目录

[第1章 万年历 1](#_Toc29403)

[1.1 万年历介绍 1](#_Toc22645)

[1.2 程序设计思路 1](#_Toc18045)

[1.3 关键技术 2](#_Toc31374)

[1.3.1 确定指定月份的天数 2](#_Toc5065)

[1.3.2 确定每月1号是星期几 3](#_Toc22525)

[1.3.3 计算农历年 4](#_Toc2851)

[1.4 程序设计步骤 5](#_Toc9248)

[1.4.1 主界面设计 5](#_Toc27853)

[1.4.2 给组件初始化数据 9](#_Toc5892)

[1.4.3 给组合框添加监听事件 12](#_Toc3605)

[1.4.4 显示选中日期及农历年 13](#_Toc112)

[1.4.5 给天数按钮添加监听事件 13](#_Toc14505)

[1.4.6 设置相关日期的颜色 17](#_Toc24656)

[1.4.7 返回今日 22](#_Toc15828)

# 万年历

## 万年历介绍

本案例实现一个简单的万年历,如图所示:



通过组合框可以选择查看任意一年,任意一个月的日历,通过单击返回今日按钮可以返回到当前日历, 在界面下面显示选择的日期及当前年的农历形式.

## 程序设计思路

本案例基于Swing组件实现, 需要使用JPanel面板,JLable标签, JButton按钮,JComboBox组合框等组件,如图所示.



需要给组合框添加ItemListener监听器,当修改组合框中的选择项时,触发JComboBox组合框的itemStateChanged事件

给按钮添加ActionListener监听器,当单击按钮时,触发按钮的actionPerformed事件

## 关键技术

### 确定指定月份的天数

一年12个月分大小月,大月31天,小月30天,2月的天数还跟年有关,如果是闰年则29天,平年28天.

|  |
| --- |
| //返回指定年和月的天数  public int getMonthDays(int year, int month) {  switch (month) {  case 1:  case 3:  case 5:  case 7:  case 8:  case 10:  case 12:  return 31;  case 4:  case 6:  case 9:  case 11:  return 30;  case 2:  return isLeay(year) ? 29 : 28;  }  return 0;  }  //判断闰年  public boolean isLeay(int year) {  return year % 4 == 0 && year % 100 != 0 || year % 400 == 0;  } |

### 确定每月1号是星期几

已知1900-1-1是星期一,计算year-month-day日期是星期几,可以从1900-1-1开始累加到year-month-day日期的总天数,总天数对7求余,余数为0表示星期日,余数1表示星期一,余数2表示星期二,....余数6表示星期六.

|  |
| --- |
| //返回指定日期是星期几  public int getWeekDay(int year, int month, int day) {  int sum = 0;  //先累加整年的天数,闰年366天,平年365天  for (int i = 1900; i < year; i++) {  sum += isLeay(i) ? 366 : 365;  }  //再累加当前year这一年,整月的天数  for (int i = 1; i < month; i++) {  sum += getMonthDays(year, i);  }  //最后累加year年month月的天数  sum += day;  //总天数对7求余  return sum % 7;  } |

### 计算农历年

中国古代纪年不用数字，而是采用天干地支组合。

天干有十个，分别是：甲、乙、丙、丁、戊、己、庚、辛、壬、癸；

地支有十二个，分别是：子、丑、寅、卯、辰、巳、午、未、申、酉、戌、亥。

使用时天干地支各取一字，天干在前，地支在后，组合成干支，例如甲子、乙丑、丙寅等等，依次轮回可形成六十种组合，以这些天干地支组合纪年，每六十年一个轮回，称为一个甲子。

十二属相又称“十二生肖”，由十一种源自自然界的动物：鼠、牛、虎、兔、蛇、马、羊、猴、鸡、狗、猪以及传说中的龙组成，用于纪年时，按顺序和十二地支组合成子鼠、丑牛、寅虎、卯兔、辰龙、巳蛇、午马、未羊、申猴、酉鸡、戌狗和亥猪。

天干地支以及十二生肖常组合起来描述农历年，比如公历2011年就是农历辛卯兔年、2012年是壬辰龙年等等。

计算某一年的天干地支，有很多经验公式，如果知道某一年的天干地支，也可以直接推算其它年份的天干地支。举个例子，如果知道2000年是庚辰龙年，则2012年的干支可以这样推算：（2012-2000）% 10=2，2012年的天干就是从庚开始向后推2个天干，即壬；2012年的地支可以这样推算：（2012 - 2000）% 12 = 0，2012年的地支仍然是辰，因此2012年的天干地支就是壬辰，十二生肖龙年。对于2000年以前的年份，计算出年份差后只要将天干和地支向前推算即可。

|  |
| --- |
| public String getGanZhiNian(int year) {  //已知1900年是庚子鼠年  //干支  String[] ganzhi = {"甲", "乙", "丙", "丁", "戊", "己", "庚", "辛", "壬", "癸"};  //地支  String[] dizhi = {"子", "丑", "寅", "卯", "辰", "巳", "午", "未", "申", "酉", "戌", "亥"};  //十二生肖  String[] animals = {"鼠", "牛", "虎", "兔", "龙", "蛇", "马", "羊", "猴", "鸡", "狗", "猪"};  //计算指定年的干支, 1900年干支庚的索引值是6  int gan = ((year - 1900) % 10 + 6) % 10;  //计算地支, 1900年地支子的索引值是0  int di = (year - 1900) % 12;  //返回天干地支+生肖组合  return ganzhi[gan] + dizhi[di] + "[" + animals[di] + "]年";  } |

## 程序设计步骤

### 1.4.1 主界面设计

|  |
| --- |
| package mycalendar;  import javax.swing.\*;  import javax.swing.border.EmptyBorder;  import java.awt.\*;  /\*\*  \* 定义日历窗口  \* 北京动力节点老崔  \*/  public class CalendarFrame extends JFrame {  //定义需要的组件  private JComboBox yearComboBox; //年组合框  private JComboBox monthComboBox; //月组合框  private JButton returnTodayBtn; //返回今日按钮  private JLabel [] weekLbls; //日历表头星期几标签  private JButton[] dayBtns; //日历中天数按钮  private JLabel selectedDateLbl; //显示选中日期  private JLabel chineseYearLbl; //显示农历年  //通过构造方法给组件初始化,并添加到框架中  public CalendarFrame(){  //组件初始化  yearComboBox = new JComboBox(); //年组合框  monthComboBox = new JComboBox(); //月组合框  returnTodayBtn = new JButton(" 返回今日 "); //返回今日按钮  //当月日历表头标签  weekLbls = new JLabel[7]; //数组初始化  String[] week = { "星期一", "星期二", "星期三", "星期四", "星期五", "星期六","星期日"};  //给数组元素赋值  for (int i = 0; i < weekLbls.length; i++) {  weekLbls[i] = new JLabel( week[i]);  weekLbls[i].setHorizontalAlignment(JLabel.CENTER); //标签内容水平居中  }  //当月天数按钮数组初始化  dayBtns = new JButton[42];  for (int i = 0; i < dayBtns.length; i++) {  dayBtns[i] = new JButton();  }  selectedDateLbl = new JLabel("选中的日期");  chineseYearLbl = new JLabel("农历年");  //把组件添加到框架中  //把年组合框,月组合框,返回今日按钮放在一个Panel面板中  JPanel yearMonthPanel = new JPanel();  yearMonthPanel.add(yearComboBox);  yearMonthPanel.add(new JLabel(" 年 "));  yearMonthPanel.add(monthComboBox);  yearMonthPanel.add(new JLabel(" 月 "));  yearMonthPanel.add(returnTodayBtn);  //把面板添加到框架中  this.add(yearMonthPanel, BorderLayout.NORTH);  //把本月日历显示的星期及日期放在一个Panel中  JPanel dayPanel = new JPanel();  //设置为网格布局,有7行7列  dayPanel.setLayout(new GridLayout(7,7,8,8));  //设置Panel的边框  dayPanel.setBorder(new EmptyBorder(5,10,5,10));  //向dayPanel中添加表头  for (int i = 0; i < weekLbls.length; i++) {  dayPanel.add(weekLbls[i]);  }  //向dayPanel添加天数按钮  for (int i = 0; i < dayBtns.length; i++) {  dayPanel.add(dayBtns[i]);  }  //把daypanel添加框架中  this.add(dayPanel,BorderLayout.CENTER);  //把选择当前日期标签与农历年标签添加到框架中  JPanel soutPanel = new JPanel();  soutPanel.add(selectedDateLbl);  soutPanel.add(chineseYearLbl);  this.add(soutPanel, BorderLayout.SOUTH);  //设置框架的相关属性  //设置标题  this.setTitle("日历");  //设置大小  this.setSize(600, 360);  //设置不可调整大小  this.setResizable(false);  //在窗口中间显示  this.setLocationRelativeTo(null);  //关闭窗口退出程序  this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  //显示窗口  this.setVisible(true);  }  } |

运行结果如下



### 1.4.2 给组件初始化数据



|  |
| --- |
| //创建日历对象  private Calendar calendar = Calendar.getInstance();  private int currentYear = calendar.get(Calendar.YEAR); //当前年  private int currentMonth = calendar.get(Calendar.MONTH); //当前月(从0开始)  private int currentDay = calendar.get(Calendar.DAY\_OF\_MONTH); //当前日  private int max\_year = currentYear + 100; //日历的最大年份  //加载初始数据  private void loadInitialData() {  //给年组合框添加初始数据  for(int i = 1900; i < max\_year; i++ ){  yearComboBox.addItem( i + "");  }  //设置年组合框的默认值为当前年  yearComboBox.setSelectedItem(currentYear + "");  //给月组合框添加初始数据  for( int i = 1; i <= 12; i++){  monthComboBox.addItem(i + "");  }  //设置月组合框的默认值为当前月  monthComboBox.setSelectedIndex(currentMonth);  //设置日历中天数按钮显示的文本  setDay();  }  //设置日历按钮显示的文本  private void setDay() {  //获得当前选中的年份与月份  Object selectYearStr = yearComboBox.getSelectedItem();  Object selectMonthStr = monthComboBox.getSelectedItem();  //把年份与月份转换为整数  int selectYear = Integer.parseInt((String)selectYearStr);  int selectMonth = Integer.parseInt((String)selectMonthStr);  //计算本月1号是星期几  int weekday = getWeekDay(selectYear, selectMonth,1);  //获得上个月的天数  int lastMonthDays = 0 ;  if ( selectMonth > 1) { //当前月份不是1月,计算当前年上个月的天数  lastMonthDays = getMonthDays(selectYear, selectMonth - 1);  }else { //当前月份是1月,则计算去年12月份天数  lastMonthDays = getMonthDays(selectYear-1, 12);  }  //获得当前月的天数  int monthDays = getMonthDays(selectYear, selectMonth);  //设置第一行前面上个月的按钮文本  int lastDays = weekday == 0 ? 7 : weekday; //打印的个月的天数  int day = lastMonthDays - lastDays + 2;  for( int i = 1; i < lastDays ; i++ ){  dayBtns[i-1] .setText( day + "");  day++;  }  //当前月按钮的文本  day = 1;  for( int i = lastDays ; i < lastDays + monthDays; i++){  dayBtns[i-1] .setText( day + "");  day++;  }  //下个月的按钮文本  day = 1;  for( int i = lastDays + monthDays; i <= dayBtns.length; i++){  dayBtns[i-1] .setText( day + "");  day++;  }  } |
| public CalendarFrame(){  ......  //加载初始化数据  **loadInitialData();**  } |

### 1.4.3 给组合框添加监听事件

|  |
| --- |
| public class CalendarFrame extends JFrame **implements ActionListener** {  //定义需要的组件  .....................  //通过构造方法给组件初始化,并添加到框架中  public CalendarFrame(){  ......  //加载初始化数据  loadInitialData();  **//添加组合框监听事件**  **yearComboBox.addActionListener(this::actionPerformed);**  **monthComboBox.addActionListener(this::actionPerformed);**  } |
| @Override  public void actionPerformed(ActionEvent e) {  if (e.getSource() == yearComboBox){  setDay();  }else if ( e.getSource() == monthComboBox){  setDay();  }  } |
|  |

### 1.4.4 显示选中日期及农历年

在CalendarFrame日历类中增加一个字段,表示当前日期的天

private int selectedDay; //选中的天

在加载默认数据private void loadInitialData() 方法中给selecteDay默认初始化为当前日期的天

selectedDay = currentDay; //默认当天的日子就是选中的日子

在 private void setDay() 方法中,调用自定义方法 freshSouthDate刷新显示当前日期

freshSouthDate()方法的定义如下

|  |
| --- |
| //刷新当前选中的日期  private void freshSouthDate(){  int selectedYear = Integer.parseInt(yearComboBox.getSelectedItem().toString());  selectedDateLbl.setText( selectedYear + "年" + monthComboBox.getSelectedItem() + "月" + selectedDay + "日 " + getGanZhiNian( selectedYear));  } |

### 1.4.5 给天数按钮添加监听事件

在日历中单击某一天,则在当前日期中动态更新你选择的日期



需要给每个按钮添加监听事件

|  |
| --- |
| public CalendarFrame(){  ...........  //当月天数按钮数组初始化  dayBtns = new JButton[42];  for (int i = 0; i < dayBtns.length; i++) {  dayBtns[i] = new JButton();  //给按钮添加监听  **dayBtns[i].addActionListener(this::actionPerformed);**  } |
| 还需要区别这个日期是单击的上个月,当前月,还是下个月的日期, 如果单击的是上个月的日期,需要把月份切换到上个月,单击的是下个月的日期把月份切换到下个月. 为每个按钮设置ActionCommand进行区分选中的是哪个月的日期 |
| private void setDay() {  ...........  //设置第一行前面上个月的按钮文本  int lastDays = weekday == 0 ? 7 : weekday; //打印的个月的天数  int day = lastMonthDays - lastDays + 2;  for( int i = 1; i < lastDays ; i++ ){  dayBtns[i-1] .setText( day + "");  **dayBtns[i-1].setActionCommand("P" + day); //上个月**  day++;  }  //当前月按钮的文本  day = 1;  for( int i = lastDays ; i < lastDays + monthDays; i++){  dayBtns[i-1] .setText( day + "");  **dayBtns[i-1].setActionCommand("C" +day); //当前月**  day++;  }  //下个月的按钮文本  day = 1;  for( int i = lastDays + monthDays; i <= dayBtns.length; i++){  dayBtns[i-1] .setText( day + "");  **dayBtns[i-1] .setActionCommand("N" + day); //下个月**  day++;  } |
|  |

修改actionPerformed()事件处理方法

|  |
| --- |
| @Override  public void actionPerformed(ActionEvent e) {  if (e.getSource() == yearComboBox){  setDay();  }else if ( e.getSource() == monthComboBox){  setDay();  }else {  //获得按钮的ActionCommand  JButton selectBtn = (JButton) e.getSource();  String actionCommand = selectBtn.getActionCommand();  //取出 月份 标志  String flag = actionCommand.substring(0,1);  //取出当前日期  String dayStr = actionCommand.substring(1);  //设置选中的当前日期  selectedDay = Integer.parseInt(dayStr);  int selecteYear = Integer.parseInt( yearComboBox.getSelectedItem().toString()); //组合框中的年  int selectedMonth = Integer.parseInt( monthComboBox.getSelectedItem().toString()); //组合框中的月  if ("P".equalsIgnoreCase(flag) ){  if ( selectedMonth > 1 ){  //切换到当前年的上个月  monthComboBox.setSelectedItem( selectedMonth - 1 + "");  }else{  //切换到上一年的12月  if ( selecteYear > 1900 ){  yearComboBox.setSelectedItem( selecteYear - 1 + "");  monthComboBox.setSelectedItem(12 + "");  }else {  //已经是第一年了, 无法切换到上一年  }  }  }else if ("N".equalsIgnoreCase(flag)){  if (selectedMonth < 12){  //切换到下个月,1月的索引值为0  monthComboBox.setSelectedIndex( selectedMonth );  }else{ //当前是12月,切换到下一年的1月  if ( selecteYear < max\_year ){  yearComboBox.setSelectedItem( selecteYear + 1 + "");  monthComboBox.setSelectedIndex(0);  }  }  }else { //当前月  freshSouthDate();  }  }  } |

### 1.4.6 设置相关日期的颜色

上个月的日历及下个月的日历显示为灰色,

|  |
| --- |
| private void setDay() {  ...............  //设置第一行前面上个月的按钮文本  int lastDays = weekday == 0 ? 7 : weekday; //打印的个月的天数  int day = lastMonthDays - lastDays + 2;  for( int i = 1; i < lastDays ; i++ ){  dayBtns[i-1] .setText( day + "");  dayBtns[i-1].setActionCommand("P" + day);  **dayBtns[i-1].setForeground(Color.GRAY);**  **dayBtns[i-1].setBackground(null);**  day++;  }  //当前月按钮的文本  day = 1;  for( int i = lastDays; i < lastDays + monthDays; i++){  dayBtns[i-1] .setText( day + "");  dayBtns[i-1].setActionCommand("C" + day);  **dayBtns[i-1].setForeground(Color.BLACK);**  **dayBtns[i-1].setBackground(null);**  **if ( day == selectedDay){**  **dayBtns[i-1].setForeground(Color.BLUE);**  **dayBtns[i-1].setBackground(Color.CYAN);**  **}**  day++;  }  //下个月的按钮文本  day = 1;  for( int i = lastDays + monthDays; i <= dayBtns.length; i++){  dayBtns[i-1] .setText( day + "");  dayBtns[i-1] .setActionCommand("N" + day);  **dayBtns[i-1].setForeground(Color.GRAY);**  **dayBtns[i-1].setBackground(null);**  day++;  }  //显示选中日期  freshSouthDate();  } |

周六,周日字体显示为红色,

|  |
| --- |
| **//定义计数器变量,统计天数按钮的数量, 周六,周日的字段显示为红色**  **int count = 0 ;**  //设置第一行前面上个月的按钮文本  int lastDays = weekday == 0 ? 7 : weekday; //打印的个月的天数  int day = lastMonthDays - lastDays + 2;  for( int i = 1; i < lastDays ; i++ ){  dayBtns[i-1] .setText( day + "");  dayBtns[i-1].setActionCommand("P" + day);  dayBtns[i-1].setForeground(Color.GRAY);  dayBtns[i-1].setBackground(null);  day++;  **count++; //计数**  }  //当前月按钮的文本  day = 1;  for( int i = lastDays; i < lastDays + monthDays; i++){  dayBtns[i-1] .setText( day + "");  dayBtns[i-1].setActionCommand("C" + day);  dayBtns[i-1].setForeground(Color.BLACK);  dayBtns[i-1].setBackground(null);  **count++; //计数**  **//设置 周六,周日的字体颜色**  **if ( count % 7 == 6 || count % 7 == 0 ){**  **dayBtns[i-1].setForeground(Color.RED);**  **dayBtns[i-1].setBackground(null);**  **}**  //设置当前日期的颜色  if ( day == selectedDay){  dayBtns[i-1].setForeground(Color.BLUE);  dayBtns[i-1].setBackground(Color.CYAN);  }  day++;  }  //下个月的按钮文本  day = 1;  for( int i = lastDays + monthDays; i <= dayBtns.length; i++){  dayBtns[i-1] .setText( day + "");  dayBtns[i-1] .setActionCommand("N" + day);  dayBtns[i-1].setForeground(Color.GRAY);  dayBtns[i-1].setBackground(null);  day++;  **count++;**  } |

选中的当前日期显示为蓝色

|  |
| --- |
| //保存上次选择的按钮,及按钮的前景色与背景色  private JButton oldSelectedBtn = null;  private Color oldForeColor = null ;  private Color oldBackColor = null; |
| //设置日历按钮显示的文本  private void setDay() {  ............  //当前月按钮的文本  day = 1;  for( int i = lastDays; i < lastDays + monthDays; i++){  dayBtns[i-1] .setText( day + "");  dayBtns[i-1].setActionCommand("C" + day);  dayBtns[i-1].setForeground(Color.BLACK);  dayBtns[i-1].setBackground(null);  count++; //计数  //设置 周六,周日的字体颜色  if ( count % 7 == 6 || count % 7 == 0 ){  dayBtns[i-1].setForeground(Color.RED);  dayBtns[i-1].setBackground(null);  }  //设置当前日期的颜色  if ( day == selectedDay){  **//保存选中按钮,及选中按钮原来的颜色**  **oldSelectedBtn = dayBtns[i-1]; //设置旧选中按钮**  **oldForeColor = dayBtns[i-1].getForeground();**  **oldBackColor = dayBtns[i-1].getBackground();**  //设置新的颜色  dayBtns[i-1].setForeground(Color.BLUE);  dayBtns[i-1].setBackground(Color.CYAN);  }  day++;  }  ..........  } |
| public void actionPerformed(ActionEvent e) {  if (e.getSource() == yearComboBox){  setDay();  }else if ( e.getSource() == monthComboBox){  setDay();  }else {  //获得按钮的ActionCommand  JButton selectBtn = (JButton) e.getSource();  **//还原旧按钮的颜色**  **oldSelectedBtn.setForeground(oldForeColor);**  **oldSelectedBtn.setBackground(oldBackColor);**  **//保存新选中的按钮**  **oldSelectedBtn = selectBtn;**  **oldBackColor = selectBtn.getBackground();**  **oldForeColor = selectBtn.getForeground();**  **//设置新选中按钮的颜色**  **selectBtn.setForeground(Color.BLUE);**  **selectBtn.setBackground(Color.CYAN);** |

### 1.4.7 返回今日

单击返回今日按钮,把日历还原为当前日期,首先给返回今日按钮添加监听

|  |
| --- |
| public CalendarFrame(){  .........  returnTodayBtn = new JButton(" 返回今日 "); //返回今日按钮  **//添加监听事件**  **returnTodayBtn.addActionListener(this::actionPerformed);** |
| @Override  public void actionPerformed(ActionEvent e) {  if (e.getSource() == yearComboBox){  setDay();  }else if ( e.getSource() == monthComboBox){  setDay();  }**else if (e.getSource() == returnTodayBtn){**  **selectedDay = currentDay;**  **yearComboBox.setSelectedItem(currentYear + "");**  **monthComboBox.setSelectedIndex(currentMonth);**  **}**else { |