西安电子科技大学

Java 程序设计 课程实验报告

实验名称 命令行个人信息管理程序（PIM GUI）

计算机科学与技术 学院 2103051 班

姓名 张平 学号 21030540006

姓名 张宏喆 学号 21030540005

实验日期 2022 年 05 月 30 日

成 绩

指导教师评语：   指导教师： 年____月____日
实验报告内容基本要求及参考格式 一、实验目的 二、实验内容 三、实验过程 四、实验结果分析 五、实验小结（实验过程感受和建议）

一、实验目的

1. 了解 Java GUI 编程技术，使用 Java 面向对象设计思想进行设计。
2. 了解 AWT 类层次结构及基本组件。
3. 熟悉容器和布局管理器的基本概念。
4. 掌握使用 JFrame、JPanel、JScrollPane、JButton 等组件设计 GUI 的方法。
5. 掌握 FlowLayout、BorderLayout、GridLayout 布局管理器的使用方法。
6. 熟悉 Java 的事件处理模型。
7. 了解 Swing 常用组件的特点和使用方法。

二、实验内容

下面列出之前的实验要求。最开始的要求如下：

This assignment involves the creation of simple Personal Information Management system that can deal with 4 kinds of items: todo items, notes, appointments and contacts. Each of these kinds of items is described in more
detail below. The assignment requires that you create a class for each item type, and that each class extends an abstract base class provided for you. In addition to creating the four classes, you need to create a manager class
that supports some simple text-based commands for creating and managing items.

注释：PIM 可以处理 4 种类别事项：待办事项，备忘，约会和联系人，PIMEntity 是公共抽象父类，创建 PIMManager 进行测试，(有给定名称的要按给定的名称)。

上次实验（PIM with I/O）在前面实验的基础上，改写基于命令行形式的个人信息管理（PIMCmd）程序，要求如下：

The PIM with I/O assignment is a Command Oriented Personal Information Manager (第 5 章实验内容 2) with same general idea (todos, appointments, etc.) stored in one(or more) file(s) accessed by I/O.

Java I/O: you need to be able to save a list of items to a file (and to load - read from a file). You can use a simple text file and the PIMEntity methods toString() and fromString() to generate/parse strings, or you can get fancy and create Serializable objects and use Object streams.

本次实验则在前面两次实验的基础上，要求实现个人信息管理器（GUI 版本）。在支持所有四种 PIMEntity 类型的基础上，通过图形用户界面提供（至少）以下内容：

1) 一次显示一个月的日历，允许用户更改月份（后退和前进按钮都可以）。日历中必须显示所有具有相关日期的条目的文本，即显示从本地文件检索到的、与日期相关的 PIMEntity（待办事项和约会）。日历需要满足预览作业的要求，但不需要确保每个条目（待办事项、约会）在日历中都是完全可见的，只有部分文本出现在正确的日期就可以了。

2) 用户必须能够通过 GUI 创建新的 PIM 条目。

3) 用户必须能够通过 GUI 编辑现有的 PIM 条目。

4) 用户必须能够按条目类型查看所有 PIM 条目，因此应该有某种方式来查看所有待办事项或约会等。各种条目的显示没有必要的格式。

5) 用户必须有某种方式来保存/加载条目，可以是本地文件（文本文件或持久化对象存储）。

6) 必须有一个菜单栏，允许用户从支持的功能（查看日历、查看待办事项、查看联系人等）中进行选择。菜单栏还应包括用于创建新 PIM 条目的菜单项（“新 TODO”、“新约会”等）。

7) 用户应该能够指定一次帐户名（用作 PIMTenties 的所有者）。每次程序启动时都会提示用户名，还可以将其保存在某种配置文件中（或在命令行上指定）。重要的问题是，用户不需要在每次创建新 Entity 时指定所有者。

8) 应使用泛型。如果从本地文件获取实体，则可以只使用 IOException（抛出异常时）。RemotePIMCollection 只是一个接口，用于从存储中获取 PIMCollection。这并不意味着要使用特定的技术。

9) 面向命令的 PIM 相关类定义将用于 GUI 项目。以下内容应更改：

应更改 PIMEntity，使其包含对所有者（字符串）的支持和指示项目是私有还是公共的布尔值。一般来说，使用的简单方案是，如果知道所有者名称，就可以访问私有项（这是由 PIMCollection 中的方法强制执行的，而不是单个 PIMEntity 派生类）。应更改 4 种类型的 Pimentity 中每种类型的构造函数，以支持 owner 和 shared 标志。fromString 方法不再是接口 PIMEntity 的一部分。PIMCollection 类支持更多的方法，特别是有一些方法可以获取公共项（当没有指定所有者时），以及获取与特定所有者对应的公共和私有项。

三、实验过程

1. 实验环境

操作系统：Windows 11

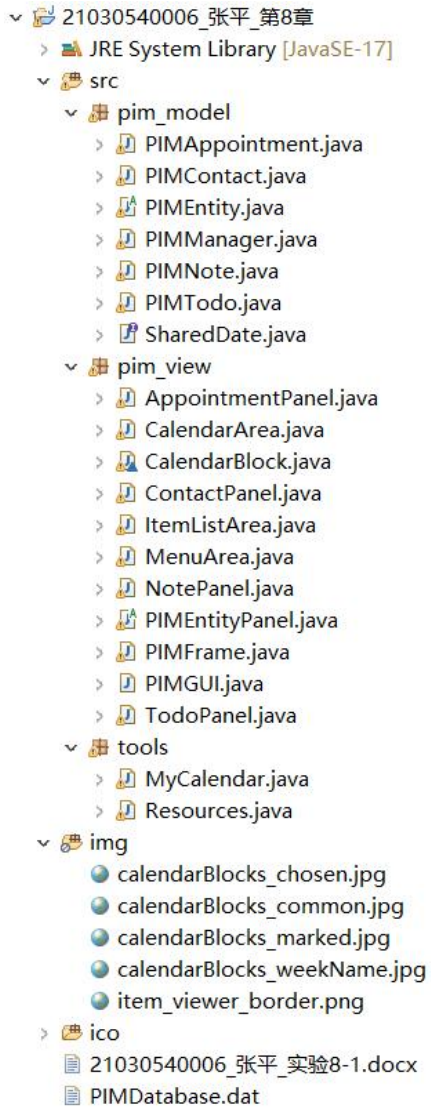
集成开发环境：Eclipse IDE for Enterprise Java and Web Developers (includes Incubating components) 2022-03 (4.23.0)

2. 题目分析

在上次实验的基础上（创建可序列化的对象、并使用对象输入输出流来实现对象的持久化存储），我们这次来实现 GUI 版本的个人信息管理程序。

限于只有两个人，且个人能力有限，加上面临期末复习压力，我们无法完全实现所有的需求，具体来说 7)、8)、9)这三项需求难以完成。特别地，对所有者 Owner 这一项需求我们几人存在理解上的分歧，不太清楚如何随着账户名的变化来使用对应的公共和私有项。

经过讨论，我们决定使用 MVC 模型-视图-控制器设计模式。模型（model）对应 PIMEntity 类（PIMTodo、PIMContact、PIMAppointment、PIMNote）和 PIMManager、以及 GUI 组件中组件的状态、文本域中的文本等，视图（view）对应 GUI 界面（PIMGUI、PIMFrame、MenuFrame、CalendarArea 类等），控制器对应事件监听器，用来处理用户输入事件、实现前后端交互。最后，完成的项目结构如下所示：



各个类之间的关系比较简单，抽象类 PIMEntity 是 PIMTodo、PIMContact、PIMAppointment、PIMNote 的父类，它实现 Serializable 接口，这样其子类也可以被序列化；PIMManager 和 PIMEntity 是组合关系，它管理一组 PIMEntity 对象；PIMGUI 创建主窗口（PIMFrame）的实例，PIMFrame 持有一个 PIMManager 对象，同时与菜单窗格 MenuFrame、日历区域类 CalendarArea 是组合关系。

在程序代码中，我们首先进入 PIMGUI 的 main 方法，初始化 PIMFrame 用户主界面。PIMFrame 会实例化一个 PIMManager 对象用来管理 PIMEntity 列表，同时用 MenuFrame 生成菜单栏（包括 File、Edit、Navigate、Help 四个菜单）。

File 菜单包含 Save、Save As、Load、Load As 和 Exit 菜单项，Edit 菜单包括四类 PIMEntity 对象的创建命令，Navigate 用来实现日历的跳转和各类事项的展示，包括 Jump To Date、Jump To Today、Show All 等多个菜单项，Help 菜单的 About 菜单项说明程序功能。

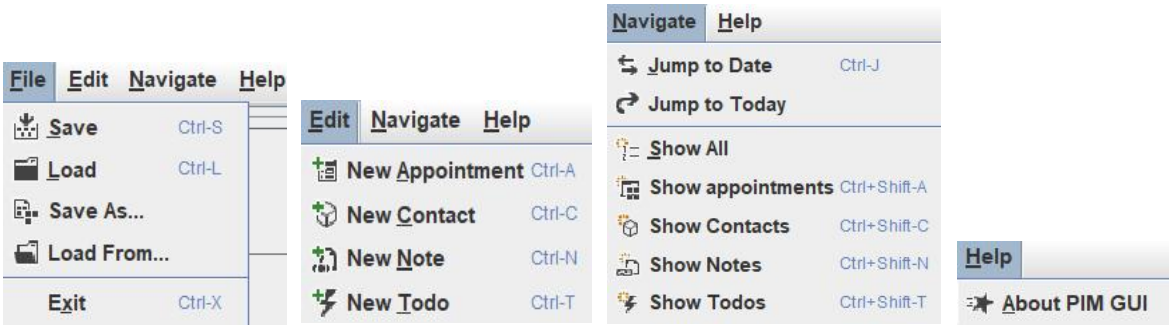
Edit 菜单中没有删除和修改菜单项，是因为 PIMFrame 还会在界面左侧创建一个条目列表，简要显示 PIMEntity 信息，点击相应事件会弹出窗口、可以编辑信息。PIMFrame 类还在界面右侧实例化一个 CalendarArea，动态绘制某年某月的日历，我们可以在 Navigate 中调整对应年月。点击日历中的某天（有背景颜色即代表有事项），就会在左侧显示对应日期的 PIMEntity 事件信息（PIMAppointment 和 PIMTodo）；如果没有事项，就会弹出一个窗口，提示是否创建与该日期有关的事项。

3. 代码实现

由于代码较长，这里只展示关键部分。PIMGUI 类很简单，在其 main 方法（整个 GUI 程序的起点）中用事件分派线程创建 PIMFrame 类对象，初始化并显示用户界面：

```
/**
 * <p>项目名称：PIM GUI
 * <p>类名称：PIMGUI
 * 创建时间：2022年5月31日 <br>
 * 类描述：创建主窗口的实例
 * @author：张平
 */
public class PIMGUI {
    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            PIMFrame frame;
            try {
                frame = new PIMFrame();
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setVisible(true);
            } catch (IOException e) {
                e.printStackTrace();
            }
        });
    }
}
```

菜单栏是本程序的关键部分，它实现了程序中的大部分功能调用。菜单栏实现后的效果如下。为了美观，我们添加了图标；为了使用方便，我们还设置了助记符和相应的快捷键：

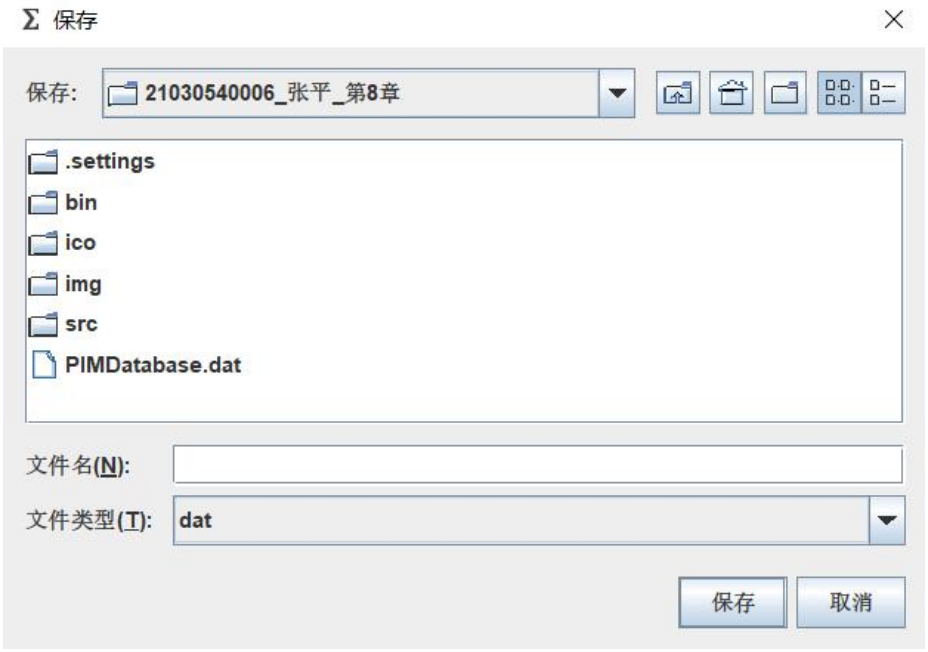


下面简要介绍菜单栏的实现代码。菜单栏中 File 菜单的关键功能是 Save As 和 Load From。为了实现这两种功能，我们使用了 Swing 提供的 JFileChooser 类，用来创建文件保存/打开对话框。对于另存为功能，我们可以选择一个目录来自动新建一个数据文件，或者自定义数据文件名；对于加载功能，我们必须选择一个扩展名为.dat 的文件。

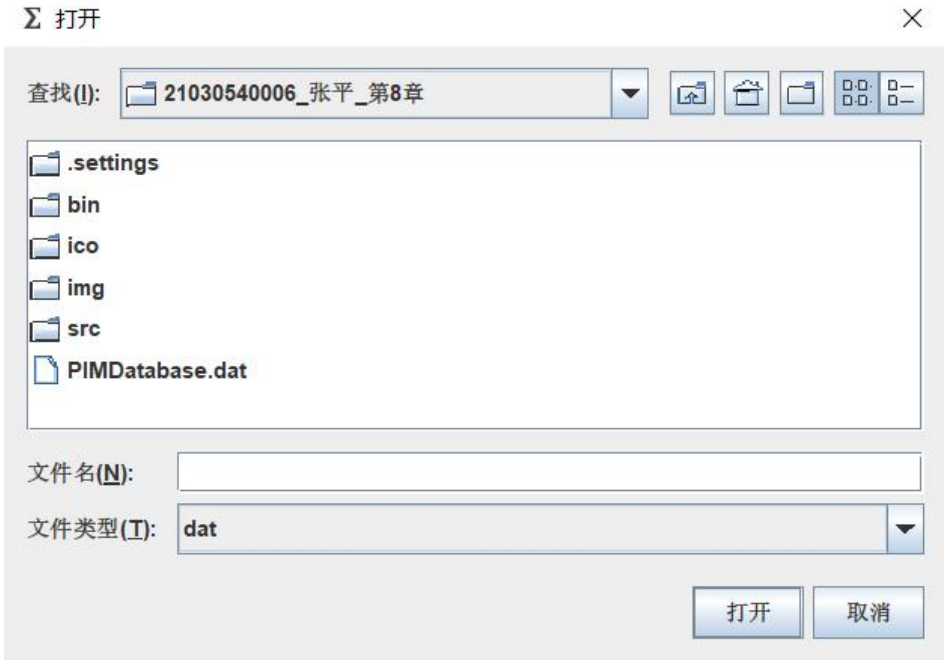
```java
/**
 * 可以选择目录,会自动在该目录下新建一个PIMDatabase.dat
 * 可以选择已存在的dat文件,方法会删除该文件,并新建同名文件
 * 还可输入带.dat扩展名的新文件名,创建新文件
 */
private void saveAs() {
    try {
        JOptionPane.showMessageDialog(null, "Please Input Extension Name "
                + "'.dat' Manually", "Tip", JOptionPane.WARNING_MESSAGE);
        int result = chooser.showSaveDialog(topFrame); // 显示保存对话框
        if (result == JFileChooser.APPROVE_OPTION) {
            String filename = null;
            File file = chooser.getSelectedFile();
            if (file.isDirectory()) { // 选择了一个目录,则在该目录下新建一个数据文件
                filename = chooser.getSelectedFile().getPath() + "PIMDatabase.dat";
            } else if ( // 选择了一个文件且文件以.dat结尾,或者输入一个带有.dat的文件名
                    new FileNameExtensionFilter("dat", "DAT").accept(file)) {
                if (file.isFile()) // 若文件存在则删除,后续会新建同名文件
                    file.delete();
                filename = file.getPath();
            }
            topFrame.getPIMManager().setDataFilePath(filename);
            save(); // 调用save方法
        }
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null, "Something Wrong Happens...", "Wrong",
            JOptionPane.ERROR_MESSAGE);
    }
}

private void loadFrom() {
    try {
        int result = chooser.showOpenDialog(topFrame); // 显示打开对话框
        if (result == JFileChooser.APPROVE_OPTION) {
            File file = chooser.getSelectedFile();
            if (file.isFile() && // 选择了一个文件
                new FileNameExtensionFilter("dat", "DAT").accept(file)) { // 如果选择的文件以.dat结尾
                topFrame.getPIMManager().setDataFilePath(file.getPath());
                load(); // 调用load方法,其内部进行刷新
            }
        }
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null, "Something Wrong happens...", "Wrong",
            JOptionPane.ERROR_MESSAGE);
    }
}
```

这两种功能实现后的窗口如下所示。点击 Save As，首先弹出提示框，点击确认后会弹出文件保存对话框，保存之后还会报告结果：



点击 Load From 则会直接弹出文件打开对话框，选择文件、完成数据加载后同样会报告结果：

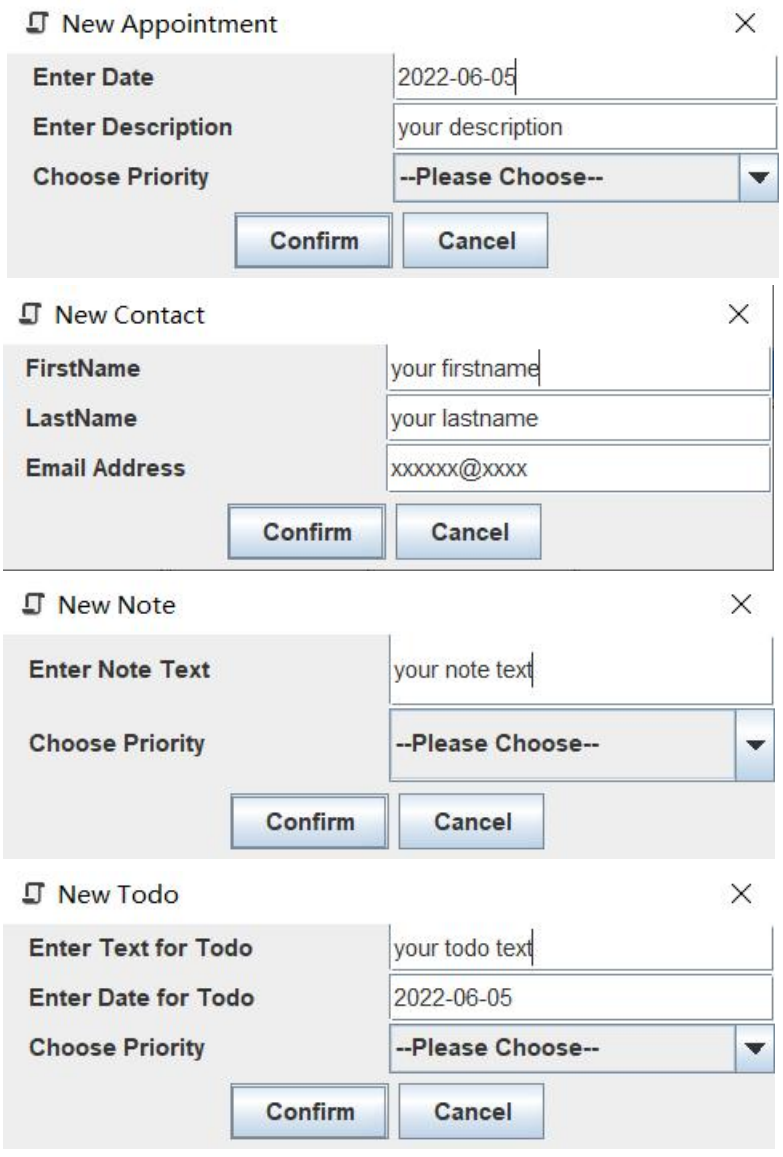


菜单栏中 Edit 菜单用来创建事项，四个菜单项对应的四个界面类都继承自抽象类 PIMEntityPanel，它的 showDialog 方法可用来创建对话框。只要继承该抽象类，并在子类的构造函数中自定义要显示的 Panel，就可以使用 showDialog 显示不同的对话框。我们的这一实现，一定程度上参考了 JDialog 类提供多个预定义对话框的方法。

```java
/*
 * 创建并在对话框中展示当前panel
 * @param parent是所有者窗口,title是对话框标题
 */
public boolean showDialog(Component parent, String title) {
    changed = false;
    Frame owner = null;
    if (parent instanceof Frame)
        owner = (Frame)parent;
    else
        owner = (Frame)SwingUtilities.getAncestorOfClass(Frame.class, parent);
    // 如果是第一次或者所有者改变,则创建对话框,使得相应对话框对象只创建一次
    if (dialog == null || dialog.getOwner() != owner) {
        dialog = new JDialog(owner, true);
        dialog.setIconImage(Resources.newShowEditIco.getImage());
        dialog.setSize(Resources.DIALOG_DEFAULT_WIDTH, Resources.DIALOG_DEFAULT_HEIGHT);
        dialog.add(this); // 将当前panel添加进对话框
        dialog.getRootPane().setDefaultButton(confirmButton);
        dialog.pack();
    }

    dialog.setTitle(title);
    dialog.setLocationRelativeTo(owner); // 调整窗口位置
    dialog.setSize(Resources.DIALOG_DEFAULT_WIDTH, Resources.DIALOG_DEFAULT_HEIGHT);
    dialog.setVisible(true); // 使对话框可见,在用户关闭这个对话框之前会阻塞
    return changed;
}
```

点击 Edit 菜单中的几个菜单项，结果如下所示。显示的几个对话框中设置了默认的文本：



Navigate 菜单中，只有 Jump To Date 有对话框，要求用户输入要跳转的年月。如果输出日期，它会弹出报错窗口：

其他几个 Show 菜单项，对应的监听器代码只是简单调用了 ItemListArea 提供的、用来刷新左侧事件列表区域的方法：

```
showAppointmentsItem.addActionListener(event -> {
    topFrame.getItemListArea().refreshLabelList(
        getType(topFrame.getPIMManager().getItemList(), "Appointment"));
});
showContactsItem.addActionListener(event -> {
    topFrame.getItemListArea().refreshLabelList(
        getType(topFrame.getPIMManager().getItemList(), "Contact"));
});
showNotesItem.addActionListener(event -> {
    topFrame.getItemListArea().refreshLabelList(
        getType(topFrame.getPIMManager().getItemList(), "Note"));
});
showTodosItem.addActionListener(event -> {
    topFrame.getItemListArea().refreshLabelList(
        getType(topFrame.getPIMManager().getItemList(), "Todo"));
});
```

最后是 Help 菜单，它的 About PIM GUI 菜单项界面如下所示：

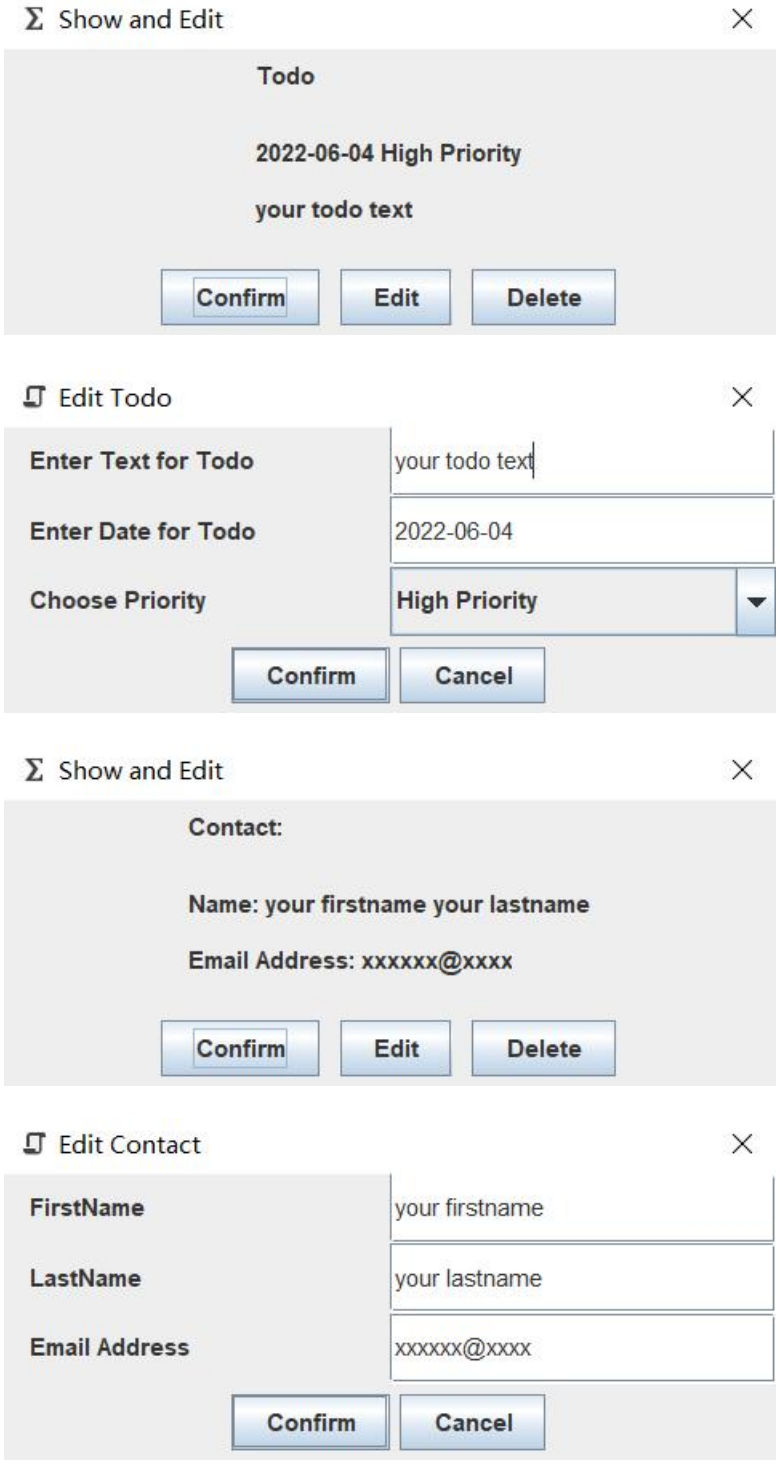
菜单栏下，左侧事件列表的实现很简单，只是一个 JScrollPane 内嵌一个 JLabel 列表。

```
public ItemListArea(PIMFrame topFrame) {
    this.topFrame = topFrame;
    itemListBox = Box.createVerticalBox();

    itemListScrollPane = new JScrollPane(itemListBox);
    itemListScrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
    itemListScrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

    itemListScrollPane.setPreferredSize(new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT));
    itemListScrollPane.setAutoscrolls(true);
    add(itemListScrollPane, BorderLayout.CENTER);
    refreshAll();
}
```

鼠标点击事件列表中的各个事项，会弹出用来展示事件信息的窗口，部分窗口如下所示。点击 Comfirm 按钮，可退出弹窗；点击 Edit 按钮，则会弹出事项编辑界面；点击 Delete 按钮，则会删除事项。

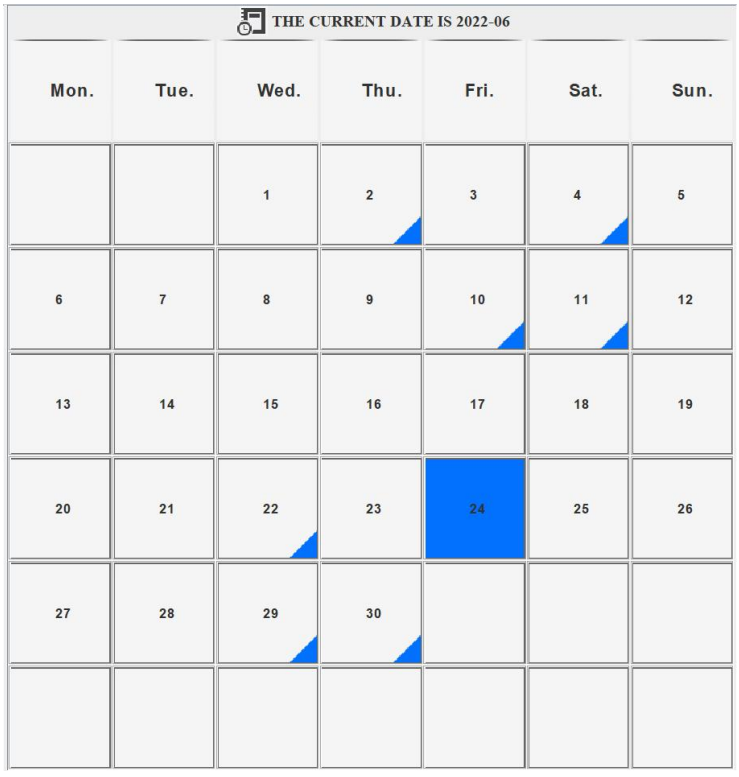


在 CalendarArea 类中，实现右侧日历绘制的代码如下，它创建了一个 CalendarBlock 对象数组，用来显示每天的日期方格。CalendarBlock 类中还定义了鼠标事件的处理方法：

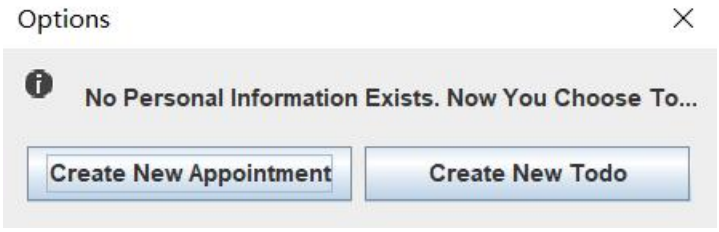
```
private void paintAll() {
    // 开始绘制界面，同时为每个标签加入交互
    String tmp = Resources.sdf.format(date);
    String text = "THE CURRENT DATE IS " + tmp.substring(0, 7);
    JLabel tip = new JLabel(text, Resources.timeLabelIco, JLabel.CENTER);
    tip.setFont(Resources.markedRomanFont);
    rootPanel.add(tip);

    for (int i = 0; i < 7; ++i) {
        dayOfWeek[i] = new JLabel(dayOfWeekName[i], Resources.calendarBlockWeekName, JLabel.CENTER);
        dayOfWeek[i].setFont(Resources.markedConSolasFont);
        dayOfWeek[i].setIconTextGap(-60);
        dayOfWeek[i].setOpaque(true);
        basePanel.add(dayOfWeek[i]);
    }
    int k = 0;
    // 创建六周42个CalendarBlock,用来表示每天
    for (int i = 0; i < 6; ++i) {
        for (int j = 0; j < 7; ++j) {
            LocalDate d = null;
            if (monthCalendar[i][j] != null) {
                d = LocalDate.of(date.getYear(), date.getMonth(), Integer.parseInt(monthCalendar[i][j]));
            }
            dayOfMonth[k] = new CalendarBlock(monthCalendar[i][j], d, SwingConstants.CENTER, topFrame);
            dayOfMonth[k].setFont(Resources.conSolasFont);
            dayOfMonth[k].setBorder(BorderFactory.createEtchedBorder());
            dayOfMonth[k].setIconTextGap(-55);
            dayOfMonth[k].setOpaque(true);
            dayOfMonth[k].addMouseListener(dayOfMonth[k]);
            basePanel.add(dayOfMonth[k++]);
        }
    }
}
```

最后绘制出的 2022 年 6 月的日历如下所示，蓝色右下角的方格代表该天存在事项，全蓝色方格代表鼠标光标移到此处。通过 Navigate 菜单的 Jump To Date 菜单项，我们可以跳到任意年月的日历：



点击已经存在事项的方格，会在左侧事项列表中显示所有具有相关日期的事件，此时可以通过 Navigate 菜单中的 Show All 菜单项，重新显示所有事件。点击不存在事项的白色方格，则会弹出如下窗口，要求创建与该日期相关的事项：



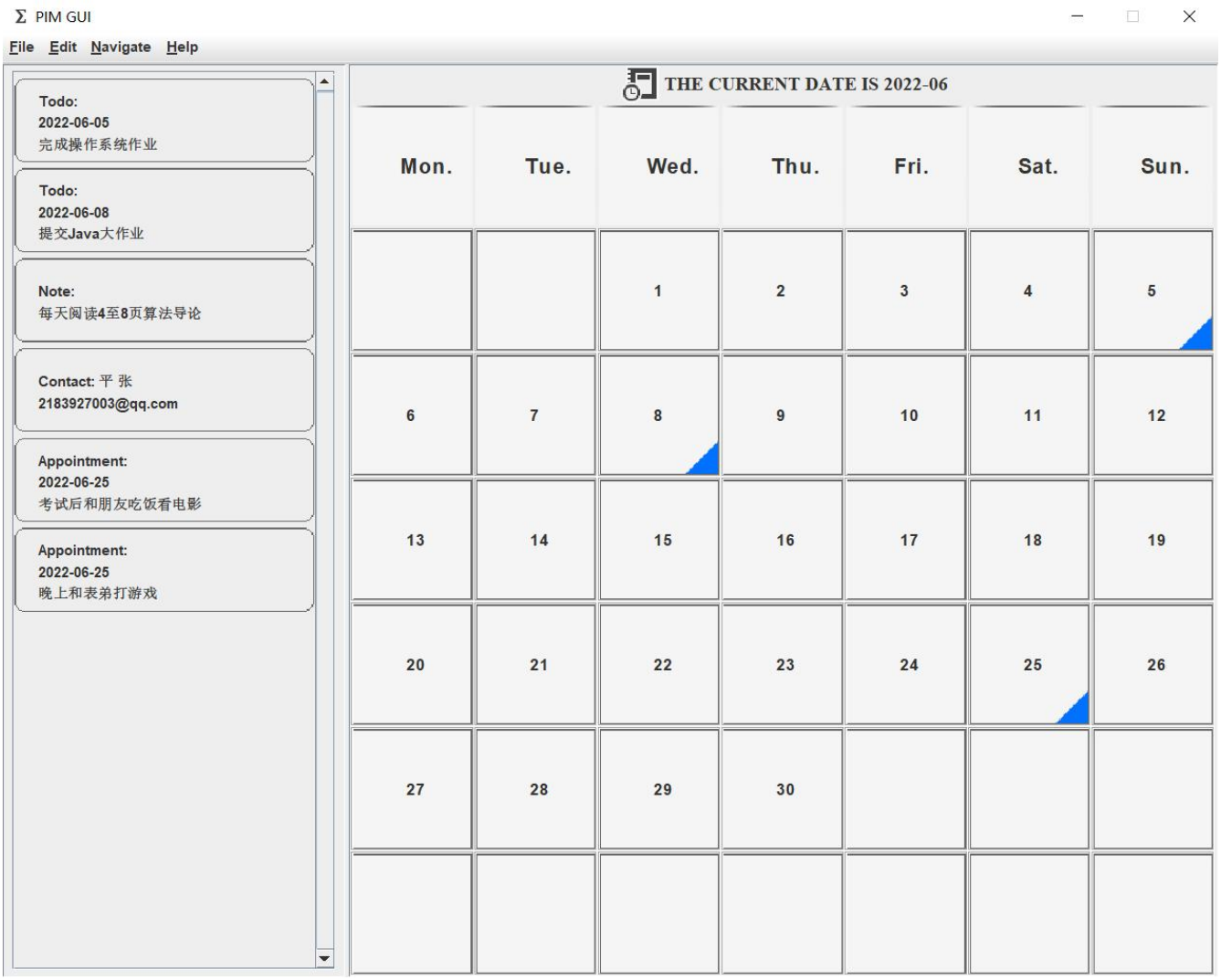
点击按钮后，会弹出相应的事件创建窗口，比如点击 16 号的方格、选择创建 TODO 后弹出的事件创建窗口如下，对应的日期已经**自动填入**：

四、实验结果分析

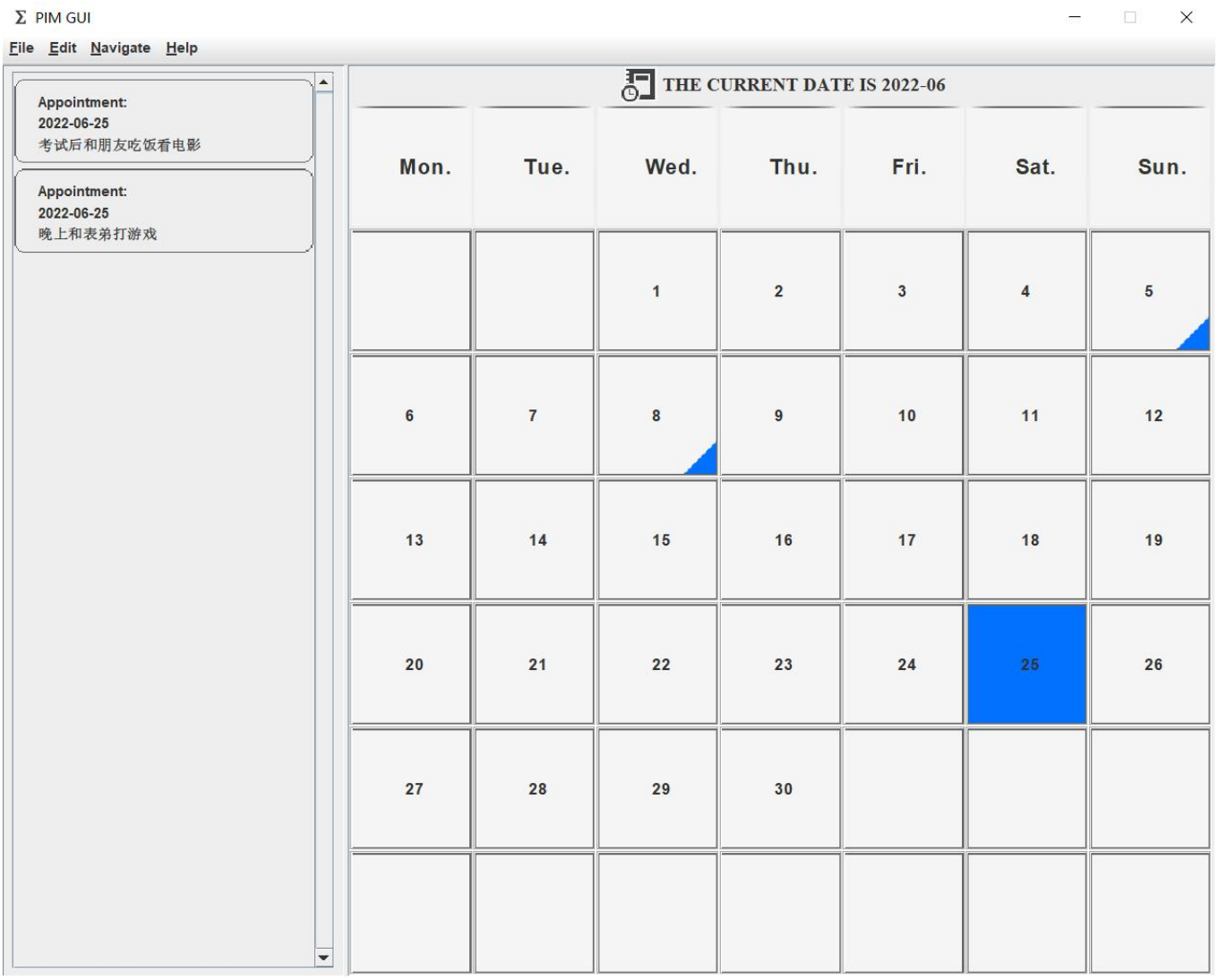
完成程序编写后，在 Eclipse 中点击 Run。在 GUI 界面中使用 Edit 菜单项，依次输入如下信息：

接着用快捷键 CTRL+S 保存数据到默认文件：

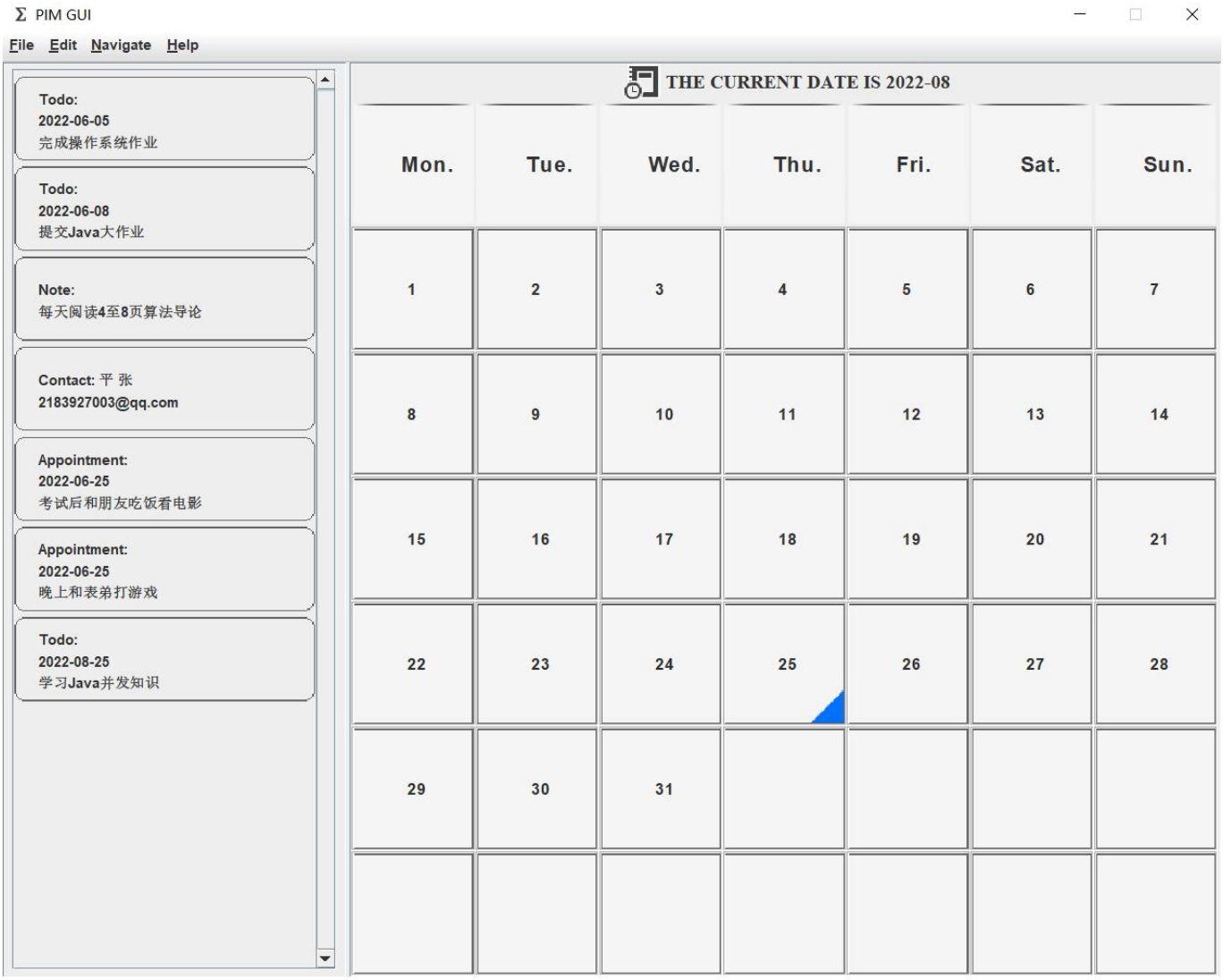
现在的界面如下所示：



点击 25 号方格，左侧事项列表显示当天的事件：



如果用 Jump To Date 跳到其他月份，左边会重新显示所有事件，右边也会重新绘制。比如，我们先创建一个 2022 年 8 月 25 日的 TODO，再跳到该月：



可见，实验结果是正确的。

五、实验小结

为了完成本次实验的 Java GUI，我特意把《Java 核心技术卷 I》第 10 章图形用户界面程序设计、第 11 章 Swing 用户界面组件和《Java 核心技术卷 II》第 2 章输入与输出仔细看了一遍，虽然有所收获，但还是感觉细节太多、难以掌握。

因为我们第一次（合作）实现这种 GUI+后端处理的项目，整个项目代码较长，相对比较混乱。这也让我们认识到编写代码之前画好类图、做好 UI 设计等的重要性。如果不画好类图，各个类之间的关系就一团乱麻、难以掌握。如果没有简单的 UI 界面示例，实现起界面来就难以确定各个组件的位置。协同工作也是一个问题，我们两人的代码风格有许多不一致的地方，因此代码看起来比较凌乱。

当然，为了完成本次的实验，我们还把之前写的 PIMManager 类、PIMEntity 类（及其子类）做了一些修改，之前的实现还是比较合适的。不过，本次的代码感觉还有很大的改善空间，个人认为有些重复代码需要优化，只是狗咬刺猬——不知从何处下手，以后还需学习设计模式思想，多向优秀开源项目学习。

此外，由于只是初步学习了 Java AWT 和 Swing 的基础知识，不会对样式进行美化，也没有时间挑选好看的图标，整个用户界面相对来说比较原始和简陋。而且对 GUI 的调试有些繁琐，可能有些隐藏的 Bug 没有找出。