河南工業大學

课 程 设 计

课程设计名称：Java 课程设计

专 业 班 级 ：计算机科学与技术 F1702

学 生 姓 名 ：刘 飞

学 号 ：201716010618

指 导 教 师 ：李 磊

课程设计时间：2019.6.3—2019.6.14

计算机科学与技术 专业课程设计任务书

学生姓名	刘飞	**专业班级**	计科 F1702	**学号**	201716010618
题　目	万年历				
学期	2018-2918-2		**课程名称**	Java 课程设计	
指导教师	李磊		**同组姓名**	无	
主要内容	制作一个万年历，包括以下功能： **基本功能：** （1） 获取当前日期并显示 （2） 获取当前时间并显示 （3） 能够跳转到任意时期并显示 （4） 使用图形用户界面 **扩展功能：** （1） 能够显示节日提醒、占卜提示 （2） 能够更改界面风格（换肤） （3） 能够设置闹钟并提醒				
任务要求	1)能够理解系统的用户和服务对象,并能**分析系统预期带来的经济与社会效益；** 2）根据所选题目搜集资料完善业务需求，了解已有设计思路和方案，重点是要结合最新社会需求和个人理解，**创新性地提出新需求**； 3）根据给定业务需求，采用面向对象思想对业务问题进行分析，并给出总体设计方案、详细设计方案，并用 Java 语言实现；在方案设计时，充分发挥创新性； 4）能够分析和评价系统方案产生的**社会、健康、安全、法律和文化影响**，要在最终实现方案中体现相应的不良影响规避措施。				
参考文献	[1] 赵满来.可视化 Java GUI 程序设计实验指导[M]. 清华大学出版社，2016 [2] 耿祥义.Java 2 实用教程(第 5 版)[M]. 清华大学出版社,2017. [3] 耿祥义.Java 课程设计(第 3 版)[M]. 清华大学出版社,2018 . [4] 林智杨、范明翔等. 深入浅出 Java Swing 程序设计[M]. 中国铁道出版社,2005 [5]王晓哲.Java Swing 组件技术[J].天津职业院校联合学报,2008(03):138-142+145. [6]潘国荣.Java 中的常见事件及处理探究[J].电脑知识与技术,2018,14(29):125-126+131.				
审查意见	**指导教师签字：** **教研室主任签字：**　　年　　月　　日				

说明：本表由指导教师填写，由教研室主任审核后下达给选题学生，装订在设计（论文）首页

目录

1. 需求分析

对于日期，1、3、5、7、8、10、12 月有 31 天；4、6、9、11 月有 30 天；闰年的 2 月有 29 天，平年有 28 天。每 4 年一个闰年，每 100 年一个闰年，每 400 年一个闰年。根据以上条件，给定任意的年份和月份，便可计算出当月的天数。

对于时间，将表盘分为 12 等份，每一份对应一小时；将表盘分为 60 等份，每一份对应一分钟；将表盘分为 3600 等份，每一份对应一秒钟。

每秒钟分针转过的角度：`minute_angle=(minute+second/60)*360/60;`

每秒钟时针转过的角度：`hour_angle=(hour-12+minute/60)*360/12;`

根据以上条件，便可以计算出任意时刻时分秒针在表盘上的具体位置，利用 Graphics2D 类在面板上绘制即可。

将标签和日期进行关联这样就可以将日期的具体信息显示出来，比如节假日、每日运势、黄道吉日等。

通过调用系统时间，制作一个计时装置，结合窗口和标签实现闹钟的功能。

通过设置时钟、日历、标签等的背景和背景图片可以实现更改界面风格，从而实现换肤功能。

再结合菜单、面板、标签、按钮，做出一个简单实用的可视化图形用户界面。

万年历是一个非常实用的小工具。然而在当代，万年历的功能已经不仅仅局限于看时间、查日期的功能了。为此，本系统加入了更加人性化的换肤功能，以及设置闹钟的功能。让小小的万年历全面接管时间！

2 概要设计

2.1 类间组合框架

为了实现代码复用、降低源程序各个模块内的耦合性以及方便修改源代码，将各个类按照层次自顶向下分层设计。利用对象组合的设计方法，让上层类组合下层类的对象，尽量将功能下沉。

各个类之间的组合关系如图所示：

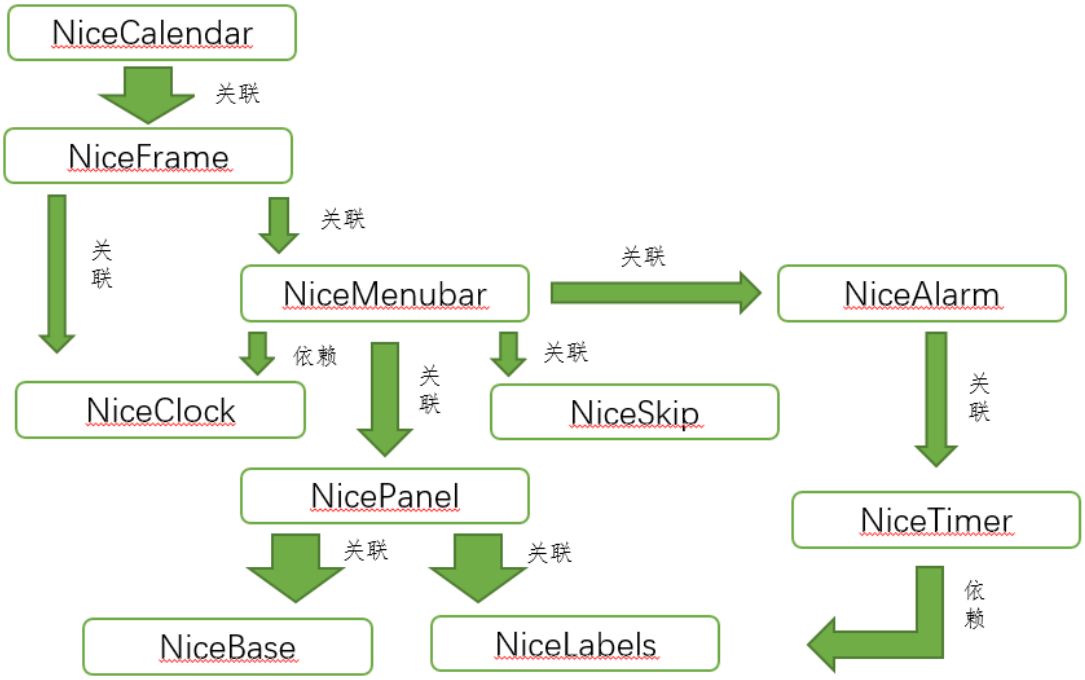

图 2.1 类之间的组合关系

2.2 布局结构示意

万年历的主框架在一个 NiceFrame 里，通过继承 JFrame 而来。包含 NiceMenubar、NicePanel、Panel_Left 三大模块。

NiceMenuba 包含 Start、Function、Help 三个菜单；当用户点击菜单项的时候，对应的部分会做出反馈并显示在面板上。

Panel_Left 上边放 NIceClock，下边放 NiceLabels；NiceLabels 包含三个标签，通过监听日历的点击事件显示内容。

NicePanel 使用 BorderLayout 布局，分为 Panel_North、Panel_Center、Panel_South

三个部分；Panel_Center 负责通过调用 NiceBase 获得计算出的日历格式，并显示在面板上。

各模块的窗口布局如图所示：

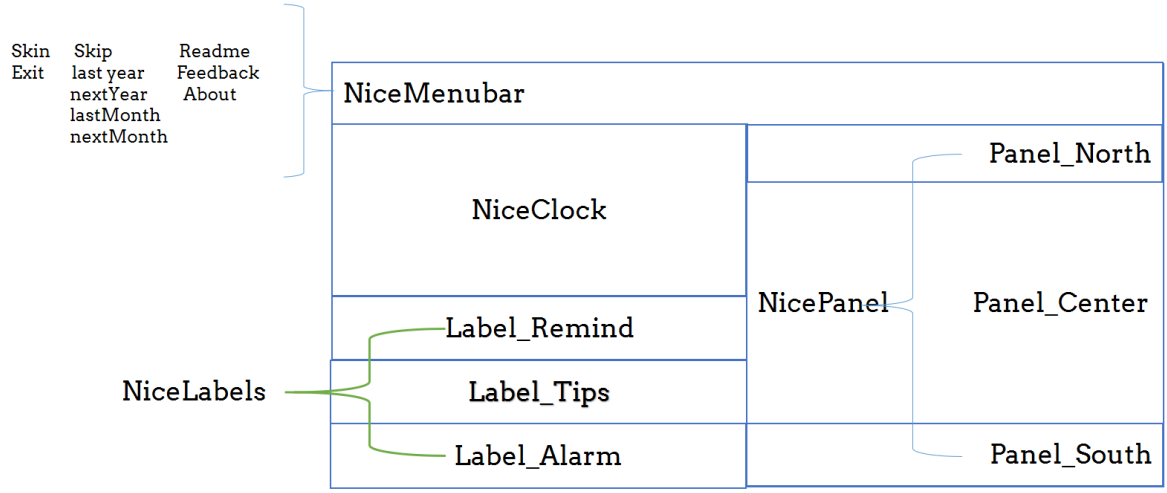

图 2.2 布局示意图

2.3 对各个类的概述

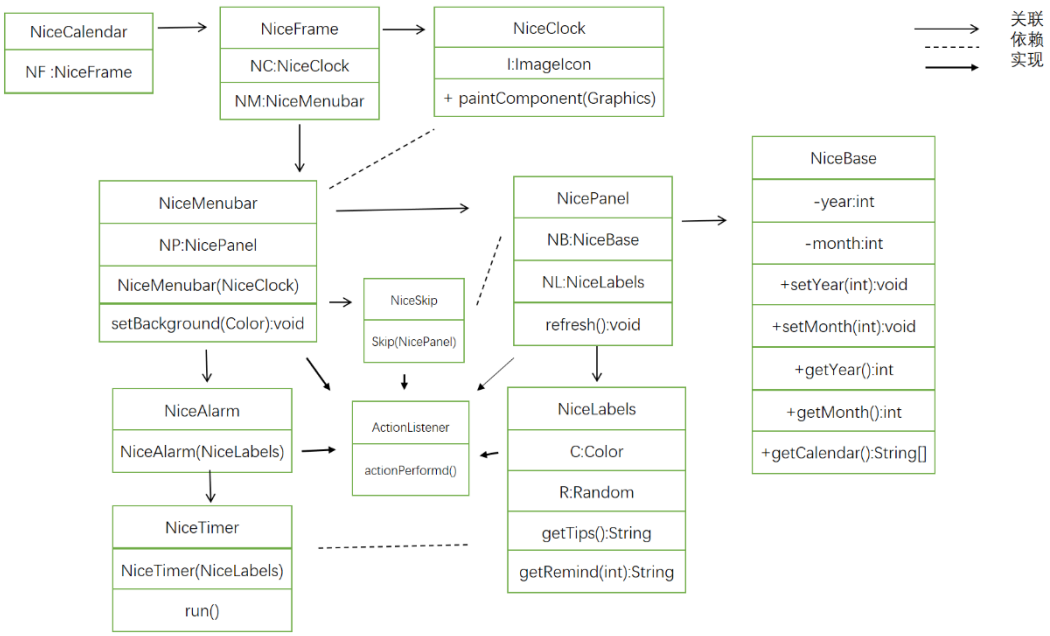

图 2.3 程序 UML 图

NiceCalendar 类，包含程序的入口，对窗口的基本设置，如窗口大小、图标、名称等。

```
public class NiceCalendar{
NiceFrame NF;
```

```
}
```

NiceFrame 类，继承 JFrame 类，是程序的主框架负责窗口的整体布局。

```
public class NiceFrame extends JFrame{
    NiceClock NC;
    NiceMenubar NM;
    JPanel Panel_Left;
}
```

NiceClock 类，继承 JPanel，实际上是一个面板，通过重载 paintComponent 方法，利用 Graphics2D 将时钟绘制在面板上。

```
class NiceClock extends JPanel{
    ImageIcon I;
    NiceClock(ImageIcon I){}
    public void paintComponent(GraphicsG){}
}
```

NiceMenuBar 类,继承了 JMenuBar;组合了 NiceClock,实现对时钟的换肤功能;实现 ActionListener 接口，对菜单项进行事件监听。定义的 changeSkin 方法对全局更改背景颜色。

```
public class NiceMenubar extends JMenuBar implements
ActionListener{
    NicePanel NP;
    NiceClock NC;
    NiceMenubar(NiceClock NC){}
    public void actionPerformed(ActionEvent e){}
    public void changeSkin(Color C){}
}
```

NiceSkip 类，继承 JFrame，引用 NicePanel，主要功能是在 Skip 按钮按下后，调用 NicePanel 的 refresh 方法刷新日历。

```
public class NiceSkip extends JFrame implements ActionListener{
    NicePanel NP;
    void Skip(NicePanel NP){}
    public void actionPerformed(ActionEvent e){}
}
```

NicePanel 类，组合了 NiceBase 和 NiceLabels，负责日历的绘制与刷新，在标签上显示日期、节日、占卜。

```
public class NicePanel extends JPanel implements ActionListener {
   NiceBase NB;
   NiceLabels NL;
    void refresh(){}
    public void actionPerformed(ActionEvent e) {}
```

```
}
```

NiceBase 类，提供日历的计算方法，接收指定的年份和月份，并返回一个当月日历的数组。

```
public class NiceBase{
   //String day[];
   pricate int year;
   private int month;
    public void setYear(int year){}
    public int getYear(){}
    public void setMonth(int month){}
    public int getMonth(){}
    public String[] getCalendar(){}
}
```

NiceLabels 类，包含三个标签，用来显示节日、占卜、闹钟。

```
public class NiceLabels extends JPanel{
   Color C;
    Random R;
    String getTips(){}
    void getRemind(int num){}
}
```

NiceAlarm 类，继承 JFramen 类，是一个弹出窗口，接收 TextField 输入的时间，并调用 NiceTimer 类计时。

```
public class NiceAlarm extends JFrame implements ActionListener{
    NiceLabels NL;
}
```

NiceTimer 类，接收时间，根据系统当前时间进行倒计时。

```
Public class NiceTimer{
   NiceLabels NL;
   int goalHour;
   int goalMinute;
    public void run() {}
}
```

根据需求分析，能够得出需要哪些功能、哪些类。定义这些类，构建类之间的组合关系，为之后的详细设计奠定基础。系统中各个类都是一个整体，使用组合的方法，当需要使用某个标签、按钮的时候仅仅去调用即可，降低代码的耦合性。

在使用 Java 的 Calendar 类获取当前日期的时候，需要进行换算。因为国外的星期的第一天是 Sunday，而中国一周的第一天是 Monday，这就是文化差异。为了规避差异，需要将 `Calendar.get(Calendar.DAY_OF_WEEK)-1;`还有时差的问题：

使用 `System.currentTimeMillis();` 获取的系统时间解析后得到的时间与我国的时间不一致，这是因为我国所处的时区是东八区，因此需要加上八个时区才可以，这在换算的时候应该考虑到。

3. 运行环境

Java Development Kit 11

Windows 10

Intel(R) Core(TM)i5-6200U

2.50GHz CPU 主频

8GB RAM

256GB SSD

图3.1 Windows10

4. 开发工具和编程语言

Visual Studio Code

Java

图4.1 Visual Studio Code

5. 详细设计

5.1 NiceCaelendar 类

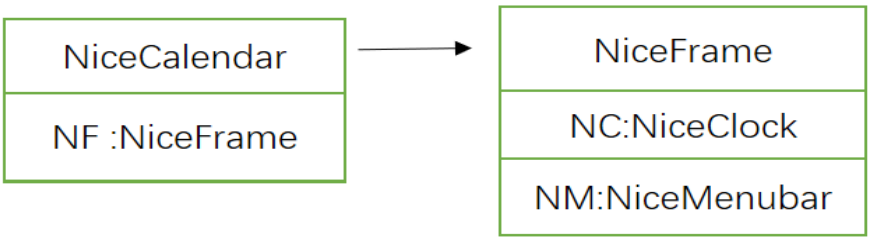

图 5.1 NiceClendar UML 图

设置窗口属性，提供程序的入口方法。

```
public class NiceCalendar{
    public static void main(String[] args){ //main 方法 //程序入口
        NiceFrame NF = new NiceFrame();       //Window//组合
        NF.setBounds(200,200,1200,600);    //设置窗口大小 //1200X600
        Image image =                  // 将图片转换为 Image 对象
Toolkit.getDefaultToolkit().createImage("logo.jpg");
        NF.setIconImage(image);                //设置窗口图标
        NF.setTitle("Nice Calendar");          //title //窗口标题
        NF.setVisible(true);  //let the window visible //窗口可见
     }; //end of main
} //end of class NiceCalendar
```

5.2 NiceFrame 类

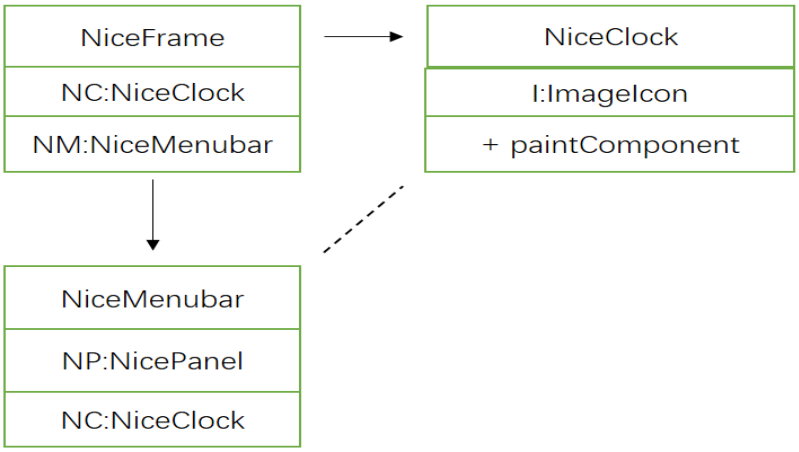

图 5.2 NiceFrame UML 图

JFrame 类型的窗口，将窗口分为菜单、左上、左下、右四个部分。菜单放

NiceMenuBar；左上放 NiceClock；左下放 NiceLabels；右放 NicePanel。

```
    NiceClock NC = new NiceClock(new ImageIcon("WHITE.jpg"));
   /*background NiceClock 的背景图片*/
    NiceMenubar NM = new NiceMenubar(NC);  //组合 NiceMenubar
    JPanel Panel_Left = new JPanel(new GridLayout(2, 1));
     /*set layout //设置布局 将面板分成上下两等
    public NiceFrame(){   //constructer //构造方法
        setVisible(true);              //visible
        setDefaultCloseOperation(DISPOSE_ON_CLOSE); //默认关闭方式
        Panel_Left.add(NC);           //add NiceClock
        Panel_Left.add(NM.NP.NL);    //add NiceLabels
        setLayout(new GridLayout(1, 2));    //set layout //左右等分
        setJMenuBar(NM);              //add Menubar
        add(Panel_Left);            //add Panel_Left
        add(NM.NP);               //add NicePanel
        NM.NP.refresh();       //show default calendar //获取初始日
历
    } //end of constructer
```

5.3 NiceClock 类

图 5.3 NiceClock UML 图

NiceClock 类，继承 JPanel 类，通过重写父类的 printComponent 方法将表盘绘制在面板上。绘制过程包括：打开抗锯齿、放入背景图片、计算表盘半径、绘制数字、绘制刻度、绘制时针、绘制分针。

```
   public void paintComponent(Graphics G){ //draw components
       Graphics2D G2D = (Graphics2D)G;
       G2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);    //抗锯齿 使指针显示逼真
       G2D.setColor(Color.RED); //前景颜色
       G2D.drawImage(I.getImage(), 0, 0, getWidth(), getHeight(),
I.getImageObserver()); //set background
        int x_Width = this.getWidth() / 2;  //get the width of
panel
```

```
        int y_Width = this.getHeight() / 2;  //get the height of
panel
        int radius = (int) (Math.min(this.getWidth(),
this.getHeight()) * 0.8 * 0.5);  //calculate the radius of clock
        for(int i = 0; i < 12; i++){
            double angle = Math.PI / 180 * i * (360 / 12);
            int x = (x_Width - 4) + (int)((radius - 12) *
Math.cos(angle));
            int y = (y_Width + 4) + (int)((radius - 12) *
Math.sin(angle));
            int j = i + 3;
            if (j > 12)
                j = j - 12;
            G2D.drawString(Integer.toString(j), x, y);  //add
numbers to clock
        }
           AffineTransform old = G2D.getTransform();
/*transform 2D coordinates to another*/
        for (int i = 0; i < 60; i++){
            int w = i % 5 == 0 ? 6 : 3;   //判断刻度的大小
             G2D.fillRect(x_Width - 2, 28, w, 3);    //绘制刻度
            G2D.rotate(Math.toRadians(6), x_Width, y_Width);
         } // add marks to clock
         G2D.setTransform(old);
         Calendar C = Calendar.getInstance();
         int hour = C.get(Calendar.HOUR_OF_DAY);
         int minute = C.get(Calendar.MINUTE);
         int second = C.get(Calendar.SECOND);

        double hour_angle = (hour - 12 + minute / 60d) * 360d /
    12d;
         G2D.rotate(Math.toRadians(hour_angle), x_Width, y_Width);
         int x_hour_array[] = { x_Width, x_Width + 9, x_Width,
x_Width - 9 };
         int y_hour_array[] = { 65, y_Width, y_Width + 10,
y_Width };
         G2D.drawPolygon(x_hour_array, y_hour_array,
x_hour_array.length);            //绘制时针
         G2D.setTransform(old);

        double minute_angle = (minute + second / 60d) * 360d / 60d;
         G2D.rotate(Math.toRadians(minute_angle), x_Width, y_Width);
         int x_minute_array[] = { x_Width, x_Width + 6, x_Width,
x_Width - 6 };      //分针长度
```

```
        int y_minute_array[] = { 45, y_Width, y_Width + 12,
y_Width };
        G2D.drawPolygon(x_minute_array, y_minute_array,
x_minute_array.length);          //绘制分针
        G2D.setTransform(old);
     }
    NiceClock(ImageIcon I){
        this.I = I;            //接收背景图片
    }
```

5.4 NiceMenubar 类

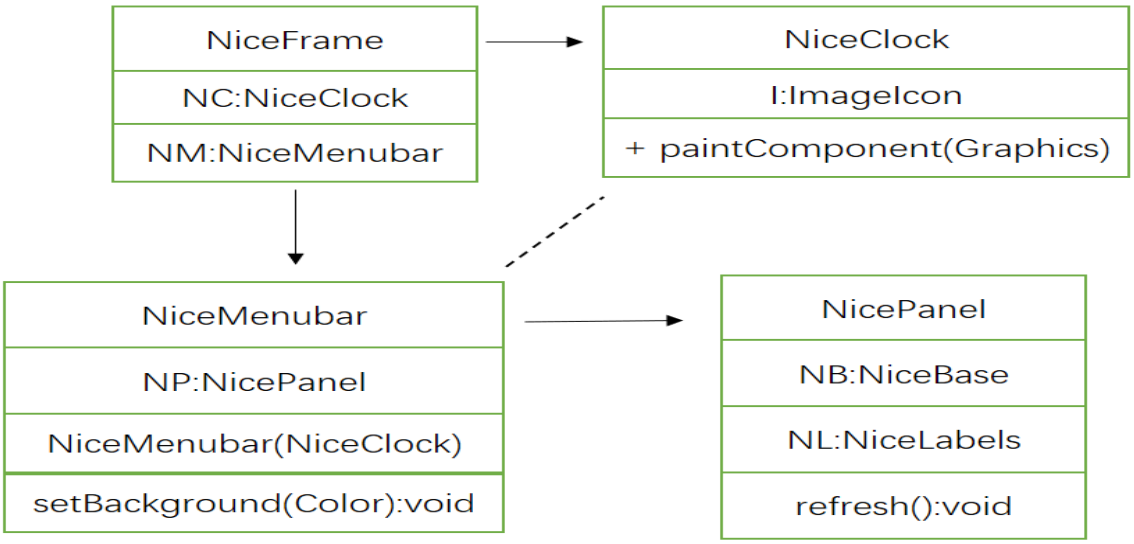

图 5.4 NiceMenubar UML 图

NiceMenubar 类，负责控制万年历所有的功能。起到承上启下的关键作用。changeSkin 方法通过设置各个按钮、标签的背景颜色完成换肤功能。其中，日历中的 42 个按钮用 for 循环遍历设置。

```
    public void changeSkin(Color C){ //set all kinds of background
        NP.NL.setBackground(C);
        NP.Button_Current.setBackground(C);
        NP.Button_Current.setBackground(C);
        NP.Button_lastYear.setBackground(C);
        NP.Button_nextYear.setBackground(C);
        NP.Button_lastMonth.setBackground(C);
        NP.Button_nextMonth.setBackground(C);
        NP.Panel_South.setBackground(C);
        NP.Panel_North.setBackground(C);
        NP.Panel_Center.setBackground(C);
        NP.Label_Year.setBackground(C);
        NP.Label_Month.setBackground(C);
```

```
            NP.Label_Calendar.setBackground(C);
            for(int i = 0; i < 42; i ++){
                NP.Button_Day[i].setBackground(C);
            }
        } //end of change skin
    } //end of class NiceMenubar
```

5.5 NicePanel 类

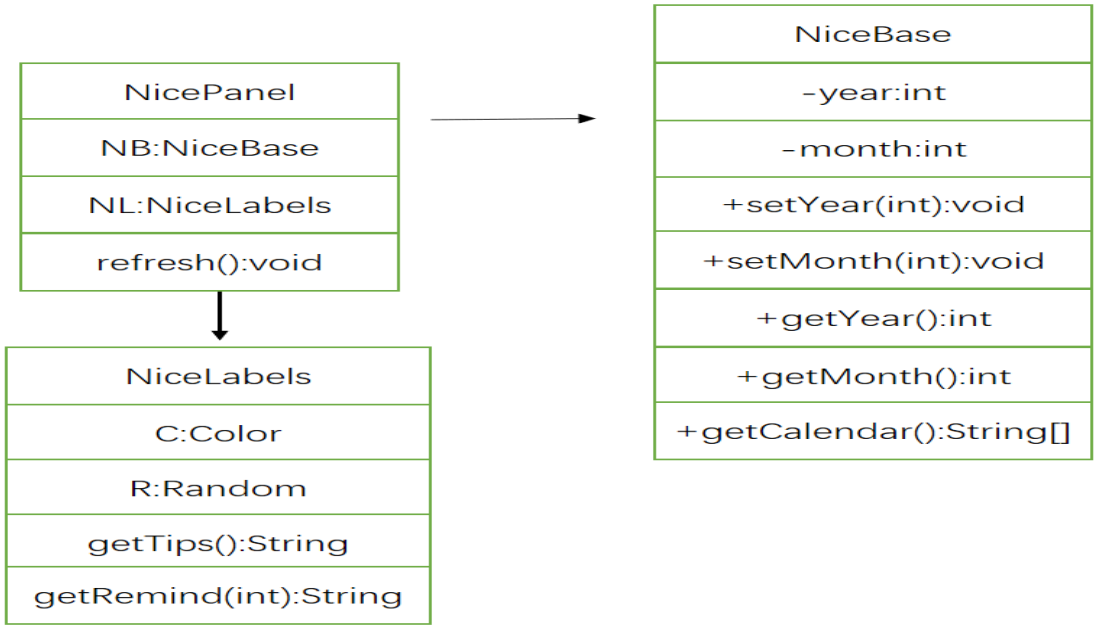

图 5.5 NicePanel UML 图

NicePanel 类，用 for 循环将按钮加入到面板里并设置按钮颜色和字体。refresh 方法用来重新获取日历并显示。

```
    NicePanel(){
        setLayout(new BorderLayout());    //border layout
        Label_Calendar.setFont(new Font("Arove", 1, 30));
        Label_Calendar.setForeground(Color.RED);
        Panel_Center.setLayout(new GridLayout(7, 7)); //7X7
        for(int i = 0; i < 7; i++){          //add week[] to panel
            Label_Week[i] = new JLabel(Week[i], JLabel.CENTER);
            Label_Week[i].setFont(new Font("Arvo", 1, 20));
            Label_Week[i].setForeground(Color.GREEN);
            Label_Week[i].setBackground(NL.C);
            Panel_Center.add(Label_Week[i]);
        }
        for(int i = 0; i < 42; i++){        //add day[] to panel
            Button_Day[i] = new JButton("None");
```

```
Button_Day[i].setBorder(BorderFactory.createLineBorder(Color.WHITE
));          //add white line to button frame
            Panel_Center.add(Button_Day[i]);
        }
}
void refresh(){                         //刷新方法
        String day[] = NB.getCalendar();
        String month;
        for(int i = 0; i<42; i++){      //重新显示日历
            Button_Day[i].setForeground(Color.CYAN);
            Button_Day[i].setFont(new Font("Arove", 1, 22));
            Button_Day[i].setText(day[i]);
        }
        switch(NB.getMonth()){ //月份要单独设置
            case 1:month = " Jan. ";break;
            case 2:month = " Feb. ";break;
            case 3:month = " Mar. ";break;
            case 4:month = " Apr. ";break;
            case 5:month = " May  ";break;
            case 6:month = " Jun. ";break;
            case 7:month = " Jul. ";break;
            case 8:month = " Aug. ";break;
            case 9:month = " Sept.";break;
            case 10:month = " Oct. ";break;
            case 11:month = " Nov. ";break;
            case 12:month = " Dec. ";break;
            default:month = " NULL ";
        }
        Label_Year.setText(" "+ NB.getYear() +" ");
        Label_Month.setText(month);    //显示
    }
```

5.6 NiceBase 类

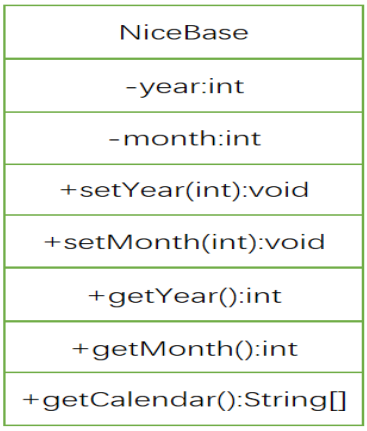

图 5.6 NiceBase UML 图

NiceBase 类，包括设置年月的方法和获取年月的方法以及获取日历按钮数字的方法。

```
public void setYear(int year){          //set year
        if(year < 2119 && year > 1919)
        this.year = year;
    }

    public int getYear(){                   //return year
        return year;
    }

    public void setMonth(int month){          //set month
        if(month < 0)
        this.month = month + 12;
        else if(month > 12)
        this.month = month -12;
        else
        this.month = month;
    }
    public int getMonth(){                  //return month
        return month;
    }

    public String[] getCalendar(){             //draw calendar
        String a[] = new String[42];
        Calendar C = Calendar.getInstance();
        C.set(year, month - 1, 1);               //the month is 0
based
        int D = C.get(Calendar.DAY_OF_WEEK)-1;

        int day = 0;
        if(month == 1||month == 3||month == 5||month == 7||month
== 8||month == 10||month == 12)
            day = 31; //1 3 5 7 8 10 12 they have 31 days a month

        else if(month == 4||month == 6||month == 9||month == 11)
            day = 30; //4 6 9 11 they have 30 days a month

        else if(month == 2){
            if(((year % 4 == 0)&&(year % 100!= 0))||(year % 400 ==
0))
```

```
            day = 29; //sometimes 29  otherwise 28
        else
            day = 28;
    }

    for(int i = D, n = 1; i < D + day; i++){
        a[i] = String.valueOf(n);
        n++;
    }

    return a;
}
```

5.7 NiceAlarm 类

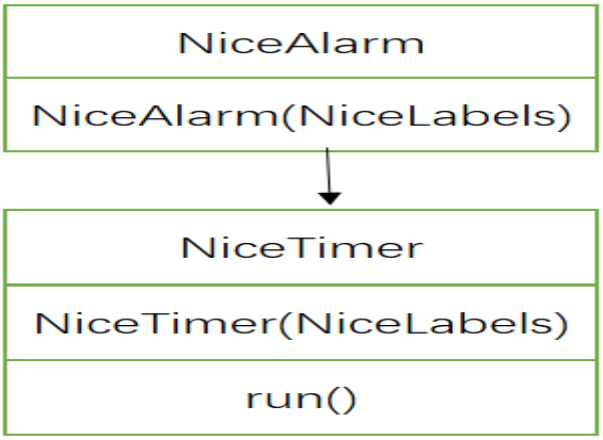

图 5.7 NiceAlarm UML 图

NiceAlarm 类，当 set 按钮被按下时，转到 NiceTImer 类去执行。

```
public void actionPerformed(ActionEvent e){

    if(e.getSource() == Textfield_H){
        NL.Label_Alarm.setText("Alarm: "+
        Textfield_H.getText() +":"+ Textfield_M.getText());

    }
    if(e.getSource() == Textfield_M){
        NL.Label_Alarm.setText("Alarm: "+
        Textfield_H.getText() +":"+ Textfield_M.getText());
    }
    if(e.getSource() == Button_Set){
        this.dispose();
        new NiceTimer(Integer.valueOf(Textfield_H.getText()),
```

```
            Integer.valueOf(Textfield_M.getText()), NL).run();
        }
    }
```

5.8 NiceTimer 类

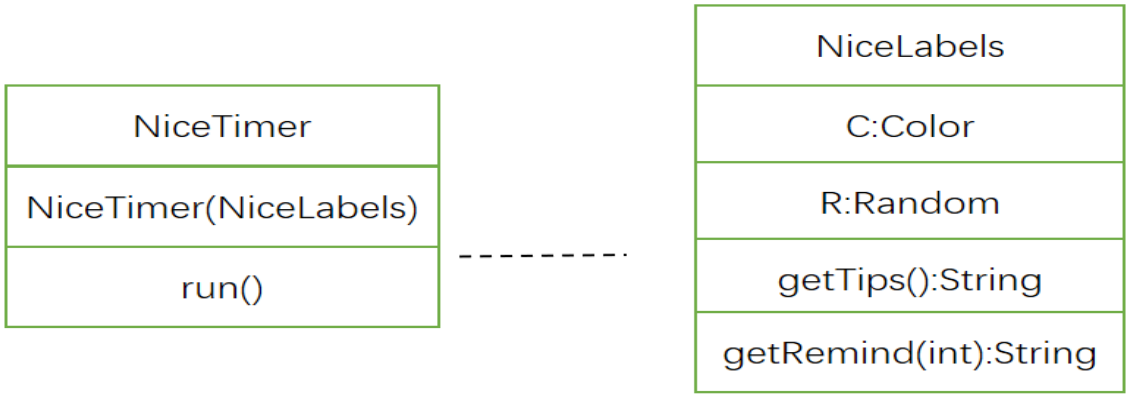

图 5.8 NiceTimer UML 图

NiceTimer 类，run 方法每秒获取一次当前时间，并与目标时间计算后得出差值显示到终端。

```
    public void run() {
        int currentSecond;
        int currentMinute;
        int currentHour;

        do{
            long currentTime = System.currentTimeMillis();
     /*get current time*/

            currentTime = currentTime / 1000;
            currentSecond = (int) (currentTime % 60); //get second
            currentTime = currentTime / 60;
            currentMinute = (int) (currentTime % 60);  //get minute
            currentTime = currentTime / 60;
            currentHour = (int) (currentTime % 24);    //get hour
            if (currentMinute >= 0)
            System.out.println("Alarm: "+ (goalHour-currentHour)
+" : "+ (goalMinute-currentMinute-1) +" : "+ (59-currentSecond));
            try {
                Thread.sleep(1000);   //system pause a second
```

```
            }
            catch (InterruptedException e){
                e.toString();
            }
    }while(goalHour*100+goalMinute >currentHour*100+currentMinute);
        NL.Label_Alarm.setText("Alarm: Time Out !"); //time out
     }
```

5.9 NiceLabels 类

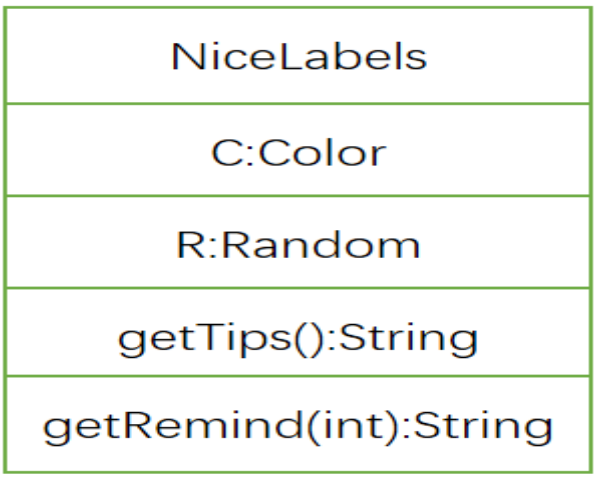

图 5.9 NiceLabels UML 图

NiceLabels 类，主要功能是设置标签内容，getTips()方法和 getRemind()方法可以返回对应日期的占卜和节日。

```
String getTips(){          //占卜提示
        int i = R.nextInt(10);
        String S[] = new String[10];
        S[0] = "Tips: Neither good nor bad";
        S[1] = "Tips: No drink/Travel/Funeral";
        S[2] = "Tips: Visit/Trade/Haircut/Trip";
        S[3] = "Tips: No felling/Marry/Bath";
        S[4] = "Tips: No pray/Build/Plant";
        S[5] = "Tips: Do nothing/Unlucky day";
        S[6] = "Tips: Pray/Marry/Bath/Trip";
        S[7] = "Tips: No marry/No swim/Decorate";
        S[8] = "Tips: No build/No trade/Mo felling";
        S[9] = "Tips: Everything goes well";
```

```
        return S[i];
    }
    void getRemind(int num){    //节日提示
        switch(num){
            case 11: Label_Remind.setText("Reminds: Jan.1st
NewYear's Day");break;
            case 214: Label_Remind.setText("Reminds: Feb.14th
Valentine's Day");break;
            case 38: Label_Remind.setText("Reminds: Mar.8th Women's
Day");break;
            case 41: Label_Remind.setText("Reminds: Apr.1st April
Fool's Day");break;
            case 422: Label_Remind.setText("Reminds: Apr.22th Earth
Day");break;
            case 51: Label_Remind.setText("Reminds: May 1st Labour
Day");break;
             case 54: Label_Remind.setText("Reminds: May 4th Youth
Day");break;
             case 57: Label_Remind.setText("Reminds: May 7th Hui's
Birthday Day");break;
            case 61: Label_Remind.setText("Reminds: June.1st
Children's Day");break;
            case 71: Label_Remind.setText("Reminds: July 1st CPC
Founding Day");break;
             case 81: Label_Remind.setText("Reminds: Aug.1st Army
Day");break;
             case 96: Label_Remind.setText("Reminds: Sept.6th My
Birthday Day");break;
            case 910: Label_Remind.setText("Reminds: Sept.10th
Teacher's Day");break;
            case 101: Label_Remind.setText("Reminds: Oct.1st
National Day");break;
            case 1128: Label_Remind.setText("Reminds: Nov.28th
Thanksgiving Day");break;
            case 1224: Label_Remind.setText("Reminds: Dec.24th
Christmas Eve");break;
            case 1225: Label_Remind.setText("Reminds: Dec.25th
Christmas Day");break;
            default: Label_Remind.setText("Reminds: NULL");
        }
    }
```

6. 调试分析

6.1 0:-1:59 问题

在计时器功能里，倒数时间显示在 terminal 终端，但是在计时结束的时候显示的不是 0 : 0 : 0，而是 0 : -1 : 59。后来找其中的原因，发现是因为：

计时函数使用的是 do-while 循环，即先输出计时数字，再判断是否计时结束。这样会导致每次计时都会多输出一秒，也就是 0 : 0 : 0 的下一秒，0 : -1 : 59。
解决办法是避免最后一轮额外循环的输出，用一个判断语句进行限定：

```
if(goalMinute-currentMinute >= 0){}
```

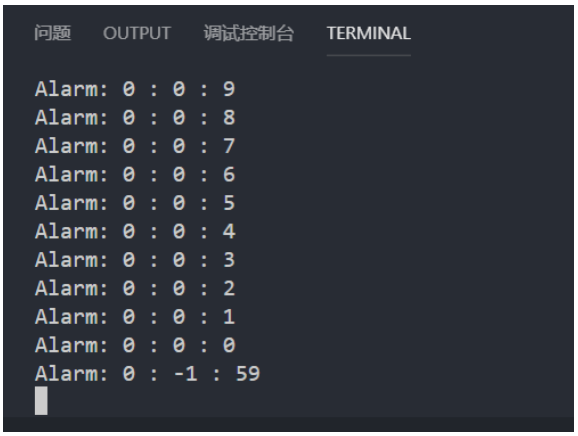

图 6.1 0:-1:59 问题

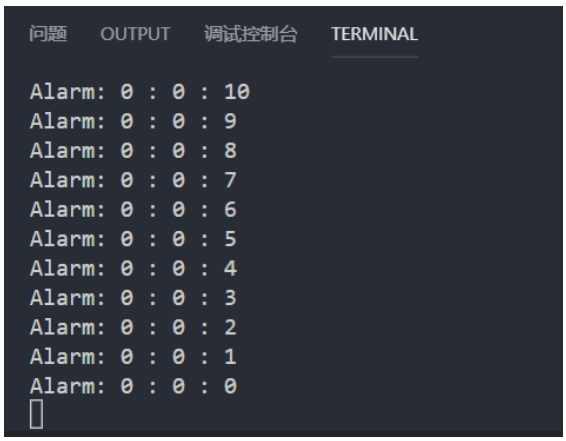

图 6.2 修复后的效果

6.2 菜单条的颜色问题

在换肤功能中，功能的实现原理是改变背景颜色。对于标签、按钮来说，使用 `setBackground()`方法即可实现，但是菜单条却不行。虽然菜单条的颜色不是很影响皮肤效果，但是若能该变，显示效果会更好。于是我在网上寻找解决办法，有的说必须重载 `void paintComponent(Graphics G)`方法、有的说可以使用 `setBackground()`但是都不合适，最终有一位网友提到了这个方法，特别有效 `getComponent().setBackground()`于是这个问题就被顺利解决了。

6.3 时差问题

在计时器功能里，为使时间同步，需要调用 System.currentTimeMillis() 方法获取当前毫秒数，然后对其进行解析，方法如下：

```
currentTime = currentTime / 1000;
currentSecond = (int) (currentTime % 60); //get second
currentTime = currentTime / 60;
currentMinute = (int) (currentTime % 60);  //get minute
currentTime = currentTime / 60;
currentHour = (int) (currentTime % 24); //get hour
```

但是，据此得到的小时数却有问题，经过几轮调试后发现，总是与当前时间相差 8 个小时。后来听过思考，应该是因为北京时间处于东八区，而获取的时间是标准时间。解决办法是将用户输入的时间减去 8，换算成标准时间再使用。

6.4 功能补充

如果有时间的话，应当再添加一个可以让用户自定义日期提醒的功能，这样用户就可以添加生日、会议等自定义事项，人机交互更加自由。

还要有登录功能，搭配数据库，可以实时保存用户自定义提醒事项和闹钟的数据。

7. 测试结果

7.1 跳转功能

Fuction 菜单下有闹钟、跳转、上一年、下一年、上个月、下个月等功能。

测试 Function 菜单下的 Skip 功能，跳转到 2022 年 10 月 1 日。

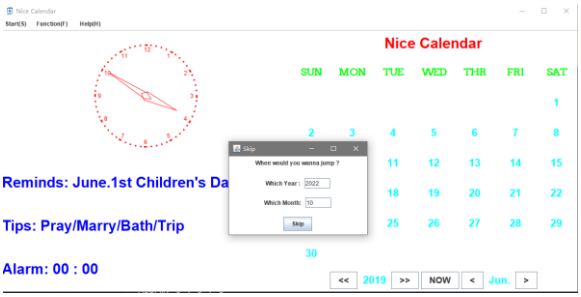

图 7.1 跳转前，窗口输入

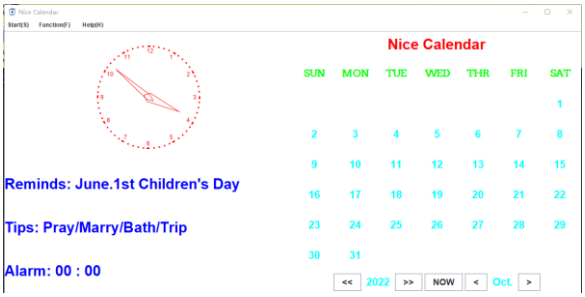

图 7.2 跳转后，日期改变

7.2 换肤功能

测试 Start 菜单下的 Skin 功能。

有 QuietWhite、SoftYellow、PureBlack、VesperalDark 四种皮肤可选，选择 VesperalDark 和 PureBlack 进行测试。

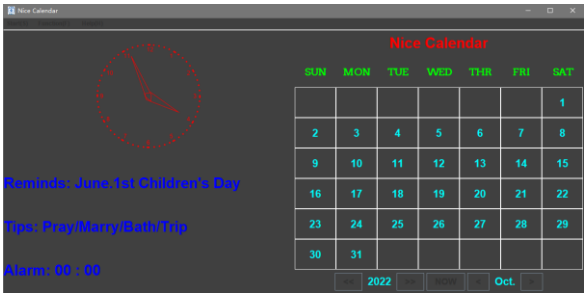

图 7.3 VersperalDark

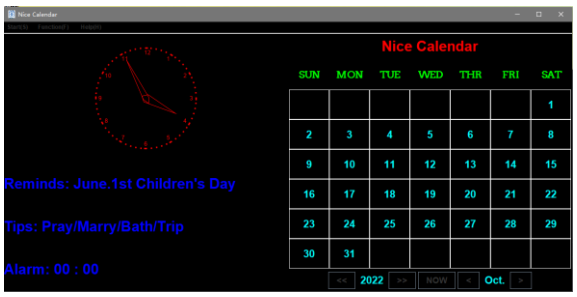

图 7.4 PureBlack

7.3 定时功能

测试 Function 菜单下的 Alarm 功能。当前时间 15：40， 定一个闹钟，15：43 提醒。

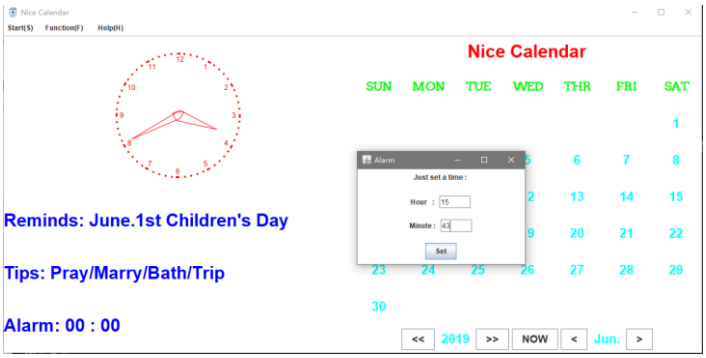

图 7.5 定时前，窗口输入

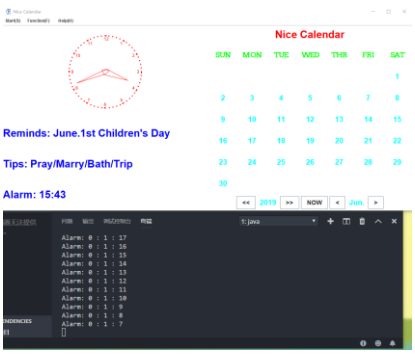

图 7.6 运行时 终端显示倒计时

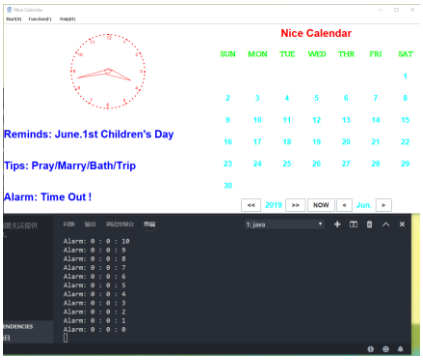

图 7.7 时间到，显示 Time Out！

7.4 显示 About

Help 菜单提供关于开发者的一些信息。其中 Readme 显示软件的使用说明、About 显示开发者的个人信息。

测试 Help 菜单下的 About 提示

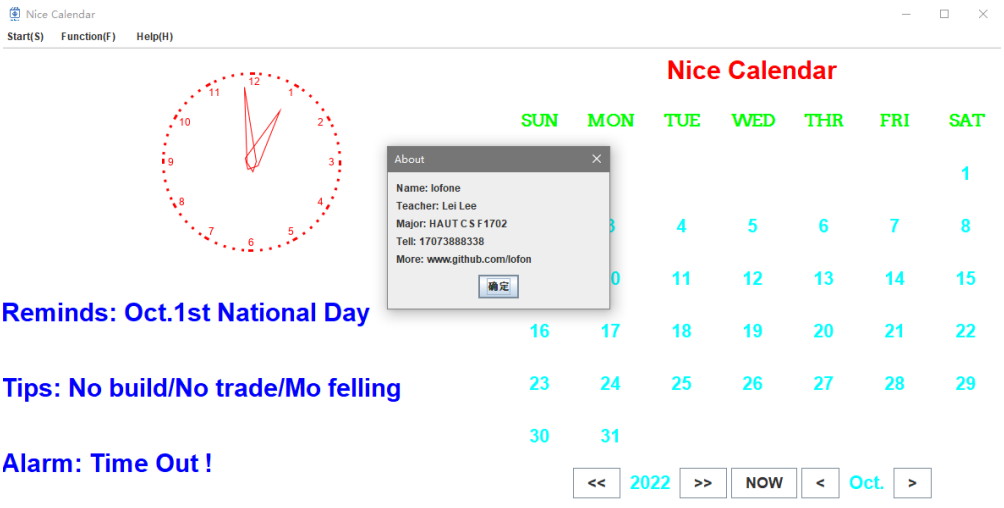

图 7.8 关于开发者

参考文献

[1] 赵满来.可视化 Java_GUI 程序设计实验指导[M]. 清华大学出版社，2016
[2] 耿祥义.Java 2 实用教程(第 5 版)[M]. 清华大学出版社,2017.
[3] 耿祥义.Java 课程设计(第 3 版)[M]. 清华大学出版社,2018 .
[4] 林智杨、范明翔等. 深入浅出 Java Swing 程序设计[M]. 中国铁道出版社,2005
[5]王晓哲.Java Swing 组件技术[J].天津职业院校联合学报,2008(03):138-142+145.
[6]潘国荣.Java 中的常见事件及处理探究[J].电脑知识与技术,2018,14(29):125-126+131.

心得体会

两周的时间能够开发一个小程序还是非常开心的。从这次课程设计中我深切地体会到需求分析的重要性。因为早期没有做好需求分析，代码被改了一遍又一遍，枉费心机。

早期第一个版本的界面是这样的，只有一个日历功能，而且仅仅能显示日期和跳转到指定日期。

图 8.1 Calendar1.0

后来感觉功能太弱了，就加了一个表盘，将界面做地扁平化，诞生了第二个版本，效果是这样的：

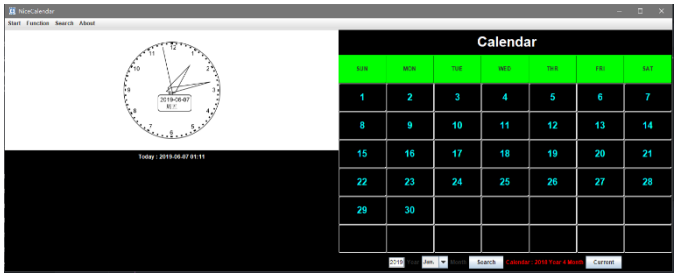

图 8.2 Calendar2.0

但是这样功能依旧很少，而且没有给用户一种交互的感觉，于是就添加了显示节日的功能。顺便把排版布局重新设定，诞生了第三个版本。可以看出，效果好于之前太多：

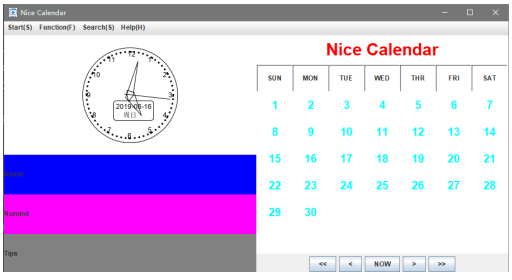

图 8.3 Calendar3.0

但是这样看来颜色并不友好，反倒是给人一种很突兀的感觉。为了实现换肤功能，将界面颜色统一，优化了类之间的组合关系，做出了这个 Calendar Beta 版本：

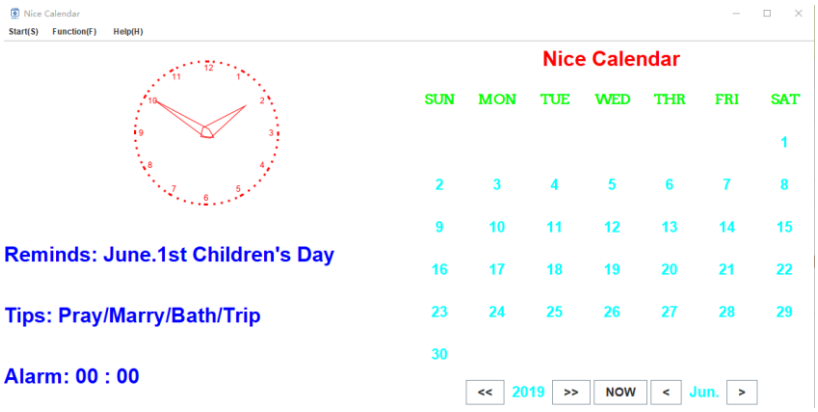

图 8.4 Calendar Beta

界面进一步显得简洁大方，朴素高雅，排版也更加合理，功能也趋于完善，就我本人来说非常满意。虽然系统已经有了很多功能，但是实现让用户添加自定义提醒却没有实现。因为时间关系，不敢再对代码做大的改动。当代码写到上千行的时候，每添加一个功能，就得考虑很多因素。

倘若需求分析能够做地完善，那么在实现的时候简直就是事半功倍。由于早期考虑不周，导致一遍遍地修改程序甚至推翻重写。做系统不能仅仅局限于眼前某一个小功能如何实现，而应该是统筹全局，合理安排，不然肯定是拆东墙补西墙，这是我最大的体会。

通过此次课程设计，我对 swing 的设计方法更加了解。深入理解了组件之间的堆叠方式、事件的处理方式。还学会使用很多实用库比如：

java.util.Calendar、**java.awt.geom.AffineTransform** 等。

Swing 更强大的地方在于绘图。重载 **paintComponent(Graphics G)**方法可以使用 Graphics 类绘制各种图形。

课程设计让我真正体会到了 Java 强大的地方，有丰富的第三方库可以嗲用，从而做出绚丽的 UI 界面，而不用再对着终端的黑白框跑程序。我也学习到了开发一套程序的基本流程：需求分析、概要设计、详细设计……而不是撸起袖子就开始写代码。总之，获益匪浅，收益颇丰。