第 5 章　标准化试题训练系统

5.1　设计要求

设计 GUI 界面的标准化试题训练系统。具体要求如下：

① 使用 Microsoft Excel 工作簿存放标准化试题，形成题库。

② 程序每次从题库随机抽取若干道题目形成一张试卷，用户可以依次做试卷上的题目，允许用户向前、向后翻阅试卷上的题目。

③ 用户每次做完一个题目必须确定该题目的答案，否则无效。

④ 有计时功能，比如指定一张试卷限用时 15 分钟，时间一到用户再无法答题，提示用户提交试卷。

⑤ 用户一旦提交试卷，程序将给出试卷的分值。

⑥ 为了达到反复训练的目的，用户提交试卷后可以继续让程序再出一套试卷。

程序运行的参考效果图如图 5.1 所示。

图 5.1　标准化试题训练系统

注意　我们按照 MVC（Model-View-Controller，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 Controller。

5.2　数据模型

根据系统设计要求在数据模型部分设计了 Excel 表，编写了有关的类。

- 创建 Excel 工作簿。

- Problem 类：其实例是一道试题。
- TestPaper 类：其实例是一张试卷。
- GiveTestPaper 接口：封装给出试卷方法。
- RamdomInitTestPaper 类：实现 GiveTestPaper 接口，其实例负责随机从题库抽取题目给出试卷。
- Teacher 接口：封装阅卷方法。
- TeacherOne 类：实现 Teacher 接口，其实例负责阅卷。

数据模型部分涉及的主要类的 UML 图如图 5.2 所示。

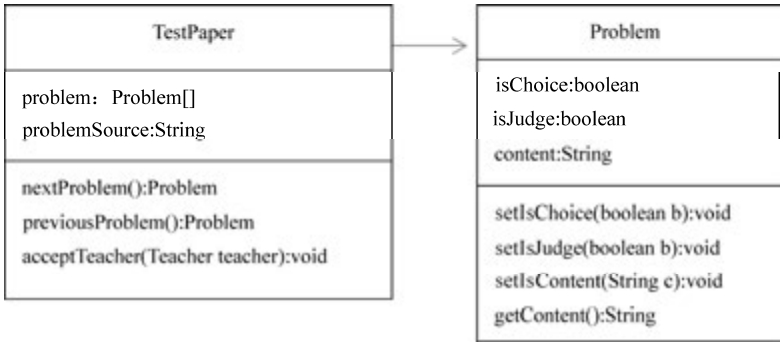

图 5.2　主要类的 UML 图

❶ Excel 工作簿

Excel 工作簿在存储数据方面有着广泛的应用（它不是数据库），其中的 Sheet 表的结构和数据库中的表类似。JDBC 没有提供操作 Excel 工作簿的 API。为了操作 Excel 工作簿，需要额外下载操作 Excel 的 API。

1）下载 Excel API

用户可以搜索 Excel API 得到一个 Excel API 的下载地址，比如：

http://download.csdn.net/download/sparkthink/4168138

然后将下载的 jexcelapi_2_6_6.zip（版本号及名字略有差异，有些下载网址的下载名称是 jxls-2.3.0）解压缩到本地，则根目录下的 jxl.jar 文件就是操作 Excel 所需要的 API 的 JAR 文件（不要解压缩该文件），根目录下的其他文件主要是 Excel API 的帮助文档。将 jxl.jar 复制到 Java 运行环境的扩展中，即将这个 JAR 文件 jxl.jar 存放在 JDK 安装目录的“\jre\lib\ext”中，比如“E:\jdk1.8\jre\lib\ext”。在安装 JDK 时还额外有一个 JRE，比如默认安装在“C:\Program Files (x86)\Java\ jre1.8.0_45”。将 jxl.jar 文件也复制到“C:\Program Files (x86)\Java\ jre1.8.0_45\lib\ext”中。

注意　可以到作者的网盘“http://pan.baidu.com/s/1qYCv0ra”下载 jexcelapi_2_6_6.zip。

2）存储试题的 Sheet 表

在标准化试题训练系统中，按照要求需要使用 Excel 工作簿中的 Sheet 表存储试题，即 Excel 工作簿中的 Sheet 表充当题库的角色。Excel 中 Sheet 表的结构对代码的编写是十分重要

的，比如列的数目以及顺序，因为后续的某些代码依赖于这些结构，即某些代码会和 Sheet 表的结构形成紧耦合关系，因此，Sheet 表的结构一旦更改，必然引起代码的修改。对 Excel 中 Sheet 表的结构要求如下：

Sheet 表一共 7 列（A、B、C、D、E、F、G），各列的取值规则如下（不可再改变其取值规则）。

- A 列：试题内容。例如“这个标志是何含义？”。
- B 列：正确答案。试题的正确答案只可以是 A、B、C、D 字母的组合（不区分大小写），例如 B、ABC、C、D。
- C 列：选择项目。例如“A.沿左侧车道掉头”。
- D 列：选择项目。例如“B.该路口不能掉头”。
- E 列：选择项目。例如“C.选择中间车道掉头”。
- F 列：选择项目。例如“D.多股铁路与道路相交”。
- G 列：题目类型。题目的类型只可以是“p”、“x”、“p#图像文件名字”或“x#图像文件名字”（字母 p、x 不区分大小写），例如 x#hello.jpg、p#java001.jpg。类型 p 表示试题类型是判断题，类型 x 表示试题类型是选择题，类型 p#表示试题类型是判断题并带有图像，x#表示试题类型是选择题并带有图像。
- Excel 中 Sheet 表的第一行不是试题，是试题的说明，说明文字可任意给定。

根据设计要求打开 Microsoft Excel 设计了 Excel 工作簿以及其中的第一张 Sheet 表（在 Excel 工作簿中是 Sheet1 表）——交通理论.xls，Sheet1 表结构如图 5.3 所示。

A	B	C	D	E	F	G
题目内容	正确答案	选择项目	选择项目	选择项目	选择项目	题目类型
遇到这种情况的路口，	B	A. 沿左侧车	B. 该路口不能	C. 选择中间	D. 在路口内	x#x001. jpg
这个标志是何含义？	B	A. 无人看守	B. 有人看守铁	C. 立交式的	D. 多股铁路	x#x002. jpg
拼装的机动车只要认	B	A. 正确	B. 错误			p
在道路与铁路道口遇到	B	A. 正确	B. 错误			p#p001. jpg
机动车发生碰撞时座椅	D	A. 减轻驾乘	B. 保护驾乘人	C. 保护驾乘	D. 保护驾乘	x

图 5.3　Excel 中的 Sheet 表的结构

下载的操作 Excel 表的 API 要求 Excel 工作簿必须是扩展名为.xls 的 Excel 文件，因此在用 Microsoft Excel 保存 Excel 工作簿时需要把保存类型选择为“Excel 97-2003 工作簿（*.xls）”。

3）题库与图像管理

需要建立一个名字是“题库”的文件夹，存放 Excel 工作簿，以及名字是“图像管理”的文件夹，存放所需要的图像文件。文件夹的位置需要和程序在同级目录中。比如，程序中的包名目录是“ch5\gui”，“ch5\gui”目录的上一层目录是 D:\，那么“题库”文件夹和“图像管理”文件夹必须存放在 D:\下，即保持和 ch5 是同级。

为了便于软件的管理以及编写，对于不需要图像的试题，程序统一用默认的图像，该默认图像的名字固定为 havenot.jpg。另外，程序还需要一个名字是 renew.jpg 的图像，当用户重新选择试卷时用该图像友好地提示用户，因此必须将 havenot.jpg 和 renew.jpg 图像保存到“图像管理”文件夹中（图像的外观可自己指定）。havenot.jpg 和 renew.jpg 图像如图 5.4 所示。

图 5.4　havenot.jpg 和 renew.jpg

❷ 试题与试卷

1）Problem 类

Sheet 表中的一行数据是一道试题。用 Problem 类来封装 Sheet 表的基本结构，即试题，这对于后续代码的设计是非常有利的。下列 Problem 类封装表结构，其实例是一道试题。

Problem.java

```
package ch5.data;
public class Problem {
    boolean isChoice;                                   //是否为选择题
    boolean isJudge;                                    //是否为判断题
    String  content;                                    //题目内容
    String  giveChoiceA,giveChoiceB,giveChoiceC,giveChoiceD;//提供选择
    String  imageName;                                  //题目所带的图像文件的名字
    String  correctAnswer="QWEQ@#$@!@#1QWEQ";     //题目的正确答案
    //用户回答的初始答案和 correctAnswer 不同，防止出题人忘记给正确答案
    String  userAnswer ="" ;                            //初始值必须是不含任何字符的串
    public boolean getIsChoice() {
       return isChoice;
    }
    public void setIsChoice(boolean b) {
       isChoice = b;
    }
    public boolean getIsJudge() {
       return isJudge;
    }
    public void setIsJudge(boolean b) {
       isJudge = b;
    }
    public void setContent(String c) {
       content = c;
    }
    public String getContent() {
       return content;
    }
    public void setCorrectAnswer(String a) {
       correctAnswer = a;
    }
    public String getCorrectAnswer() {
```

```
        return correctAnswer;
    }
    public void setUserAnswer(String u) {
        userAnswer = u;
    }
    public String getUserAnswer() {
        return userAnswer;
    }
    public void setGiveChoiceA(String a) {
        giveChoiceA = a;
    }
    public String getGiveChoiceA() {
        return giveChoiceA;
    }
    public void setGiveChoiceB(String b) {
        giveChoiceB = b;
    }
    public String getGiveChoiceB() {
        return giveChoiceB;
    }
    public void setGiveChoiceC(String c) {
        giveChoiceC = c;
    }
    public String getGiveChoiceC() {
        return giveChoiceC;
    }
    public void setGiveChoiceD(String d) {
        giveChoiceD = d;
    }
    public String getGiveChoiceD() {
        return giveChoiceD;
    }
    public void setImageName(String c) {
        imageName = c;
    }
    public String getImageName() {
        return imageName;
    }
}
```

2）TestPaper 类

训练时需要从题库获得若干个试题，即用若干道试题组成一张试卷，这里用 TestPaper 类封装试卷，即该类的实例就是一张试卷。

TestPaper.java

```
package ch5.data;
```

```
public class TestPaper {  //试卷
    private Problem [] problem=null;//数组的每个单元存放一道试题（一个 Problem 对象）
    int index = -1;
    String  problemSource ;          //试卷的题库
    public void setProblem(Problem [] problem){
        this.problem = problem;
    }
    public  Problem  getProblem(int i) {
       if(problem == null) {
          return null;
       }
       if(problem.length==0){
          return null;
       }
       if(i>=problem.length||i<0) {
           return null;
       }
       return problem[i];
    }
    public  Problem  nextProblem() {
       index++;
       if(problem == null) {
          return null;
       }
       if(problem.length==0){
         return null;
       }
       if(index==problem.length) {
           index = problem.length-1;  //到最后一个题目停止
       }
       return problem[index];
    }
    public  Problem  previousProblem() {
       index--;
       if(problem == null) {
          return null;
       }
       if(problem.length==0){
         return null;
       }
       if(index<0) {
           index =0;                  //到第一个题目停止
       }
       return problem[index];
    }
    public  Problem [] getAllProblem(){
```

```
        if(problem == null) {
           return null;
        }
        if(problem.length==0){
          return null;
        }
        return problem;
    }
    public int getProlemAmount(){
        if(problem == null) {
           return 0;
        }
        return problem.length;
    }
    public void setProblemSource(String source){
        problemSource = source;
    }
    public String getProblemSource(){
        return problemSource;
    }
    public void acceptTeacher(Teacher teacher) {     //让老师来批卷（访问者模式）
        teacher.giveTestPaparScore(this);            //teacher 批卷
    }
}
```

❸ **阅卷**

试卷本身不能给出自己的分数，需要其他对象访问试卷，根据试卷中的数据给出相应的评判。为了便于系统的后期扩展，这里将评判试卷的方法封装在 Teacher 接口中：

```
public void giveTestPaparScore(TestPaper testPaper)
```

Teacher.java

```
package ch5.data;
public interface Teacher {
    public void giveTestPaparScore(TestPaper testPaper);
}
```

下列 TeacherOne 类实现 Teacher 接口，给出了一种评卷方式（只给出正确率）。

TeacherOne.java

```
package ch5.data;
import javax.swing.*;
public class TeacherOne implements Teacher {
    public void giveTestPaparScore(TestPaper testPaper){
         int correctAmount=0;          //百分比计分
         if(testPaper==null){
           JOptionPane.showMessageDialog
             (null,"没答题","消息对话框",JOptionPane.WARNING_MESSAGE);
```

```
            return;
          }
          Problem p[] = testPaper.getAllProblem();
          if(p==null||p.length==0){
            JOptionPane.showMessageDialog
              (null,"没答题","消息对话框",JOptionPane.WARNING_MESSAGE);
            return;
          }
          for(int i=0;i<p.length;i++){
             boolean b = compare(p[i].getUserAnswer(),p[i].getCorrectAnswer());
             if(b) {
                correctAmount++;
             }
          }
          double result = (double)correctAmount/(double)p.length;
          int r =(int)(result*100);
          String s = "共有:"+p.length+"道题."+
                 "您做对了"+correctAmount+"题,"+"正确率大约"+r+"%";
          JLabel mess = new JLabel(s);
          JOptionPane.showMessageDialog(null,mess,"成绩",JOptionPane.PLAIN_
         MESSAGE );
      }
      private boolean compare(String s,String t) {
          boolean isTrue = true;
          for(int i=0;i<s.length();i++) {
             String temp = ""+s.charAt(i);
             if(!(t.contains(temp)))
               isTrue = false;
          }
          for(int i=0;i<t.length();i++) {
             String temp = ""+t.charAt(i);
             if(!(s.contains(temp)))
               isTrue = false;
          }
          return isTrue;
      }
  }
```

❹ **抽取试题**

为了便于系统的后期扩展，这里将抽取试题形成试卷的方法封装在 GiveTestPaper 接口中：

```
public TestPaper getTestPaper(String excelFileName,int amount)
```

GiveTestPaper.java

```
package ch5.data;
```

```
public interface GiveTestPaper {
   public TestPaper getTestPaper(String excelFileName,int amount);
}
```

下列 RamdomInitTestPaper 类实现 GiveTestPaper 接口，其实例负责给出试卷（随机从题库中抽取若干道试题形成一张试卷）。

RamdomInitTestPaper.java

```
package ch5.data;
import java.io.*;
import jxl.*;
import java.util.*;
import javax.swing.JOptionPane;
public class RamdomInitTestPaper implements GiveTestPaper {
                                                    //将试题放入试卷（出卷）
   TestPaper testPaper ;                            //试卷
   File fileExcel;
   Problem [] problem;
               //组成试卷的一套题（problem 的单元存放一道试题，即一个 Problem 对象）
   InputStream in = null;
   Workbook wb = null;  //封装 Excel，Workbook 是 jxl 包中的类
   Sheet sheet = null;  //封装 Excel 中的 sheet，Sheet 是 jxl 包中的类
   LinkedList<Integer> list;                        //随机抽取试题时用
   public RamdomInitTestPaper() {
      testPaper = new TestPaper();
      list = new LinkedList<Integer>();
   }
   public TestPaper getTestPaper(String excelFileName,int amount) {
      boolean b =setExcel(excelFileName);    //设置用户存放试题的电子表格
      if(b) {
        try {
           randomGiveProblem(amount);
                       //随机给出 amount 道试题，见类后面的 random GiveProblem 方法
        }
        catch(ArrayIndexOutOfBoundsException e){
           System.out.println("试题必须有类型，请检查题库");
           System.exit(0);
        }
        testPaper.setProblem(problem);          //试卷上设置的一套试题是 problem
        return testPaper;                       //返回试卷
      }
      else {
        JOptionPane.showMessageDialog
        (null,"没有预备题库","消息对话框",JOptionPane.WARNING_MESSAGE);
        return null;
      }
   }
```

```
private boolean setExcel(String excelFileName){
    boolean b =true;
    try {
       fileExcel =new File(excelFileName);
       in =new FileInputStream(fileExcel);
       testPaper.setProblemSource(fileExcel.getAbsolutePath());
                                    //试卷设置题库来源
    }
    catch(IOException exp){
       JOptionPane.showMessageDialog
    (null,"没有预备题库 Excel","消息对话框",JOptionPane.WARNING_MESSAGE);
        b = false;
    }
    try {
       wb=Workbook.getWorkbook(in);
       in.close();
    }
    catch(Exception exp){
       b = false;
    }
    return b;
}
private void randomGiveProblem(int amount) {
                                    //随机给出 amount 道试题放入 problem 数组中
    list.clear();
    if(wb==null) {
     JOptionPane.showMessageDialog
    (null,"没有预备题库 Excel","消息对话框",JOptionPane.WARNING_MESSAGE);
       return ;
    }
    sheet = wb.getSheet(0);       //得到 Excel 中的第一个 sheet（索引从 0 开始）
    int rowsAmount = sheet.getRows();     //得到 sheet 的总行数
    //注意原始 Excel 表中 sheet 中的第 0 行不是试题，是用户输入的说明
    int realAmount = Math.min(amount,rowsAmount-1);//实际抽取的试题数量
    problem = new Problem[realAmount];    //用于存放试题的数组 problem
    for(int i=0;i<rowsAmount-1;i++){      //将 1～rowsAmount-1 放入链表
       list.add(i+1);
    }
    Random random=new Random();
    for(int i=0;i<problem.length;i++) {
        int m = random.nextInt(list.size());//[0,list.size())中的一个随机数
        int index =list.remove(m);//删除 list 的第 m 个节点，同时得到节点数字
        Cell [] cell = sheet.getRow(index); //返回 sheet 中的第 index 行
        //注意原始 Excel 表中 sheet 中的第 0 行不是试题，是用户输入的说明
        //cell 的第 0 列是试题内容，索引从 0 开始
        problem[i] = new Problem();
```

```
            int number = i+1;
            problem[i].setContent("第"+number+"题."+cell[0].getContents());
               //试题的内容
            problem[i].setCorrectAnswer(cell[1].getContents().trim());//试题的答案
            problem[i].setGiveChoiceA(cell[2].getContents().trim());//试题的A选择
            problem[i].setGiveChoiceB(cell[3].getContents().trim());//试题的B选择
            problem[i].setGiveChoiceC(cell[4].getContents().trim());//试题的C选择
            problem[i].setGiveChoiceD(cell[5].getContents().trim());//试题的D答案
            String typeStr = cell[6].getContents().trim();//试题的类型（判断或选择）
            //因为试题有图像，所以typeStr有4种，即p、p#、x、x#
            if(typeStr.equalsIgnoreCase("p")){
                 problem[i].setIsJudge(true);
                 problem[i].setIsChoice(false);
                 problem[i].setImageName("havenot.jpg");
            }
            if(typeStr.equalsIgnoreCase("x")) {
                 problem[i].setIsJudge(false);
                 problem[i].setIsChoice(true);
                 problem[i].setImageName("havenot.jpg");
            }
            if(typeStr.startsWith("p#")||typeStr.startsWith("P#")) {
                 problem[i].setIsJudge(true);
                 problem[i].setIsChoice(false);
                 String imageName=typeStr.substring(typeStr.indexOf("#")+1);
                 problem[i].setImageName(imageName);
            }
            if(typeStr.startsWith("x#")||typeStr.startsWith("X#")) {
                problem[i].setIsJudge(false);
                problem[i].setIsChoice(true);
                String imageName = typeStr.substring(typeStr.indexOf("#")+1);
                problem[i].setImageName(imageName);
            }
        }
    }
}
```

5.3　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:\ch5\data
```

编译各个源文件，例如：

```
D:\>javac ch5/data/Problem.java
```

也可以如下编译全部源文件：

```
D:\>javac ch5/data/*.java
```

将“题库”的文件夹（存放 Excel 工作簿），以及“图像管理”文件夹存放在包名目录的父目录中（这里需要保存在 D:\中）。

把 5.2 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，测试标准化试题训练，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:\ch5\test
```

编译源文件：

```
D:\>javac ch5/test/AppTest.java
```

运行 AppTest 类（运行效果如图 5.5 所示）：

```
D:\>java ch5.test.AppTest
```

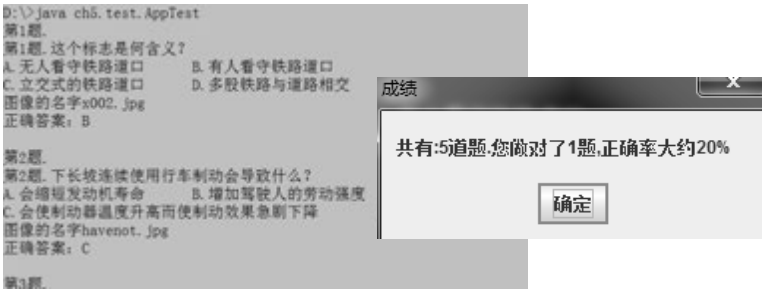

图 5.5　简单测试

AppTest.java

```
package ch5.test;
import ch5.data.*;
public class AppTest {
   public static void main(String []args) {
       GiveTestPaper initTestPaper = new RamdomInitTestPaper();//创建初始试卷对象
       TestPaper testPaper=
       initTestPaper.getTestPaper("题库/交通理论.xls",5); //得到有 5 个题目的试卷
       Problem [] problem = testPaper.getAllProblem();  //得到试卷中的全部试题
       for(int i = 0;i<problem.length;i++ ) {
           int m = i+1;
           System.out.println("第"+m+"题.");
           System.out.println(problem[i].getContent());
           if(problem[i].getIsJudge()){
              inputOne(problem[i]);
           }
           else if(problem[i].getIsChoice()){
```

```
                inputTwo(problem[i]);
            }
            System.out.println();
        }
         //模拟用户答题
        problem[0].setUserAnswer("B");                    //模拟用户给的答案是 B
        problem[1].setUserAnswer("A");
        problem[2].setUserAnswer("C");
        problem[3].setUserAnswer("A");
        problem[0].setUserAnswer("B");
        problem[1].setUserAnswer("D");
        testPaper.acceptTeacher(new TeacherOne());    //让老师批卷
    }
    static void inputOne(Problem  problem){
        System.out.printf("%s\t%s\n",problem.getGiveChoiceA(),problem.getGiveChoiceB());
        System.out.println("图像的名字"+problem.getImageName());
        System.out.println("正确答案："+problem.getCorrectAnswer());
    }
    static void inputTwo(Problem problem){
        System.out.printf("%s\t%s\n",problem.getGiveChoiceA(),problem.getGiveChoiceB());
        System.out.printf("%s\t%s\n",problem.getGiveChoiceC(),problem.getGiveChoiceD());
        System.out.println("图像的名字"+problem.getImageName());
        System.out.println("正确答案："+problem.getCorrectAnswer());
    }
}
```

5.4　视图设计

设计 GUI 程序除了使用 5.2 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 5.3 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 5.1 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 5.6 所示）。

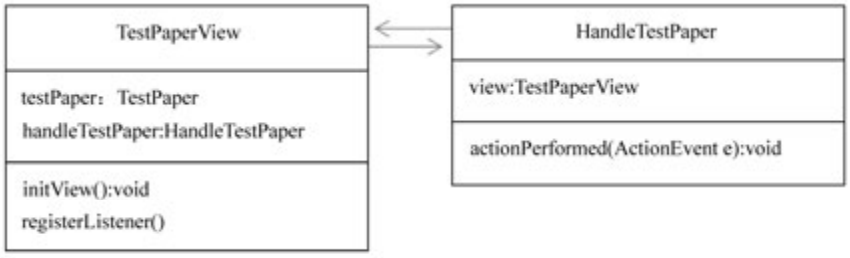

图 5.6　主要类的 UML 图

- TestPaperView 类：其实例为试卷提供视图。
- ImageJPanel 类：其实例为试题的图像提供面板视图。

- ShowImageDialog 类：其实例为试题的图像提供对话框视图。
- IntegrationView 类：其实例将其他视图集成为一个视图。
- HandleTestPaper 类：其实例负责处理 TestPaperView 视图上的界面事件。

❶ 视图相关类

1）TestPaperView

TestPaperView 类是 JPanel 类的子类，其实例提供了试卷的视图，用户可以在该视图看见试卷的内容并进行答题操作（如图 5.7 所示）。该视图使用 javax.swing.Timer 类，根据试卷的时间要求进行计时。

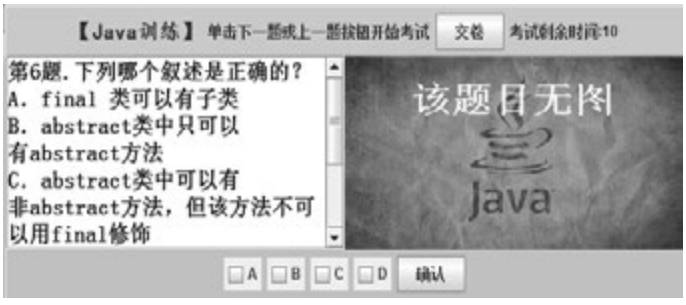

图 5.7　试卷视图

TestPaperView.java

```
package ch5.view;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import ch5.data.*;
public class TestPaperView extends JPanel implements ActionListener{
   TestPaper testPaper;                    //本视图需要显示的试卷
   public Teacher teacher ;                //批卷老师
   public JTextArea showContent;           //显示试题内容
   public ImageJPanel showImage;           //显示试题的图像
   public JCheckBox choiceA,choiceB,choiceC,choiceD;//显示选项内容
   public JButton nextProblem,previousProblem;//选择“上一题”“下一题”的按钮
   public JButton aProblemSubmit;          //确认某道题的回答或选择
   public JButton renewJButton;            //重新开始
   public JButton submit;                  //交卷
   HandleTestPaper handleTestPaper;        //负责处理 testPaper 试卷中的数据
   public int  totalTime = 0;              //考试用时（单位：分）
   public int  usedTime = totalTime;
   public javax.swing.Timer time;          //考试计时器
   public JLabel showUsedTime ;            //显示用时
   JLabel testName ;                       //显示考试名称
   public TestPaperView() {
      time = new Timer(60*1000,this);      //每隔 1 分钟计时一次(触发 ActionEvent),
                                           //本容器作为其监视器
```

```
    initView();
    registerListener();
}
public void setTeacher(Teacher teacher){
    this.teacher = teacher;
}
public void initView() {
    setLayout(new BorderLayout());
    showImage = new ImageJPanel();
    showContent = new JTextArea(12,12);
    showContent.setToolTipText("如果题中有图像，在图上单击鼠标可调整观看");
    showContent.setForeground(Color.blue);
    showContent.setWrapStyleWord(true);
    showContent.setLineWrap(true); //自动回行
    showContent.setFont(new Font("宋体",Font.BOLD,18));
    choiceA = new JCheckBox("A");
    choiceB = new JCheckBox("B");
    choiceC = new JCheckBox("C");
    choiceD = new JCheckBox("D");
    choiceA.setVisible(false);
    choiceB.setVisible(false);
    choiceC.setVisible(false);
    choiceD.setVisible(false);
    nextProblem = new JButton("下一题");
    previousProblem = new JButton("上一题");
    aProblemSubmit = new JButton("确认");
    aProblemSubmit.setVisible(false);
    renewJButton = new JButton("再来一次");
    renewJButton.setVisible(false);
    submit = new JButton("交卷");
    JPanel pNorth = new JPanel();
    pNorth.setBackground(Color.cyan) ;
    showUsedTime = new JLabel();
    testName = new JLabel();
    testName.setFont(new Font("楷体",Font.BOLD,18));
    pNorth.add(testName);
    pNorth.add(new JLabel("单击下一题或上一题按钮开始考试"));
    pNorth.add(submit);
    pNorth.add(renewJButton);
    pNorth.add(showUsedTime);
    add(pNorth,BorderLayout.NORTH);
    JPanel pCenter = new JPanel();
    pCenter.setLayout(new GridLayout(1,2));
    pCenter.add(new JScrollPane(showContent));
    pCenter.add(showImage);
    add(pCenter,BorderLayout.CENTER);
```

```
    JPanel pSouth = new JPanel();
    pSouth.setLayout(new GridLayout(2,1));
    JPanel oneInPSouth = new JPanel();
    JPanel twoInPSouth = new JPanel();
    oneInPSouth.setBackground(Color.green) ;
    oneInPSouth.setBackground(Color.pink) ;
    oneInPSouth.add(choiceA);
    oneInPSouth.add(choiceB);
    oneInPSouth.add(choiceC);
    oneInPSouth.add(choiceD);
    oneInPSouth.add(aProblemSubmit);
    twoInPSouth.add(nextProblem);
    twoInPSouth.add(previousProblem);
    pSouth.add(oneInPSouth);
    pSouth.add(twoInPSouth);
    add(pSouth,BorderLayout.SOUTH);
    validate();
}
public void registerListener(){
    handleTestPaper = new HandleTestPaper();
    nextProblem.addActionListener(handleTestPaper);
    previousProblem.addActionListener(handleTestPaper);
    aProblemSubmit.addActionListener(handleTestPaper);
    submit.addActionListener(handleTestPaper);
    renewJButton.addActionListener(handleTestPaper);
    handleTestPaper.setView(this);
}
public void setTestPaper(TestPaper testPaper) {
    this.testPaper = testPaper;
    handleTestPaper.setTestPaper(testPaper);
}
public void actionPerformed(ActionEvent e){
    showUsedTime.setText("考试剩余时间:"+usedTime);
    if(usedTime == 0){
        time.stop();
        showUsedTime.setText("请交卷");
        nextProblem.setVisible(false);
        previousProblem.setVisible(false);
    }
    usedTime--;
}
public void setTestName(String name){
    testName.setText("【"+name+"】");
}
public void setTotalTime(int n) {
    totalTime = n;
```

```
            usedTime = n;
            showUsedTime.setText("考试剩余时间:"+usedTime);
        }
    }
```

2）ImageJPanel

ImageJPanel 类是 JPanel 类的子类，其实例显示试题中的图像，用户可以在该视图中看见图像（如图 5.8 所示），如果需要，用户可以单击图像，弹出一个可以仔细观察图像的对话框。

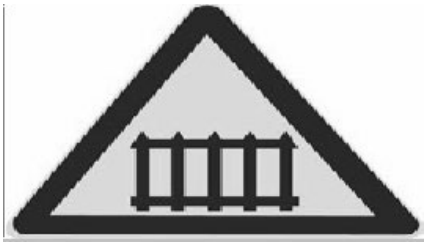

图 5.8　显示图像的面板

ImageJPanel.java

```
package ch5.view;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ImageJPanel extends JPanel implements MouseListener {
    Image image;
    ImageJPanel() {
        setOpaque(false);
        setBorder(null);
        setToolTipText("单击图像单独调整观看");
        addMouseListener(this);
    }
    public void setImage(Image img){
        image = img;
    }
    public void paintComponent(Graphics g ) {
        super.paintComponent(g);
        g.drawImage(image,0,0,getBounds().width,getBounds().height,this);
    }
    public void mousePressed(MouseEvent e) {
        ShowImageDialog showImageDialog = new ShowImageDialog(image);
        showImageDialog.setVisible(true);
    }
    public void mouseReleased(MouseEvent e){}
```

```
    public void mouseEntered(MouseEvent e)  {}
    public void mouseExited(MouseEvent e){}
    public void mouseClicked(MouseEvent e) {}
}
```

3）ShowImageDialog

ShowImageDialog 类是 JDialog 类的子类，其实例用对话框视图显示试题中的图像，以方便用户仔细地观察图像（如图 5.9 所示）。

图 5.9　显示图像对话框

ShowImageDialog.java

```
package ch5.view;
import java.awt.*;
import javax.swing.*;
public class ShowImageDialog extends JDialog {
    Image img;
    ShowImageDialog(Image img) { //构造方法
       setTitle("显示图像对话框");
       this.img = img;
       setSize(500,470);
       GiveImage image = new GiveImage();
       add(image);
       setModal(true);
       setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }
    class GiveImage extends JPanel  {  //内部类，专门为该对话框提供图片
       public void paintComponent(Graphics g ) {
          super.paintComponent(g);
          g.drawImage(img,0,0,getBounds().width,getBounds().height,this);
       }
    }
}
```

4）IntegrationView

IntegrationView 类是 JFrame 类的子类，其实例使用 JTabbedPane 将各个视图集成到当前

IntegrationView 窗体中（如图 5.10 所示）。

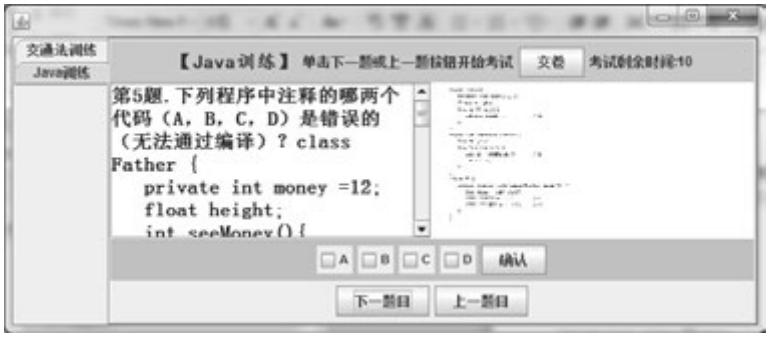

图 5.10　集成视图的窗体

IntegrationView.java

```
package ch5.view;
import java.awt.BorderLayout;
import javax.swing.JFrame;
import javax.swing.JTabbedPane;
public class IntegrationView extends JFrame{
    JTabbedPane tabbedPane; //用选项卡集成 TestPaperView 视图
    public IntegrationView(){
        tabbedPane= new JTabbedPane(JTabbedPane.LEFT);//卡在左侧
        tabbedPane.validate();
        add(tabbedPane,BorderLayout.CENTER);
        setBounds(100,100,1200,560);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setVisible(true);
    }
    public void addTestPaperView(String cardName,TestPaperView view){
       tabbedPane.add(cardName,view);
       validate();
    }
}
```

❷ 事件监视器

事件监视器负责处理视图上触发的用户界面事件，以便完成相应的任务。

HandleTestPaper 类的实例负责监视 TestPaperView 视图上的事件，比如负责停止计时器、负责处理用户提交的答案、负责让老师批卷等。

HandleTestPaper 类实现了 ActionListener 接口，其实例负责处理用户单击按钮触发的 ActionEvent 事件。当用户单击“下一题”或“上一题”按钮时让视图显示相应的试题，当用户单击确认某题目答案的按钮时，将用户的答案保存到相应的 Problem 对象中，当用户单击“交卷”按钮时，让 Teacher 对象阅卷。

HandleTestPaper.java

```
package ch5.view;
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import ch5.data.*;
public class  HandleTestPaper implements ActionListener{
    TestPaperView  view;
    TestPaper   testPaper;        //需要处理的试卷
    Problem problem;              //当前的题目
    Toolkit tool;                 //处理图像
    public HandleTestPaper(){
        tool = Toolkit.getDefaultToolkit();
    }
    public void setView(TestPaperView view) {
       this.view = view;
    }
    public void setTestPaper(TestPaper testPaper) {
       this.testPaper = testPaper;
    }
    public void actionPerformed(ActionEvent e) {
       if(e.getSource()==view.nextProblem){
          view.time.start(); //开始计时
          if(testPaper!=null){
            problem = testPaper.nextProblem();
            handleProblem(problem);
          }
          else {
             JOptionPane.showMessageDialog
              (view,"没有试题","消息对话框",JOptionPane.WARNING_MESSAGE);
          }
       }
       if(e.getSource()==view.previousProblem){
          view.time.start(); //开始计时
          if(testPaper!=null){
            problem = testPaper.previousProblem();
            handleProblem(problem);
          }
          else {
            JOptionPane.showMessageDialog
              (view,"没有试题","消息对话框",JOptionPane.WARNING_MESSAGE);
          }

       }
       if(e.getSource()==view.aProblemSubmit){  //确认一道题目的答案
          String answer ="";
          if(view.choiceA.isSelected()){
             answer = answer+"A";
```

```
    }
    if(view.choiceB.isSelected()){
      answer = answer+"B";
    }
    if(view.choiceC.isSelected()){
      answer = answer+"C";
    }
    if(view.choiceD.isSelected()){
      answer = answer+"D";
    }
    if(problem==null) {
       JOptionPane.showMessageDialog
       (view,"没有试题","消息对话框",JOptionPane.WARNING_MESSAGE);
      return;
    }
    view.choiceA.setVisible(false);
    view.choiceB.setVisible(false);
    view.choiceC.setVisible(false);
    view.choiceD.setVisible(false);
    view.aProblemSubmit.setVisible(false);
    problem.setUserAnswer(answer);
}
if(e.getSource()==view.submit){
    testPaper.acceptTeacher(view.teacher); //试卷让老师批阅
    view.renewJButton.setVisible(true);
    view.submit.setVisible(false);
    view.time.stop();
    view.showUsedTime.setText("交卷了");
}
if(e.getSource()==view.renewJButton) {     //再来一套题目
    view.showUsedTime.setText("");
    view.usedTime = view.totalTime;
    view.showUsedTime.setText("考试剩余时间:"+view.totalTime);
    view.showContent.setText(null);
    Image img = tool.getImage("图像管理/renew.jpg");
    handleImage(img);
    view.showImage.repaint();
    view.nextProblem.setVisible(true);
    view.previousProblem.setVisible(true);
    String problemSource= testPaper.getProblemSource(); //得到原始题库
    GiveTestPaper initTestPaper = new RamdomInitTestPaper();
    testPaper=initTestPaper.getTestPaper(problemSource,testPaper.
        getProlemAmount());
    view.renewJButton.setVisible(false);
    view.submit.setVisible(true);
}
```

```
    view.choiceA.setSelected(false);
    view.choiceB.setSelected(false);
    view.choiceC.setSelected(false);
    view.choiceD.setSelected(false);
}
private void handleProblem(Problem problem) {
     if(problem==null) {
         JOptionPane.showMessageDialog
         (view,"没有试题","消息对话框",JOptionPane.WARNING_MESSAGE);
     }
     else {
         view.aProblemSubmit.setVisible(true);
         view.showContent.setText(problem.getContent());
         view.showContent.setVisible(true);
         if(problem.getIsChoice()) {
             handelChoice();
         }
         else if(problem.getIsJudge()) {
             handelJudge();
         }
         String imageName = problem.getImageName();
          //用户必须把图像存放到“图像管理”文件夹
         Image img = tool.getImage("图像管理/"+imageName);
         handleImage(img);
     }
}
private void handelJudge() {
     view.choiceA.setText(problem.getGiveChoiceA());
     view.choiceB.setText(problem.getGiveChoiceB());
     view.choiceA.setVisible(true);
     view.choiceB.setVisible(true);
     view.choiceC.setVisible(false);
     view.choiceD.setVisible(false);
}
private void handelChoice() {
     view.choiceA.setText(problem.getGiveChoiceA());
     view.choiceB.setText(problem.getGiveChoiceB());
     view.choiceC.setText(problem.getGiveChoiceC());
     view.choiceD.setText(problem.getGiveChoiceD());
     view.choiceA.setVisible(true);
     view.choiceB.setVisible(true);
     view.choiceC.setVisible(true);
     view.choiceD.setVisible(true);
}
private void handleImage(Image image) {
     view.showImage.setImage(image);
```

```
            view.showImage.repaint();

        }
    }
```

5.5　GUI 程序

按照源文件中的包语句将 5.4 节中相关的源文件保存到以下目录中：

```
D:\ch5\view\
```

编译各个源文件，例如：

```
D:\>javac ch5/view/IntegrationView.java
```

也可以如下编译全部源文件：

```
D:\>javac ch5/view/*.java
```

根据代码的结构需要把“题库”的文件夹（存放 Excel 工作簿）以及“图像管理”文件夹（存放所需要的图像文件）存放在 D:\下，即保持和 ch5 是同级。

把 5.2 节和 5.4 节给出的类看作一个小框架，下面用框架中的类编写 GUI 应用程序，完成 5.1 节给出的设计要求。

将 AppWindow.java 源文件按照包名保存到以下目录中：

```
D:\ch5/gui
```

编译源文件：

```
D:\>javac ch5/gui/AppWindow.java
```

运行 AppWindow 类（运行效果如本章开始给出的图 5.1）：

```
D:\>java ch5.gui.AppWindow
```

AppWindow.java

```
package ch5.gui;
import ch5.data.GiveTestPaper;
import ch5.data.RamdomInitTestPaper;
import ch5.data.TestPaper;
import ch5.data.TeacherOne;
import ch5.view.TestPaperView;
import ch5.view.IntegrationView;
public class AppWindow {
   public static void main(String []args) {
       String testName="";
       IntegrationView integrationView = new IntegrationView();
       GiveTestPaper initTestPaper = new RamdomInitTestPaper();
```

```
                                                    //创建初始试卷对象
        TestPaper testPaper=
        initTestPaper.getTestPaper("题库/交通理论.xls",5); //得到有 5 个题目的试卷
        TestPaperView testView = new TestPaperView();
        testView.setTestPaper(testPaper);               //设置试卷
        testView.setTeacher(new TeacherOne());          //设置阅卷老师
        testName = "交通法训练";
        testView.setTestName(testName);
        testView.setTotalTime(15);                      //考试时间 15 分钟
        integrationView.addTestPaperView(testName,testView);
        initTestPaper = new RamdomInitTestPaper();      //创建初始试卷对象
        testPaper= initTestPaper.getTestPaper("题库/java 基础.xls",6);
        testView = new TestPaperView();
        testView.setTestPaper(testPaper);
        testView.setTeacher(new TeacherOne());
        testName = "Java 训练";
        testView.setTestName(testName);
        testView.setTotalTime(10);
        integrationView.addTestPaperView(testName,testView);
    }
}
```

5.6 程序发布

用户可以使用 jar.exe 命令制作 JAR 文件来发布软件。

❶ 清单文件

编写以下清单文件（用记事本保存时需要将保存类型选择为“所有文件(*.*)”）：

ch5.mf

```
Manifest-Version: 1.0
Main-Class: ch5.gui.AppWindow
Created-By: 1.8
```

将 ch5.mf 保存到 D:\，即保存在包名所代表的目录的上一层目录中。

注意 清单中的 Manifest-Version 和 1.0 之间、Main-Class 和主类 ch5.gui.AppWindow 之间以及 Created-By 和 1.8 之间必须有且只有一个空格。

❷ 用批处理文件发布程序

使用 jar 命令创建.jar 文件：

```
D:\>jar cfm TestTrain.jar ch5.mf ch5/data/*.class ch5/view/*.class
ch5/gui/*.class
```

其中，参数 c 表示要生成一个新的 JAR 文件，f 表示要生成的 JAR 文件的名字，m 表示清单

文件的名字。如果没有任何错误提示，在 D:\下将生成一个名字是 TestTrain.jar 的文件。

编写以下 train.bat，用记事本保存该文件时需要将保存类型选择为“所有文件(*.*)”。

train.bat

```
path.\jre\bin
pause
javaw -jar TestTrian.jar
```

将该文件保存到自己命名的某个文件夹中，例如名字是 2000 的文件夹中。然后将 TestTrain.jar、“题库”和“图像管理”文件夹以及 JRE（并保证 jxl.jar 文件：操作 Excel 所需要的 API 的 JAR 文件也在“JRE\lib\ext”中）复制到 2000 文件夹中。在 2000 文件夹中再保存一个软件运行说明书，提示双击 train.bat 即可运行程序。

可以将 2000 文件夹作为软件发布，也可以用压缩工具将 2000 文件夹下的所有文件压缩成.zip 或.jar 文件发布。用户解压后双击 train.bat 即可运行程序。

如果客户计算机上肯定有 JRE，可以不把 JRE 复制到 2000 文件夹中，同时去除.bat 文件中的“path.\jre\bin”内容，并提示用户将操作 Excel 所需要的 API 复制到 JRE 的扩展中。

mineClearance.bat

```
echo 将操作 Excel 所需要的 API 复制到 JRE 的扩展中
pause
javaw -jar MineClearance.jar
```

5.7　课设题目

❶ 标准化试题训练系统

在学习本章代码的基础上可以为程序增加任何合理的并有能力完成的功能，但至少要增加下列①～④（⑤可独立进行）所要求的功能。

① 编写几个实现 Teacher 接口的类，使得 AppWindow 可以使用这些类的实例评判试卷。

② 编写几个实现 GiveTestPaper 接口的类，使得 AppWindow 可以使用这些类的实例得到试卷（比如按顺序从题库中获得若干试题，或抽取题库中题号能被 3 除尽的试题等）。

③ 当考试剩余时间不多时（比如剩余时间少于全部用时的 5%）将弹出一个警示对话框警示用户。

④ 增加用户查看试题正确答案的功能。当用户回答某试题答案后可以看见一个按钮，单击该按钮可以查看该试题的正确答案，然后该按钮又变得不可见。

⑤ 将题库更改为某种数据库，同时增加①～④所要求的功能。

❷ 自定义题目

通过老师指导或自己查找资料自创一个题目。