenuItem 组成，扫雷游戏的菜单有 6 个菜单项（由数组 menuItemNames 确定）。首先创建这 6 个菜单项，并将菜单项添加到菜单 JMenu 中，然后将菜单添加到菜单条 JMenuBar 中，再调用 setJMenuBar()方法将菜单条添加到主窗口中，最后为所有的菜单项注册监听器。

内部类 ButtonMonitor 是按钮监听器，目前其 actionPerformed()方法没有代码，具体功能在后面实现。

内部类 MenuMonitor 是菜单监听器，目前只实现了“退出”菜单项的功能，其他菜单项的功能在后面实现。

在构造方法中调用 createMenu()方法和 createUpPanel()方法，创建菜单和计时区，设置窗口的关闭功能。方法 setLocationRelativeTo(null)的作用是设置窗口的相对位置，参数 null 表示相对于屏幕居中。

1.2.3　主程序类 Saolei

添加 Saolei 类，代码如下：

```
public class Saolei {
    public static void main(String[] args) {
        MineFrame mineFrame = new MineFrame();
    }
}
```

在 main()方法中创建一个 MineFrame 对象，显示主窗口。运行这个程序的结果如图 1.6 所示。此时选择菜单项“退出”或者单击窗口右上角的关闭按钮，都能够退出程序。

1.3 MinePanel 类和 Block 类

MinePanel 类代表雷区，由于雷区是由若干行、若干列小方块组成的，因此要首先设计 Block 类。下面首先添加几个辅助类，然后再介绍 Block 类和 MinePanel 类。MinePanel 类设计好之后，扫雷的主窗口就全部实现了，运行后的界面（游戏难度初级，雷区大小是 10 行、10 列）如图 1.8 所示。

图 1.8 扫雷游戏界面

1.3.1 几个辅助类

1. BlockType 类

BlockType 类中定义一组常量，表示小方块的类型，数值 0～8 分别表示与该小方块相邻的雷数，数值 9 表示该小方块下面就是雷。由于每个小方块有 8 个相邻的小方块，因此相邻的雷数的可能值是 0～8，代码如下：

```
public class BlockType {
    static final int ZERO = 0;   //周围的雷数
    static final int ONE = 1;
    static final int TWO = 2;
    static final int THREE = 3;
    static final int FOUR = 4;
    static final int FIVE = 5;
    static final int SIX = 6;
    static final int SEVEN = 7;
    static final int EIHHT = 8;
    static final int MINE = 9;   //是雷
}
```

2. BlockState 类

BlockState 类中定义一组常量，表示小方块的状态，扫雷游戏中的小方块共有 5 种状态，分别是原始状态、翻开状态、标记为雷状态、标记为问号状态、雷爆炸状态，代码如下：

```
Public class blockstate {
    Static final int ORIGINAL = 0;              //初始状态
    Static final int OPEN = 1;                  //翻开状态
    Static final int MINE_FLAG = 2;             //标记为雷状态
    Static final int QUESTION_FLAG = 3;         //标记为问号状态
    Static final int EXPLODED = 4;              //爆炸状态
}
```

3. Grade 类

Grade 类中定义一组常量，表示游戏的难度级别，扫雷游戏中的难度分为 4 个级别，分别是初级、中级、高级和自定义，代码如下：

```
public class Grade {
    public   static final int LOWER = 1;          //初级
    public   static final int MEDIAL = 2;         //中级
    public   static final int HIGHER = 3;         //高级
    public   static final int SELF_DEFINE = 4;    //自定义
}
```

BlockType 类和 BlockState 类是本节中用到的两个类，Grade 类在后面要用到。

1.3.2　Block 类

创建小方块类 Block，该类从 JPanel 类继承，由于小方块是不能单独存在的，一定是某个雷区中的小方块，因此在 Block 类中定义一个雷区类 MinePanel 类型的属性，其他属性包括该小方块在雷区中的行列号、小方块的宽和高、小方块的类型和状态等，代码如下：

```
public class Block extends JPanel {
    private MinePanel minePanel;
    private int row;                    //在雷区中的行号
    private int col;                    //在雷区中的列号
    public final int WIDTH = 19;        //方块的宽度
    public final int HEIGHT = 19;       //方块的高度
    private int type;                   //0,1,2,3,4,5,6,7,8,9（雷）
    private int state;                  //0 原始状态，1 翻开，2 标记为雷，3 标记为问号
    public static Toolkit tk;
    public static final Image[] numberImage;      //0～8
    public static final Image[] flagImage ;       //0 标记为雷，1 标记为问号
    public static final Image[] bombImage;        //0 未爆炸，1 已爆炸
    public static final Image backImage;          //未翻开时的背面
    static{ //静态代码段装载图标资源
        tk = Toolkit.getDefaultToolkit();
        numberImage = new Image[9];
        flagImage = new Image[2];
        bombImage = new Image[2];
        for(int i=0; i<numberImage.length; i++){
```

```
            String fileName = "image/"+i+".jpg";
            numberImage[i] = tk.getImage(fileName);
        }
        for(int i=0; i<flagImage.length; i++){
            String fileName = "image/flag"+i+".jpg";
            flagImage[i] = tk.getImage(fileName);
        }
        for(int i=0; i<bombImage.length; i++){
            String fileName = "image/bomb"+i+".jpg";
            bombImage[i] = tk.getImage(fileName);
        }
        backImage = tk.getImage("Image/back.jpg");
    }
    public Block(){

    }
    public Block(MinePanel minePanel,int row, int col, int type, int state) {
        super();
        this.minePanel = minePanel;
        this.row = row;
        this.col = col;
        this.type = type;
        this.state = state;
    }
    public int getType() {
        return type;
    }
    public void setType(int type) {
        this.type = type;
    }
    public int getState() {
        return state;
    }
    public void setState(int state) {
        this.state = state;
    }
    /**
     * 翻开小方块
     * @return  true：翻开成功；false：翻开失败
     */
    public boolean open(){
        if(type!=BlockType.MINE){
            state = BlockState.OPEN;
            draw(minePanel.getGraphics());
            return true;
        }
```

```
            else{
                state = BlockState.EXPLODED;
                draw(minePanel.getGraphics());
                return false;
            }
        }
        /**
         * 显示小方块
         * @param g   来源于 MinePanel
         */
    public void draw(Graphics g){
        int x = col*minePanel.GRID_WIDTH;
        int y = row*minePanel.GRID_HEIGHT;
            switch(state){
            case BlockState.ORIGINAL:
                g.drawImage(backImage,x,y, WIDTH,HEIGHT,minePanel);
                break;
            case BlockState.MINE_FLAG:
                g.drawImage(flagImage[0],x, y, WIDTH,HEIGHT,minePanel);
                break;
            case BlockState.QUESTION_FLAG:
                g.drawImage(flagImage[1],x, y, WIDTH,HEIGHT,minePanel);
                break;
            case BlockState.OPEN:
                if(type==BlockType.MINE){
                    g.drawImage(bombImage[0],x, y, WIDTH,HEIGHT,minePanel);
                }
                else{
                    g.drawImage(numberImage[type],x, y, WIDTH,HEIGHT,minePanel);
                }
                break;
            case BlockState.EXPLODED:
                if(type==BlockType.MINE){
                    g.drawImage(bombImage[1],x, y, WIDTH,HEIGHT,minePanel);
                }
                break;
            }
        }
    }
```

静态代码块用于装载图标资源，除了构造方法和一组 get、set 方法外，Block 只有 draw()和 open()两个方法。

draw()方法是将小方块在雷区中显示出来，首先获取小方块的左上角坐标，然后根据不同的状态显示不同的图标。如果是原始状态，则显示小方块的背面；如果是标记为雷的状态，则显示小旗图标；如果是怀疑为雷的状态，则显示问号；如果是翻开状态，则显示具体数字图标；如果是爆炸状态，则显示爆炸图标。