第3章　选择结构

任务　分时问候

3.1　任务描述

程序运行时通常是按从上到下的顺序执行的，但是有时程序会根据不同情况，或是选择不同的语句块来执行，或是反复运行某一个语句块，或是跳转到某一语句块执行，以上几种运行方式在程序中被称为“程序流程控制”。Java 语言中的流程控制有选择（分支）结构、循环结构和跳转语句 3 种。本章的任务是通过选择（分支）结构，编写分时问候的机器人程序，我们不妨给机器人起名为“小二”，当程序运行后，机器人“小二”会根据操作系统当前的时间给出不同的问候语。

3.2　任务分析

按照人们的生活习惯，一般粗略地把一天分为如表 3-1 所示几个时间段。

表 3-1　一天中的几个时间段

时间段	[0,6)	[6,9)	[9,12)	[12,14)	[14,17)	[17,19)	[19,22)	[22,24)
含义	凌晨	早晨	上午	中午	下午	傍晚	晚上	深夜

通过表中的时间段划分，可以发现，在不同的时间运行程序，程序所处的时间段不同，所要给出的问候也不同，也就是存在多种选择或是分支情况，因此只使用顺序结构的程序方式是不能完成任务的。在 Java 语言中，可以使用 if 结构、if-else 结构、多重 if-else 结构、嵌套 if-else 结构或 switch 结构来实现选择（分支）程序。

3.3　相关知识

选择结构又称为分支结构，意为“如果……则……”，根据条件的成立与否，再决定执行哪些语句的一种程序结构，选择（分支）结构分为简单 if 结构、if-else 结构、if-else if 多重结构等，下面对几类分支结构一一进行介绍。

3.3.1　简单 if 结构

基本 if 选择结构是单分支结构，是对某种条件做出相应的处理，表现含义为“如果……那么……”。if 结构的语法格式如下：

```
if(判断条件){
    [代码块]
}
```

简单 if 结构如图 3-1 所示。

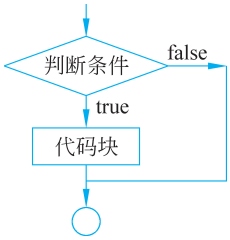


图 3-1　简单 if 结构

判断条件必须是一个布尔表达式，一旦条件中的值为 true 就执行代码块。

注意

代码块可以是一条语句，也可以是多条语句。

代码块是可选参数。当判断条件为 true 时执行代码块。当代码块省略时，可以保留花括号，也可以省略花括号，并在 if 语句的末尾添加分号“;”，如果该语句块只有一条语句，花括号也可以省略不写，但是在编写程序过程中，为了增强程序的可读性和可维护性最好不要省略。下面的代码都是正确的。

(1) `if(时间在 6 点前);`

(2)
```
if(时间在 6 点前)
    问候早晨好;
```

(3)
```
if(时间在 6 点前){
    问候早晨好;
    }
```

注意

代码(3)是较好的程序，读者在学习和编写程序时要注意一个原则：程序是人编写来操作计算机的，要让程序正常运行，就要遵循程序的语法，但是程序不可能只运行一次，在使用过程中要进行维护，维护人可能是自己，也可能是别人，因此所编写的程序也要让人能快速清晰看懂，要遵循和养成良好的编程习惯。

【例 3-1】 比较两个整数的大小，并输出比较结果。

【程序实现】

```
public class CompareTwoIntegers {
    public static void main(String[] args) {
        int num1=10,num2=11;                    //定义两个整数并赋值
        if(num1<num2){                          //比较两个值的大小
            System.out.println("num1 小于 num2");
        }
    }
}
```

【运行结果】

```
num1 小于 num2
```

3.3.2　if-else 结构

if-else 结构是条件语句的一种最通用的形式，else 是可选的，表现的含义为“如果……那么……否则……”。if-else 结构的语法格式如下：

```
if(判断条件){
    [代码块 1]
}else{
    [代码块 2]
}
```

if-else 结构的程序结构如图 3-2 所示。

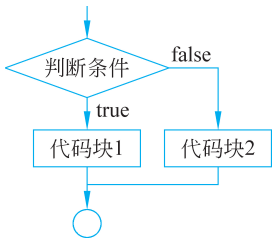


图 3-2　if-else 结构

注意

语法格式中的[代码块 1],[]中的内容为可选参数。

【例 3-2】 根据给定的二月份的天数,判断今年是平年还是闰年。

【问题分析】

按平年与闰年的定义,平年的二月份有 28 天,闰年的二月份有 29 天,因此若给定的二月份的天数为 29 天,则为闰年,其他则认为是平年。

【程序实现】

```
import java.util.Scanner;
public class LeapOrNotByFebDays {
    public static void main(String[] args) {
        System.out.println("请输入二月份的天数(29天【是闰年】或其他值【认为是平年】): ");
        Scanner input=new Scanner(System.in);
        int FebDays=input.nextInt();                    //从键盘读取二月份的天数
        if(FebDays==29){
            System.out.println("您所输入的月份所在的年份为闰年");
        }else{
            System.out.println("您所输入的月份所在的年份为平年");
        }
        input.close();
    }
}
```

【运行结果】

根据二月份的天数判断是闰年还是平年程序的运行结果如图 3-3 所示。

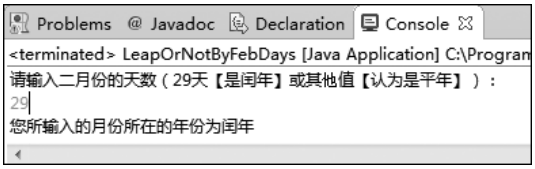


图 3-3　根据二月份的天数判断是闰年还是平年程序的运行结果

3.3.3　多重 if-else 结构

多重 if-else 结构用于针对某一个事件的多种情况进行处理,表现的含义为“如果……那么……否则如果……”。多重 if-else 结构的语法格式如下:

```
if(判断条件 1){
    [代码块 1]
```

```
}else if(判断条件 2){
    [代码块 2]
} …
else if(判断条件 n){
    [代码块 n]
}else{
    [代码块 n+1]
}
```

多重 if-else 结构的流程图如图 3-4 所示。

【例 3-3】 输入一个年份，判断是平年还是闰年。

【问题分析】

闰年产生的根本的原因是：地球绕太阳运行周期为 365 天 5 小时 48 分 46 秒(合 365.24219 天)，即一回归年(tropical year)。公历的平年只有 365 日，比回归年短约 0.2422 日，所余下的时间约为 4 年累计一天，故 4 年在 2 月加 1 天，使当年的天数为 366 天，这一年就为闰年。现行公历中每 400 年有 97 个闰年。按照每 4 年一个闰年计算，平均每年就要多算出 0.0078 天，这样经过 400 年就会多算出大约 3 天来。因此，每 400 年中要减少 3 个闰年。所以公历规定：年份是整百数时，必须是 400 的倍数才是闰年；不是 400 的倍数的年份，即使是 4 的倍数也是平年。这就是通常所说的：四年一闰，百年不闰，四百年再闰。例如，2000 年是闰年，1900 年则是平年。

判断闰年程序的流程图如图 3-5 所示。

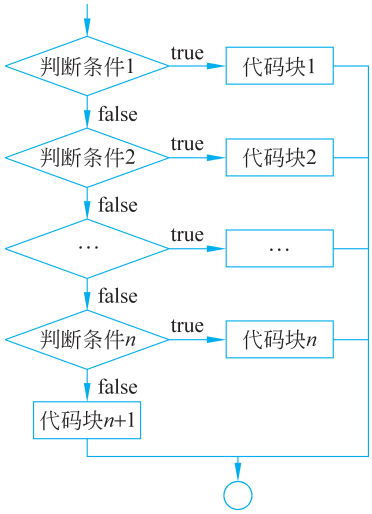


图 3-4 多重 if-else 结构

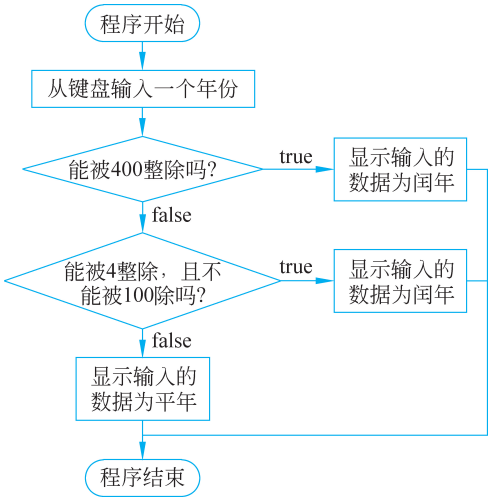


图 3-5 判断闰年程序的流程图

【程序实现】

```
import java.util.Scanner;
public class LeapYear {
    public static void main(String[] args) {
        System.out.println("请输入一个年份：");
        Scanner input=new Scanner(System.in);
        int year=input.nextInt();                    //从键盘读取一个年份
```

```
        if(year %400==0){
            System.out.println("您所输入的年份是闰年");
        }else if(year %4==0 && year %100!=0){
            System.out.println("您所输入的年份是闰年");
        }else{
            System.out.println("您所输入的年份是平年");
        }
    }
}
```

【运行结果】

运行程序时，输入一个数据，则会给出所输入的数据是闰年还是平年的判断，图 3-6 是在 Eclipse 控制台中运行 4 次程序后的结果。

(a) (b) (c) (d)

图 3-6 检查输入的年份是否为闰年程序的 4 次运行结果

3.3.4 if-else 条件语句的嵌套

if 语句的嵌套就是在 if 或 else 子句中又包含一个或多个 if 语句。这样的语句一般都用在比较复杂的分支结构程序中。if-else 嵌套结构的语法格式如下：

```
if(判断条件 1){
    if(判断条件 2){
        [代码块 1]
    }else{
        [代码块 2]
    }
}else{
    if(判断条件 3){
        [代码块 3]
    }else{
        [代码块 4]
    }
}
```

if-else 条件语句的嵌套结构的程序流程图如图 3-7 所示。

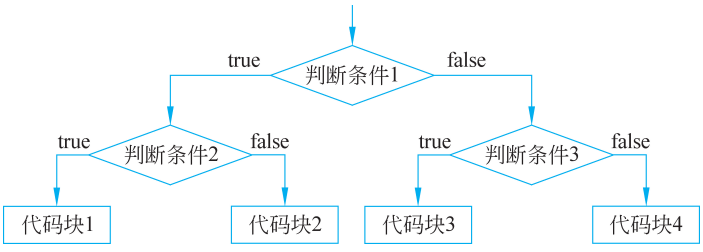


图 3-7 if-else 条件语句的嵌套结构的程序流程图

注意

代码块中的代码如果只有一行时，上述语法格式中的花括号也可以去掉，else 子句与同一代码块中离得最近的 if 子句相匹配。代码如下：

```
public class MyAge {
    public static void main(String[] args) {
        int age=10;                    //定义一个年龄变量
        if(age>=10)
            if(age>10)
                System.out.println("大于 10 岁");
            else
                System.out.println("等于 10 岁");
        else
            System.out.println("小于 10 岁");
    }
}
```

大家在阅读这段代码时，会有些吃力，因为不能快速地找到 if 程序块的开头和结尾。另外在编写程序的过程中，程序可能会需要改动，一行的代码会变成多行，变成多行后，上面的程序就会出错，而带有括号的程序能让人清楚地了解每个程序段的开头和结尾，一行修改为多行后也不易出错，更容易阅读，也更容易修改和维护。因此，在实际应用中，if-else 各类结构代码中的花括号都不要省略，以免造成视觉的错误与程序的混乱。

【例 3-4】 假设某航空公司规定，乘客可以免费托运质量不超过 40kg 的行李。当行李质量超过 40 千克时，对头等舱的国内乘客超重部分每千克收费 3 元，对其他舱的国内乘客超重部分每千克收费 5 元，对外国乘客超重部分每千克的收费比国内乘客多一倍，对残疾乘客超重部分每千元收费比正常乘客少一半。编写一个程序，能根据乘客行李质量、乘客的国别和是否为残疾情况计算出托运行李要支付的费用。

【问题分析】

通过上面的描述，可以发现这是一个非常复杂的选择分支结构程序，分析后发现，影响托运费用的因素有行李质量、乘客国别、舱位级别、是否为残疾人几个要素，根据各个要素的关系，得到程序的流程图如图 3-8 所示。

【程序实现】

```
import java.util.Scanner;
public class ShippingRates {
    public static void main(String[] args) {
        double weight, money;                          //双精度的质量、运费
        String strShow="该乘客为：";                    //最终要显示的信息
        Scanner input=new Scanner(System.in);          //输入设置
        System.out.println("请输入行李质量(单位：kg)：");
        weight=input.nextDouble();                     //从键盘输入行李质量的值
        if(weight<=40){                                //小于 40kg 时，没有托运费
            money=0;
        }else{
            //先定义和取得关键信息
            System.out.println("是国内乘客吗(1：是，其他值：不是)?");
            int iInner=input.nextInt();                //从键盘输入是否为国内乘客
```

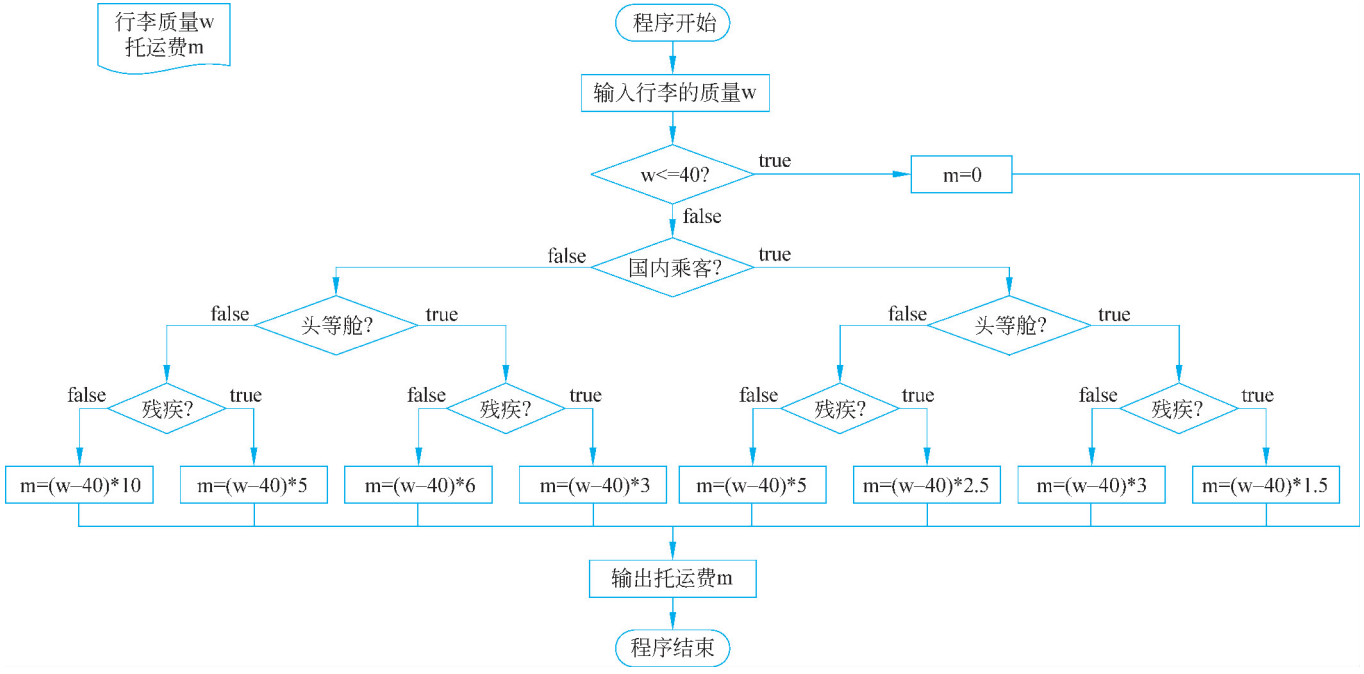


图 3-8　托运费的计算方法流程图

```
        System.out.println("是头等舱吗(1: 是,其他值: 不是)?");
        int iHead=input.nextInt();                          //从键盘输入是否为头等舱
        System.out.println("是残疾人吗(1: 是,其他值: 不是)?");
        int iDefomity=input.nextInt();                      //从键盘输入是否为残疾人

        //根据几个关键信息计算运费
        if(iInner==1){                                      //国内乘客
            strShow +="来自国内的一位乘坐";
            if(iHead==1){                                   //头等舱
                strShow +="头等舱";
                if(iDefomity==1){                           //残疾乘客
                    strShow +="的残疾乘客";
                    money=(weight-40) * 1.5;
                }else{                                      //正常乘客
                    strShow+="的正常乘客";
                    money=(weight-40) * 3;
                }
            }else{                                          //其他舱
                strShow+="其他舱";
                if(iDefomity==1){                           //残疾乘客
                    strShow +="的残疾乘客";
                    money=(weight-40) * 2.5;
                }else{                                      //正常乘客
                    strShow+="的正常乘客";
                    money=(weight-40) * 5;
                }
            }
        }else{                                              //外国乘客
            strShow +="来自国外的一位乘坐";
            if(iHead==1){                                   //头等舱
                strShow +="头等舱";
                if(iDefomity==1){                           //残疾乘客
                    strShow +="的残疾乘客";
                    money=(weight-40) * 3;
                }else{                                      //正常乘客
                    strShow+="的正常乘客";
                    money=(weight-40) * 6;
                }
            }else{                                          //其他舱
                strShow +="其他舱";
                if(iDefomity==1){                           //残疾乘客
                    strShow +="的残疾乘客";
                    money=(weight-40) * 5;
                }else{                                      //正常乘客
                    strShow+="的正常乘客";
                    money=(weight-40) * 10;
                }
            }
        }
    }
    input.close();                                          //关闭输入设备
    System.out.println(strShow);
```

```
            System.out.println("行李重: "+weight+",所需运费为: "+money+"元。");
        }
    }
```

【运行结果】

程序的一个运行结果如下：

```
请输入行李质量(单位: kg): 41
是国内乘客吗(1: 是,其他值: 不是)? 1
是头等舱吗(1: 是,其他值: 不是)? 1
是残疾人吗(1: 是,其他值: 不是)? 0
该乘客为: 来自国内的一位乘坐头等舱的正常乘客
行李重: 41.0,所需运费为: 3.0元。
```

3.3.5 switch case 结构

switch case 结构是多分支的开关语句结构。根据表达式的值来执行输出的程序结构。这种结构一般用于多条件多值的分支程序。它的一般形式如下：

```
switch(表达式){
    case 常量表达式 1:代码块 1;
        [break;]
    case 常量表达式 2:代码块 2;
        [break;]
    …
    case 常量表达式 n:代码块 n;
        [break;]
    [default:代码块 n+1;[break;]]
}
```

switch 语句中表达式的值必须是整型或字符型，即 int、short、byte 和 char 型。

case 为分支开关，case 中的常量表达式的值也必须是整型或字符型的，与表达式的数据类型相兼容的值。

代码块 1 是一条或多条 Java 语句，当常量表达式 1 的值与表达式的值相同时，则执行该代码块，如果不同则继续判断，直到达到表达式 n。

代码块 n 是一条或多条 Java 语句，当常量表达式 n 的值与表达的值相同时，则执行该代码块，如果不同则执行 default 处理中的代码块。

default 为可选参数，如果没有这个参数，而且所有的常量值与表达式的值都不匹配时，那么 switch 语句就不会执行任何操作。

break 为可选参数，主要用于跳出 switch 分支结构，如果没有使用 break 参数，则程序会继续向下执行下一个 case 判断和代码块。switch 分支结构的程序流程图如图 3-9 所示。

【例 3-5】 使用 Java 程序随机生成一个字母，并判断这个字母是元音字母还是辅音字母。

【问题分析】

简单地说，元音字母包括 a、e、i、o、u，而 y、w 在一些词中也可以读成元音，被称为半元音，如 my、cycle、paw、few 等，其他的字母则为辅音字母。程序流程图如图 3-10 所示。

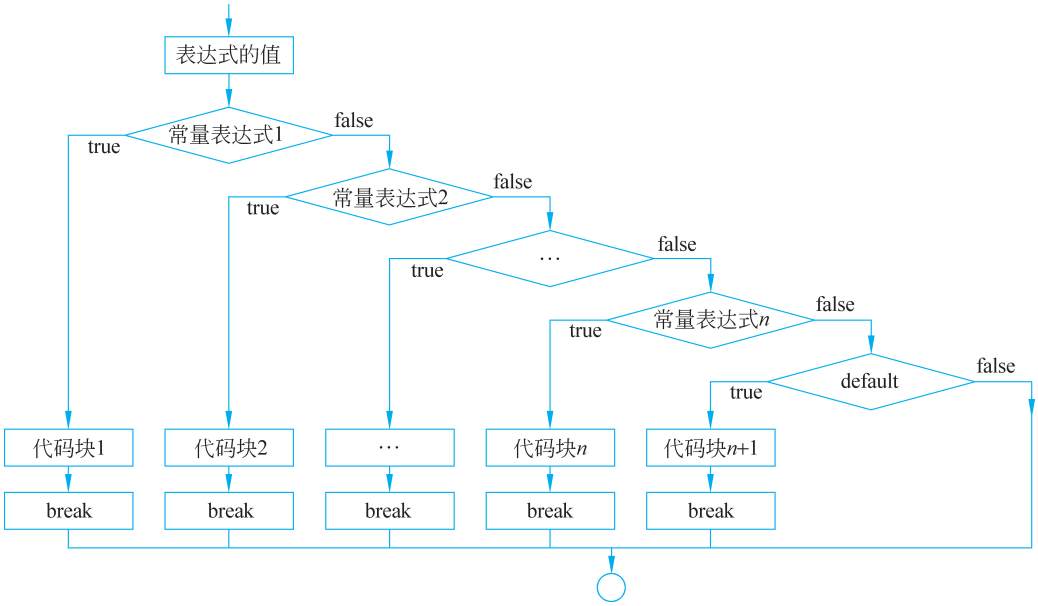


图 3-9　switch 分支结构程序流程图

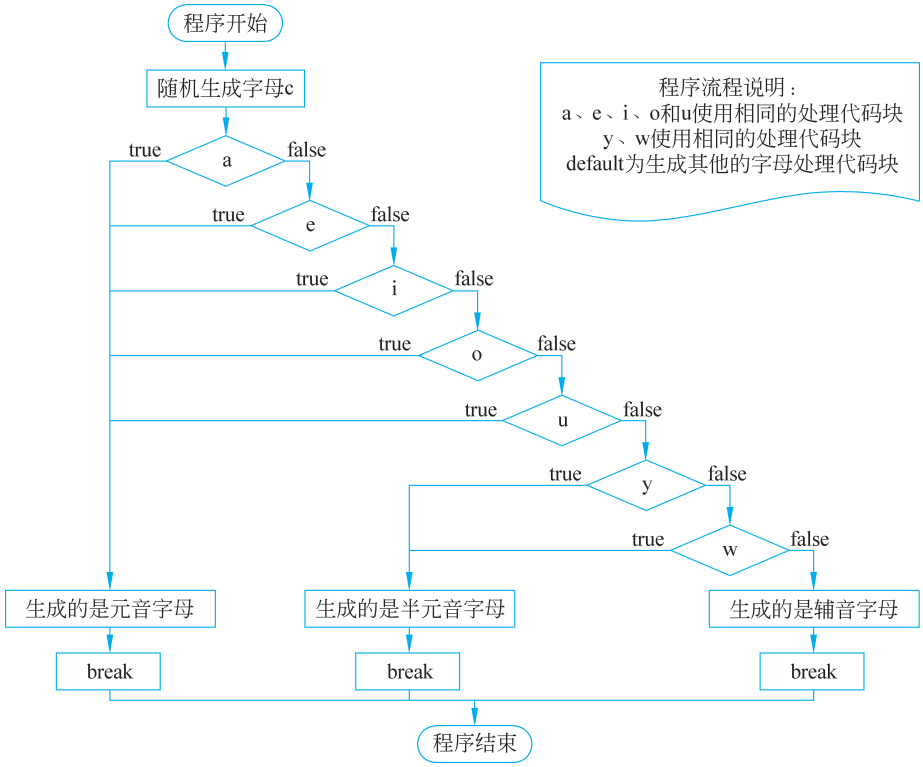


图 3-10　随机生成一个字母并判断是否为元音程序流程图

【程序实现】

```
public class VowelsAndConsonants {
```

```
    public static void main(String[] args) {
        char c=(char) (Math.random() * 26+'a');
        System.out.print("生成的字母是: "+c+",");
        switch (c) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
             System.out.println("是元音字母。");
             break;
        case 'y':
        case 'w':
             System.out.println("半元音字母。");
             break;
        default:
             System.out.println("是辅音字母。");
        }
    }
}
```

【运行结果】

图 3-11 是程序运行 3 次所得到的结果。

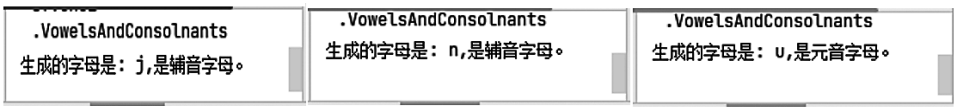


图 3-11　随机生成字母并判断是否为元音的程序的 3 次运行结果

3.4　任务实现

学习了几类分支结构后，下面着手编写本章开头所提出的分时问候程序吧！

【问题分析】

通过前面的任务分析，我们知道分时问候所涉及的时间段比较多，要采用选择结构来处理分支，程序在不同的时间段运行时，所面临的选择情况也比较多，因此采用多重 if-else 分支结构来实现程序。分时问候的程序流程结构如图 3-12 所示。

【程序实现】

```
import java.util.Date;
public class TipByHour {
    public static void main(String[] args) {
        Date dt=new Date();              //生成日期时间对象
        int hour=dt.getHours();          //取出当前的时间
        if(hour<6){
            System.out.println("主人,凌晨好,您起得真早啊。");
        }else if(hour<9){
            System.out.println("主人,早晨好,新的一天开始了。");
        }else if(hour<12){
```

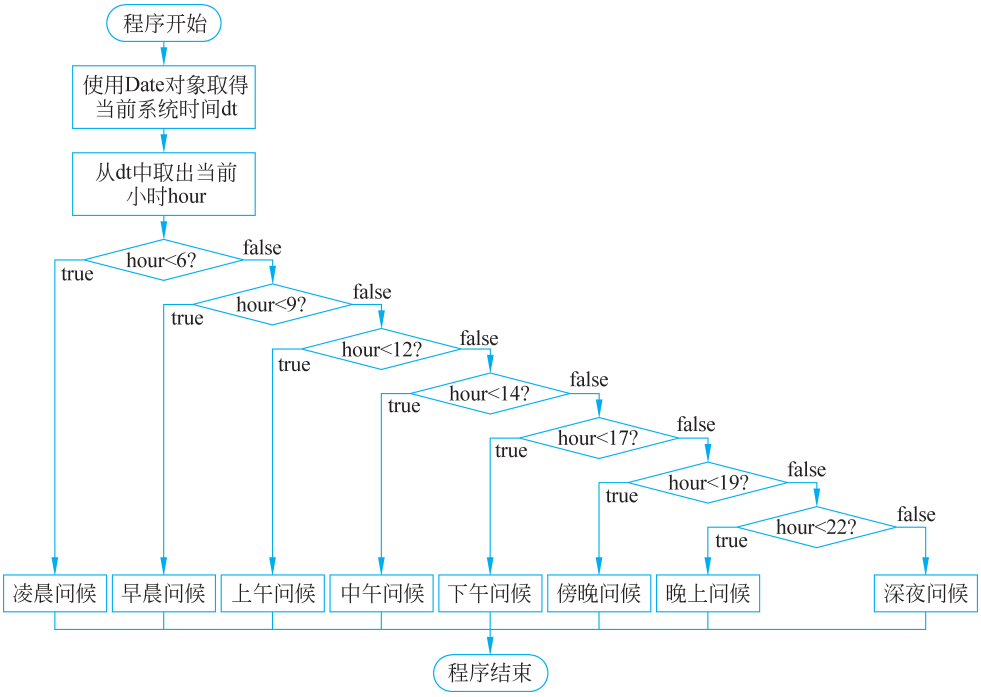
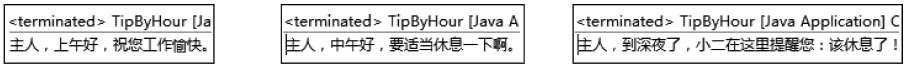


图 3-12　分时问候程序流程图

```
            System.out.println("主人,上午好,祝您工作愉快。");
        }else if(hour<14){
            System.out.println("主人,中午好,要适当休息一下啊。");
        }else if(hour<17){
            System.out.println("主人,下午好,要打起精神来工作啊。");
        }else if(hour<19){
            System.out.println("主人,傍晚好,到了吃晚饭的时间了。");
        }else if(hour<22){
            System.out.println("主人,晚上好,该休息放松一下了。");
        }else{
            System.out.println("主人,到深夜了,小二在这里提醒您:该休息了!");
        }
    }
}
```

【运行结果】

运行程序,该程序会按当前的操作系统时间给出对应的问候语。图 3-13 是在 Eclipse 控制台中的 3 个时间段分别运行程序后的效果。



(a) [9,12)时间段的运行效果　(b) [12,14)点前的运行效果　(c) [22,24)时间段的运行效果

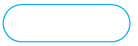
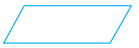
图 3-13　在 3 个不同的时间段运行分时问候程序的效果

3.5 知识拓展

3.5.1 程序流程图

本章程序设计过程中用到了程序流程图，流程图是逐步解决指定问题的步骤和方法的一种图形化的表示方法，流程图能帮助设计者直观、清晰地分析问题或是设计解决方案，是程序开发人员的得力助手。流程图是程序分析中最基本、最重要的分析技术，它是进行流程程序分析过程中最基本的工具。程序流程图运用工序图示符号对生产现场的整个制造过程做详细的记录，以便对零部件、产品在整个制造过程中的生产、加工、检验、存储等环节进行详细的研究与分析，特别适用于分析生产过程中的成本浪费，提高经济效益。表 3-2 为程序流程图的符号说明。

表 3-2 程序流程图的符号说明

符 号	意 义	符 号	意 义
	计算步骤/处理过程		判断和分支
	程序的开始或结束		文档及处理说明
	子计算步骤/子处理过程		连接点
	输入输出指令(或数据)		流程线(控制流)

3.5.2 switch 表达式

从 JDK 12 开始引入，JDK 14 中正式引入新的 switch 语法，代码模板与语法规则如下：

```
var res=switch(obj) {          //新的 switch 语法可以有返回值,可以被变量接收
    // case 中的匹配值可以有多个,使用逗号分隔
    //使用 ->箭头操作符返回匹配此 case 语句的结果
    case[匹配值, ...] ->直接返回选项值 1;
    case[匹配值, ...] ->{  在子块中使用 yield 返回值(不能使用 return);  };
    case ...                   //根据不同的分支,可以存在多个 case
    //注意,如果前面的 case 分支已经覆盖了所有条件的话,default 分支可以不写
    //否则 switch 表达式要求必须涵盖所有的可能,所以需要添加 default
    default->其他情况下的返回选项值;
};
```

【例 3-6】 从键盘输入一个年份和月份，使用 switch 表达式输出该月份的天数。

【问题分析】

一年中的月份天数有 3 种情况：第 1 种是 1、3、5、7、8、10、12 月份有 31 天，第 2 种是 4、6、9、11 月份有 30 天，第 3 种是 2 月份，闰年的 2 月份有 29 天，平年的 2 月份有 28 天。因此，本例题要先看月份，如果是前两种情况，则直接输出天数信息；如果是第 3 种情况，还要

检查是闰年还是平年，根据年份类型输出天数信息。

【程序实现】

在 examp 包中创建类文件 Examp06NewSwitch，修改该文件的内容如下：

```
import java.util.Scanner;
//根据键盘输入年份与月份，输出该月的天数
public class Examp06NewSwitch {
    public static void main(String[] args) {            //主入口方法
        Scanner sc=new Scanner(System.in);
        System.out.print("请输入要查看的年份：");
        int year=sc.nextInt();
        System.out.print("请输入要查看的月份：");
        int month=sc.nextInt();
        //使用 switch 表达式计算出天数来
        int daysNum=switch (month){
            case 1,3,5,7,8,10,12->31;                    //月份：1,3,5,7,8,10,12
            case 4,6,9,11->30;                           //月份：4,6,9,11
            default->{
            if(year%400==0 || (year%4==0 && year% 100!=0)){  //闰年，返回 29 天
                    yield 29;
                }else{                                   //平年，返回 28 天
                    yield 28;
                }
            }
        };
System.out.println(year+"年"+month+"月有："+daysNum+"天");
    }
}
```

【运行结果】

运行程序，按提示输入一个年份，再输入一个月份，其中的一个输入实例的运行结果如图 3-14 所示。

图 3-14　根据输入的年份与月份输出该月的天数

3.5.3　新的日期时间 API

经典的日期时间类 Date 从 JDK 1.0 就有了，但使用起来比较麻烦，而且不是线程安全的。为了解决经典时间类存在的问题，与日期时间的国际标准 ISO-8601 进行接轨，JDK 8 的 java.time 包中发布了一套新的日期时间 API。在新的 JDK 中再使用旧的 Date 类及其方法时，会看到一些调用的方法上加上了删除线，并显示 Xxx() is deprecated，即该方法不推荐使用了。主要有如下几类。

1. LocalDate、LocalTime、LocalDateTime

LocalDate：本地日期，值无时区属性。

LocalTime：本地时间，值无时区属性。

LocalDateTime：本地日期与时间，值无时区属性，可以看作是 LocalDate 与 LocalTime

的组合。

这一组类的用法类似，本书只介绍 LocalDateTime 的简单用法。本组类不能使用 new 关键字进行实例化，可以通过 now()或 of()方法来实例化。

```
LocalDateTime ldt = LocalDateTime.now();        //获取当前日期时间
```

常用的方法如下。

getYear()：获取当前实例的年份信息(LocalDate 也有这个方法)。

getMonth()：获取当前实例的月份信息(LocalDate 也有这个方法)。

getDayOfMonth()：获取当前实例是该月的几号(LocalDate 也有这个方法)。

getDayOfWeek()：获取当前实例是周几(LocalDate 也有这个方法)。

getDayOfYear()：获取当前实例是当年中的第几天(LocalDate 也有这个方法)。

getHour()：获取当前实例小时信息(LocalTime 也有这个方法)。

getMinute()：获取当前实例的分钟信息(LocalTime 也有这个方法)。

getSecond()：获取当前实例的秒信息(LocalTime 也有这个方法)。

getNano()：获取当前实例的纳秒信息(LocalTime 也有这个方法)。

2. Instant 时间戳

Instant 表示了时间线上一个确切的点，定义为距离初始时间的时间差(初始时间为 UNIX 元年 GMT 1970 年 1 月 1 日 00:00)，经测量一天有 86400 秒，从初始时间开始不断向前移动。

```
Instant inst=Instant.now();
```

获取到的日期实例是 0 时区当前的日期与时间信息，如果要获取当前时区的日期时间的信息，则要根据当前系统所在的时区，设置时间的偏移量。

Instant 有如下一些常用的方法。

plusSeconds()：增加一些秒数。

plusMillis()：增加一些毫秒数。

plusNanos()：增加一些纳秒数。

minusSeconds()：减去一些秒数。

minusMillis()：减去一些毫秒数。

minusNanos()：减去一些纳秒数。

【例 3-7】 使用 Instant 获取日期时间的简单示例。

【问题分析】

使用 Instant 获取到的是 0 时区的日期与时间，要获取某个时区的当地时间，要添加时间的偏移设置。

【程序实现】

```
import java.time.*;
//例 3-7 使用 Instant 获取日期时间的简单示例
public class InstantTest {
    public static void main(String[] args) {
        Instant inst=Instant.now();          //调用静态方法 now 构建 Instant 类对象
        System.out.println("默认时间: "+inst);
        /*默认为 0 时区的时间，通过设置时偏移量，转换到当前系统所在的时区，
```

```
        结果含时区信息,时区: 东 1~12 区+n,+号可以省略,
        西 1~12 区-n,如中国是东 8 区,则为+8 */
        OffsetDateTimeo dt=inst.atOffset(ZoneOffset.ofHours(8));
        System.out.println("设置时间偏移量后的时间: "+odt);
        Instant inst2=inst.plusSeconds(5);          //获取一个新的 Instant 值
        System.out.println("增加 5 秒后的时间: "+inst2);
        /* 也可以通过 atZone 设置所在的时区来获取对应时区的时间
        含时区及时区名称信息,要生成新的时间变量 */
        ZonedDateTime zdt2=inst2.atZone(ZoneId.of("Asia/Shanghai"));
        System.out.println("本地增加 5 秒后的时间: "+zdt2);
    }
}
```

【运行结果】

程序的运行结果如图 3-15 所示。

```
默认时间: 2022-08-15T01:20:00.970786900Z
设置时间偏移量后的时间: 2022-08-15T09:20:00.970786900+08:00
增加5秒后的时间: 2022-08-15T01:20:05.970786900Z
本地增加5秒后的时间: 2022-08-15T09:20:05
```

图 3-15　使用 Instant 获取日期时间并转换成本地时间

3. 日期时间间隔

Duration: 计算两个时间之间的间隔。

Period: 计算两个日期之间的间隔。

【例 3-8】 计算两个时间与两个日期之间的间隔。

【问题分析】

使用 Duration 类计算两个时间之间的间隔,时间实例可以使用 LocalTime,可以使用 LocalDateTime,也可以使用 Instant 生成。使用 Period 类计算两日期之间的间隔。

【程序实现】

```
import java.time.*;
//例 3-8 计算日期时间之间的间隔
public class PeriodAndDuration {
    public static void main(String[] args) throws InterruptedException {
        LocalTime lt1=LocalTime.now();
        Thread.sleep(1000);  //线程暂停 1000 毫秒,即 1 秒
        LocalTime lt2=LocalTime.now();
        Duration dura1=Duration.between(lt1, lt2);
        System.out.println("两个 LocalTime 实例间的时间间隔:"+dura1.toMillis()+"毫秒");
        System.out.println("================================");
        Instant inst1=Instant.now();
        Thread.sleep(1000);   //线程暂停 1000 毫秒,即 1 秒
        Instant inst2=Instant.now();
        System.out.println("两个 Instant 实例间的时间间隔:"+Duration.between
        (inst1, inst2).toMillis()+"毫秒");
        System.out.println("================================");
        LocalDate ld1=LocalDate.now();
        LocalDate ld2=ld1.plusDays(2);  //实例 1 增加 2 天变成实例 2
```

```
        System.out.println("两个 LocalDate 间的日期间隔:"+Period.between(ld1,
        ld2).getDays()+"天");
    }
}
```

【运行结果】

程序的运行结果如图 3-16 所示。

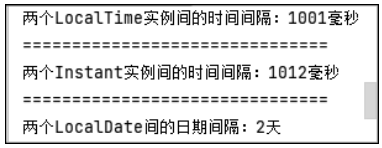


图 3-16　两个时间、两个日期的间隔实例

4. DateTimeFormatter 日期时间格式化

使用 DateTimeFormatter 日期时间格式化器对日期与时间进行格式化,其中提供了一些静态的格式,也可以自定义格式化器对日期时间进行格式化。

【例 3-9】　使用日期时间格式化器对日期时间进行格式化。

【问题分析】

本例主要测试 DateTimeFormatter 格式化器提供的静态格式与自定义格式的使用方法,可以通过 DateTimeFormatter 类的操作,查看其提供的静态格式,其中以 ISO_开头的静态常量为国际标准日期与时间格式,如 ISO_DATE、ISO_DATETIME 等,可以使用 ofXxx()方法设置自定义格式,如 ofPatten(String patter)等。

【程序实现】

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
//例 3-9 日期时间格式化实例
public class DateTimeFormatterTest {
    public static void main(String[] args) {
        DateTimeFormatter dtf=DateTimeFormatter.ISO_LOCAL_DATE_TIME;
        LocalDateTimeldt=LocalDateTime.now();
        System.out.println("未格式化时间:"+ldt);   // ISO 国际标准格式
        System.out.println("格式化后时间:"+dtf.format(ldt));
        //自定义格式
        DateTimeFormatter dtf2=DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
        System.out.println("自定义格式:"+dtf2.format(ldt));
    }
}
```

【运行结果】

该程序的运行结果如图 3-17 所示。

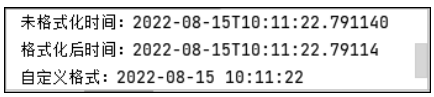


图 3-17　ISO 国际标准时间格式、本地时间格式、自定义时间格式

3.6 本章小结

本章介绍了解决选择分支问题的简单 if 结构、if-else 结构、多重 if-else 结构、if-else 语句嵌套结构、switch 结构，学习了程序流程图的相关知识。学习本章后，读者应该达成如下目标。

(1) 会编写简单 if 结构程序。
(2) 会编写 if-else 双分支程序。
(3) 会编写 if-else if-else 多分支结构程序。
(4) 会编写 switch 分支结构程序。
(5) 能说出 JDK 14 中 switch 语法的新变化。
(6) 能应用选择结构解决实际问题。

3.7 强化练习

3.7.1 判断题

1. 简单 if 结构是顺序程序结构。(　　)
2. if-else 结构程序有 2 个分支。(　　)
3. 多重 if-else 结构中的花括号不能写。(　　)
4. switch case 结构中的 case 块中必须加括号。(　　)
5. switch case 结构中的 default 为必选参数，必须得写上，否则程序会出错。(　　)
6. 新的 switch 语法(switch 表达式)的子块中要使用 yield 返回结果。(　　)
7. LocalDateTime 可以使用 new 关键字生成实例。(　　)

3.7.2 选择题

1. 阅读下面程序，分析该程序的功能是(　　)。

```
public class T1{
    public static void main(String[] args) {
    int a,b,c; a=78; b=67; c=12;
    if(a>b) {
      if(b>c) System.out.println("The min number:"+c);
          else System.out.println("The min number:"+b);
    }else {
      if(c<a)  System.out.println("The min number:"+c);
          else    System.out.println("The min number:"+a);
    }
  }
}
```

A. 求出 a、b、c 三个数中最小的数的值　　B. 求出 a、b、c 三个数中最大数的值
C. 求出 a、b、c 三个数的公约数的值　　D. 求出 a、b、c 三个数的公倍数的值

2.下面程序的输出结果是(　　)。

```
m=7; n=3;
if(m %n==2)
{
    n=m-2;
}
System.out.println(n);
```

A. 5　　B. 3　　C. 1　　D. 2

3. 下面程序执行后，c 的值是(　　)。

```
public class SwitchTest {
    public static void main(String[] args) {
        int c=2;
        switch (c) {
        case 1:
            c++;
        case 2:
            c++;
        case 3:
            c++;
        case 4:
            c++;
        case 5:
            c++;
            break;
        default:
            c=0;
        }
        System.out.println("现在 c 的值为: "+c);
    }
}
```

A. 5　　B. 6　　C. 2　　D. 0

4. 下列类中，(　　)不是 java.time 包中提供的新的日期时间的类。

A. LocalDate　　B. Instant　　C. Date　　D. LocalTime

3.7.3　简答题

1. 简述 if-else 结构的语法。
2. 简述多重 if-else 结构的语法。
3. 简述 switch 结构的语法，并说明 break 语句的作用。
4. 说出程序流程图中的几个常用的元素，简要介绍一下流程图的作用。
5. 说明输入一个年份，判断该年份是否为闰年的程序流程。

3.7.4　编程题

1. 对任意输入的整数，判断其是否能被 7 整除，如果能被 7 整除输出该数除以 7 的商，否则，输出信息“不能被 7 整除”。

2. 输入一个数，如果这个数在除以 2 后的余数为 0，则屏幕上就会显示 The number is Even，否则就在屏幕上显示 The number is Odd。

3. 从键盘上录入一个学生的成绩，如果这个成绩在[90，100]内，则输出“优秀”；如果这个成绩在[70，90)内，则输出“良好”；如果这个成绩在[60，70)内，则输出“及格”；如果这个成绩在[0，60)内，则输出“不及格”；输入的是其他成绩时，输出“输入的成绩不合法”。

4. 从键盘录入一个字符串，判断录入的每个字符是大写字母还是小写字母：如果是大写字母则将其转换成小写字母后输出；如果是小写字母则将转换成大写字母后输出；如果是其他的字符则直接输出。

5. 编写一个智能购物计算小程序，在一家商店有书本、铅笔、橡皮、可乐、零食 5 种商品，商品价格如表 3-3 所示。

表 3-3　商品价格

商 品 名 称	价格/元
书本	12
铅笔	1
橡皮	2
可乐	3
零食	5

假如你带了 20 元，且必须购买一本书，剩余的钱还可以购买哪种商品？可以购买几件？购买完后又能剩余多少钱？

6. 个人所得税的计算方法是根据工资范围所在的工资段，分别使用各段的税率：[1，5000]部分为 0%，(5000，8000]部分为 3%，(8000，17000]部分为 10%，(17000，30000]部分为 20%，(30000，40000]部分为 25%，(40000，60000]部分为 30%，(60000，85000]部分为 35%，>85000 部分为 45%。根据这个税率计算方法，编写一个程序，实现输入一个人的工资，积累计算所有部分的所得税，并输出应该缴纳的个人所得税。