

Java 第三章 语言基础上机练习

22009200601 汤栋文

2023 年 3 月 5 日

目录

1.	个人信息	- 2 -
2.	任务重述	- 2 -
3.	PPT 练习	- 2 -
3.1	CharConst on P15	- 2 -
3.2	Assign on P21&P22	- 3 -
3.3	TDouble on P24	- 3 -
3.4	TestInit on P39	- 4 -
3.5	UnaryConversion on P43	- 4 -
3.6	Expression on P46	- 5 -
3.7	Equivalence on P48	- 6 -
3.8	CastType on P56	- 6 -
3.9	Compare on P58	- 7 -
3.10	FloatCompare on P62	- 7 -
4.	日历程序	- 8 -
4.1	基本思路	- 8 -
4.2	获取星期几解决方案	- 8 -
4.3	打印日历解决方案	- 9 -
4.4	人机交互性	- 10 -
4.5	任务总结	- 12 -
5.	总体总结	- 12 -

1. 个人信息

姓名：汤栋文

学号：22009200601

日期：2023 年 3 月 5 日

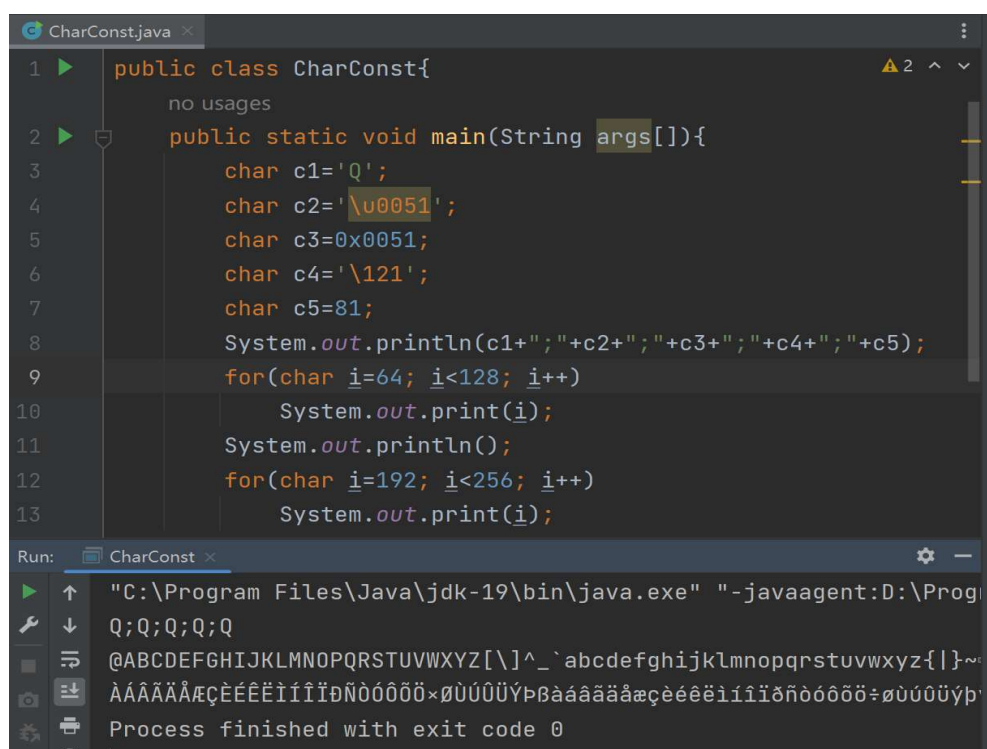
2. 任务重述

1. 执行 PPT 《JAVA 语言基础》中练习
(页码:15, 21, 22, 24, 39, 43, 46, 48, 56, 58, 62, 63, 76, 82)
2. 编写一个日历程序，要求程序具有良好的人机交互性能并完成：
 - i. 根据用户输入的年份输出该年日历
 - ii. 根据用户输入的日期输出该天星期

3. PPT 练习

3.1 CharConst on P15

1. 输出：Q; Q; Q; Q
2. 分析：在 Java 中 char 类型的字符是使用 Unicode 码储存的。
 - c1 直接输入字符；
 - c2 使用转义 Unicode 码，在转义过程中将会把对应的值转换成对应的字符；
 - c3 直接使用 Unicode 码表示，说明 Java 储存字符时储存的就是 Unicode 码；
 - c4 使用转义 ASCII 码表示，与 c2 类似，转义过程中将会转换成对应的字符；
3. 拓展：
 - i. c5 直接赋值十进制数字同样可以打印出 Q 验证了确实储存的仅是一个数字。
 - ii. 可以通过如下的循环打印出一部分 ASCII 码中的字符，以及一段 ASCII 码以外但在 Unicode 码以内的字符。

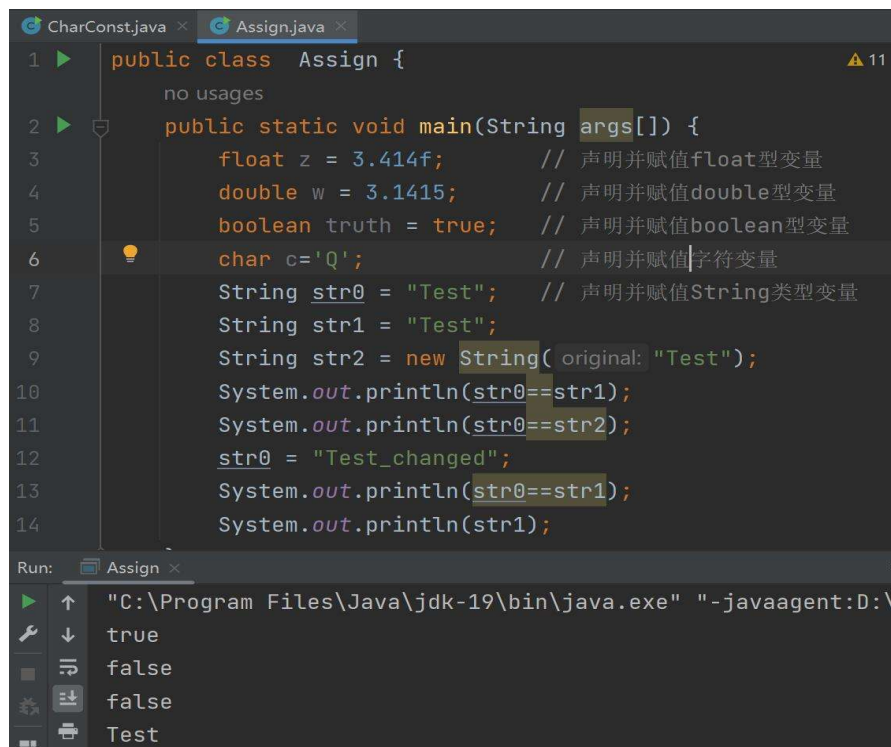


```
CharConst.java
1 public class CharConst{
    no usages
2     public static void main(String args[]){
3         char c1='Q';
4         char c2='\u0051';
5         char c3=0x0051;
6         char c4='\121';
7         char c5=81;
8         System.out.println(c1+";"+c2+";"+c3+";"+c4+";"+c5);
9         for(char i=64; i<128; i++)
10            System.out.print(i);
11        System.out.println();
12        for(char i=192; i<256; i++)
13            System.out.print(i);
    }
}

Run: CharConst
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:D:\Progr
Q;Q;Q;Q;Q
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæçèéêëìíîïðñóôõö÷øùúûüýþ
Process finished with exit code 0
```

3.2 Assign on P21&P22

1. 这变量声明。
2. 需要注意的:
 - i. float 类型后面要加上 F, long 类型后面要加上 L。
 - ii. 布尔类型用 Boolean 而不是 bool。
 - iii. 字符串为 String, 它的首字母要大写。
3. 字符串的初始化方式是多样的。
 - i. String str = "Test" 和 String str = new String("Test") 有区别: 第 10 行和第 11 行的比较可以说明, 前者会检索字符池, 如果存在 Test, 就会直接返回引用而不创建一个新的字符常量。
 - ii. 这时我们考虑如果修改的 str0 是否会导致 str1 的值发生改变: 通过 12 到 14 行的代码可以说明, 在修改 str0 的时候, Java 创建了一个新的字符串常量, 而没有改变之前常量和索引



```
CharConst.java x Assign.java x
1 public class Assign {
2     public static void main(String args[]) {
3         float z = 3.414f; // 声明并赋值float型变量
4         double w = 3.1415; // 声明并赋值double型变量
5         boolean truth = true; // 声明并赋值boolean型变量
6         char c = 'Q'; // 声明并赋值字符变量
7         String str0 = "Test"; // 声明并赋值String类型变量
8         String str1 = "Test";
9         String str2 = new String(original: "Test");
10        System.out.println(str0==str1);
11        System.out.println(str0==str2);
12        str0 = "Test_changed";
13        System.out.println(str0==str1);
14        System.out.println(str1);
15    }
16 }

Run: Assign x
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:D:\f
true
false
false
Test
```

3.3 TDouble on P24

Java 将基本类型封装成了类的形式, 以 Double 类为例:

它提供了常量 NaN, NEGATIVE_INFINITY 以及 POSITIVE_INFINITY

1. NaN: NaN 意思是 Not a Number, 它不等于任何数, 也不大于或者小于任何数。判断一个结果是否为 NaN 需要用到方法 isNaN()。
2. INFINITY: 正无穷比任何数大, 负无穷比任何数小。判断一个数是否为无穷在 Double 类中也有相应的方法, isInfinite()。
3. 注意在给类命名的时候不要与 Java 内部的类发生冲突, 否则会把内部的类覆写掉, 这个地方命名为 TDouble 就是如此, 如果使用 Double 命名将不能正常运行。

```

CharConst.java x Assign.java x TDouble.java x
2 public static void main(String args[]) {
3     System.out.print(0.0 == -0.0);System.out.print(" ");
4     System.out.println();
5
6     System.out.print(1.0 < Double.NaN);System.out.print(" ");
7     System.out.print(1.0 > Double.NaN);System.out.print(" ");
8     System.out.print(1.0 == Double.NaN);System.out.print(" ");
9     System.out.print(1.0 != Double.NaN);System.out.print(" ");
10    System.out.print(Double.NaN == Double.NaN);
11    System.out.println();
12
13    System.out.print(0.0 / 0.0);System.out.print(" ");
14    System.out.print(1.0 / 0.0);System.out.print(" ");
15    System.out.print(1.0 / -0.0);System.out.print(" ");
16    System.out.println();
17
Run: TDouble x
  "C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:D:\Pro
  true
  false false false true false
  NaN Infinity -Infinity

```

3.4 TestInit on P39

0. 无论如何新建一个变量顺手初始化是一个必要的良好习惯。
1. 对象的成员变量有默认初始值，局部变量必须在使用前手工赋初始值，若局部变量未初始化就使用，编译器报错。
2. Math.random()会返回一个 0-1 之间的 double 类型随机数。

```

TestInit.java x
3 public static void main(String args[]) {
4     TestInit init = new TestInit();
5     int x = (int) (Math.random() * 100);
6     int z;
7     int y=0; // int y 会报错;
8     if (x > 50) y = 9;
9     z = x + y + init.x;
10    System.out.println("x=" + x + " y=" + y +
11        " z=" + z + " init.x=" + init.x);
Run: TestInit x
  "C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:
  x=8 y=0 z=8 init.x=0

```

3.5 UnaryConversion on P43

1. 关于强制类型转换，低精度可以直接赋给高精度，高精度若要赋给低精度需要强制类型转换。

- 比 int 精度小的整数类型进行运算时会一致转换为 int, 所以 `byte b=0; b=b+1;` 是错误的, 因为在运算中 `b` 已经被转化为了 `int`, 而高精度赋值给低精度需要强制类型转换。同样的, 浮点运算会一直转换为 `double`, 所以 `float f=1; f=f+1.0;` 也是错的。
- 在具体写程序的时候如果不是对内存有着过分的要求, 一致使用 `int` 和 `double` 以免产生不必要的错误。如果真的需要榨干最后一点内存的编程, 那么我们应该使用 C 语言或者在汇编层面上去对内存做更直接的操作。

```
UnaryConversion.java
2 public static void main(String[] args){
3     byte b=2;
4     char c='\u1234';
5     int x=8,y=3;
6     //byte b2 = -b; //int值不能直接赋给byte类型变量b2
7     //char c2 = +c; //int值不能直接赋给char类型变量c2
8     System.out.println((-b) + ";" + (+c));
9     int i=~b; //byte转换为int
10    System.out.println(Integer.toHexString(i));
11    System.out.println(x/y);
12    System.out.println(x/(float)y);
13 }
```

Run: UnaryConversion

"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:..."
-2;4660
fffffffd
2
2.6666667

3.6 Expression on P46

一些表达式以及运行结果。注意区分`++a` 和 `a++`。与 C 完全一致。

```
UnaryConversion.java Expression.java CharConst.java
2 public static void main(String args[]){
3     int a; double d;
4     a=1; a = 46 / 9;
5     System.out.println(a);
6     a=1; a = 46 % 9 + 4 * 4 - 2;
7     System.out.println(a);
8     a=1; a = 45 + 43 % 5 * (23 * 3 % 2);
9     System.out.println(a);
10    d=1.0; d = 4 + d * d + 4;
11    System.out.println(d);
12    d=1.0; a=1; d += 1.5 * 3 + (++a);
13    System.out.println(d);
14    d=1.0; a=1; d -= 1.5 * 3 + a++;
15    System.out.println(d);
}
```

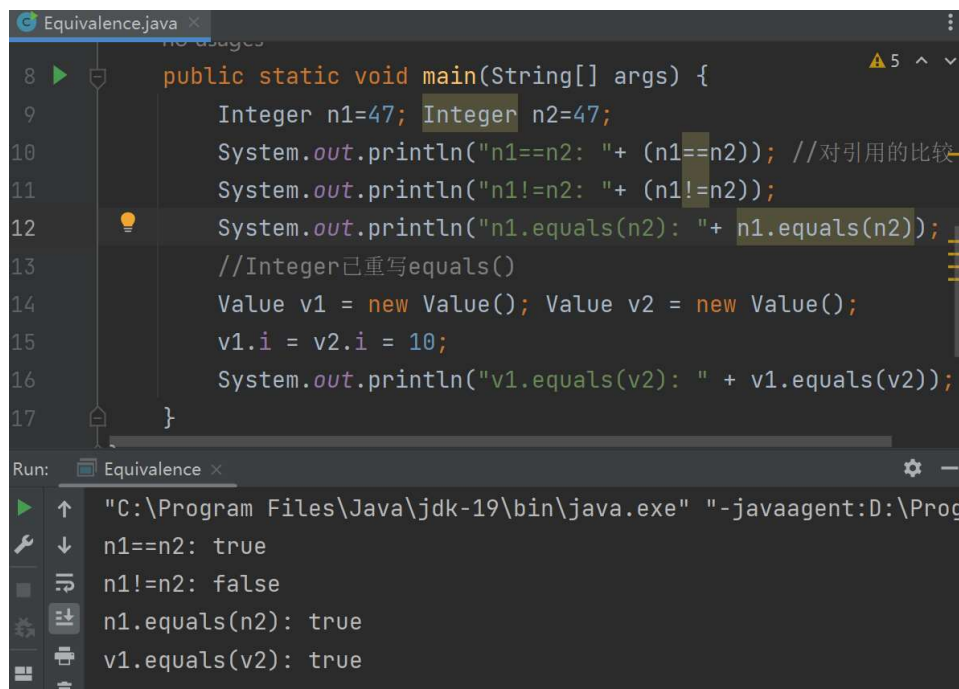
Run: Expression

"C:\Program Files\Java\jdk-19\bin\java.exe" "-java..."
5
15
48
9.0
7.5
-4.5

3.7 Equivalence on P48

这里主要是说 == 和 equals() 方法：

1. ==：在比较两个对象时，直接比较引用是否是一个，而不关心具体的值。
2. equals()：是由类的编写者定义的，按照一定的方法来比较具体的值。
3. 补充：Integer(value)，这个构造方法已经弃用，现在直接赋值就可以。
Java 允许像 a=b=2 这种连续赋值方式。



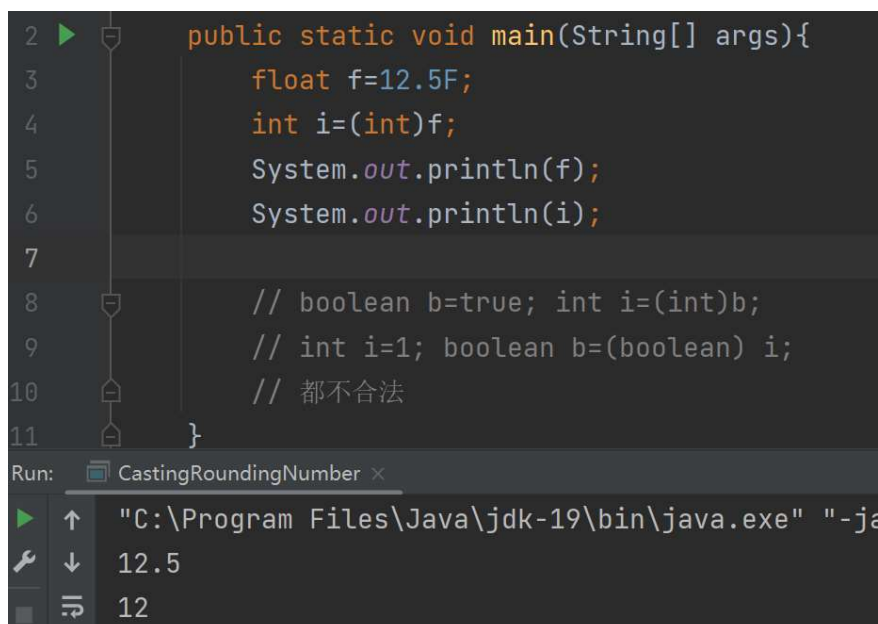
```
Equivalence.java
8 public static void main(String[] args) {
9     Integer n1=47; Integer n2=47;
10    System.out.println("n1==n2: " + (n1==n2)); //对引用的比较
11    System.out.println("n1!=n2: " + (n1!=n2));
12    System.out.println("n1.equals(n2): " + n1.equals(n2));
13    //Integer已重写equals()
14    Value v1 = new Value(); Value v2 = new Value();
15    v1.i = v2.i = 10;
16    System.out.println("v1.equals(v2): " + v1.equals(v2));
17 }
```

Run: Equivalence

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:D:\Progr
n1==n2: true
n1!=n2: false
n1.equals(n2): true
v1.equals(v2): true
```

3.8 CastType on P56

1. Boolean 类型不允许任何的强制类型转换。这个做法使得代码更加清晰，布尔型就是布尔型，整形就是整形，不容易出现混乱。
2. Java 中浮点数转换为整型采取的是舍小数。我个人认为这非常好，四舍五入可能引起不可意料的错误。



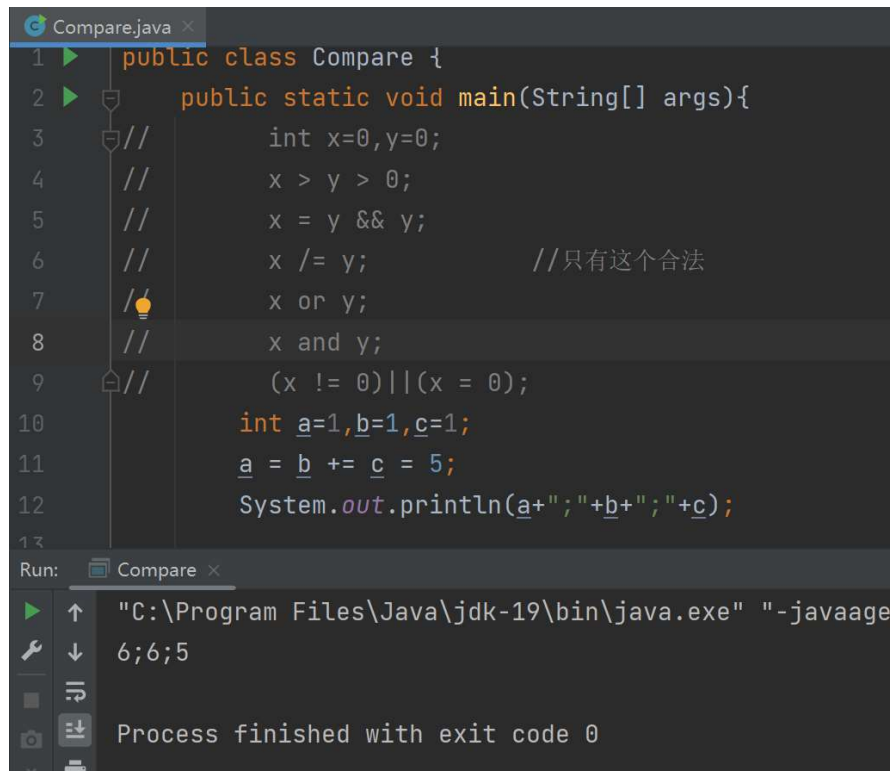
```
2 public static void main(String[] args){
3     float f=12.5F;
4     int i=(int)f;
5     System.out.println(f);
6     System.out.println(i);
7
8     // boolean b=true; int i=(int)b;
9     // int i=1; boolean b=(boolean) i;
10    // 都不合法
11 }
```

Run: CastingRoundingNumber

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-ja
12.5
12
```

3.9 Compare on P58

1. Java 不允许 $x > y > 0$ 类似的连续比较，这也与 boolean 和 int 不互通有关，如果算过来 $x > y$ 是 boolean 型，不能与 int 比较。
2. Java 中任何需要判断真假，或者进行“与”和“或”操作时，必须都是布尔型。
3. Java 中 if 后面的东西必须是布尔型，如果要判断 x 非零，不能直接 `if(x)`，而一定要写成 `if(x!=0)`。这是一种非常规范的要求，这样可以提高代码的可读性让逻辑更加清楚。
4. 永远不要写 `a=b+=c=5` 这种代码。（容易挨打...）



```
1 public class Compare {
2     public static void main(String[] args){
3         // int x=0,y=0;
4         // x > y > 0;
5         // x = y && y;
6         // x /= y; //只有这个合法
7         // x or y;
8         // x and y;
9         // (x != 0) || (x = 0);
10        int a=1,b=1,c=1;
11        a = b += c = 5;
12        System.out.println(a+" "+b+" "+c);
13    }
14 }
```

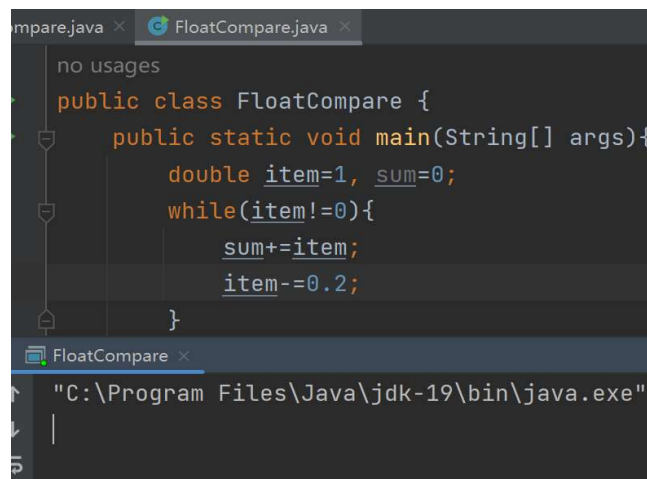
Run: Compare

"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaage
6;6;5

Process finished with exit code 0

3.10 FloatCompare on P62

1. 浮点数不要用 `==` 作为结束条件，很好理解，因为浮点数并不总是精确的。否则可能导致不精确的循环次数和结果，更严重的会把出现死循环把计算机卡死还发现不了问题。
2. 如果一定要用浮点数计数，那么可以将判断条件改为 `!(item > -0.00001 && item < 0.00001)`，即可达到效果。（图片为陷入了死循环）



```
no usages
public class FloatCompare {
    public static void main(String[] args){
        double item=1, sum=0;
        while(item!=0){
            sum+=item;
            item-=0.2;
        }
    }
}
```

FloatCompare

"C:\Program Files\Java\jdk-19\bin\java.exe"

4. 日历程序

4.1 基本思路

题目要求实现一个日历程序，有两个基本功能：

一个是给定年份输出日历，一个是给定日期确定星期。

1. 这里我从第二个问题开始解决，因为第一个问题可以在第二个问题的基础上降低难度。出一个基准日期。计算出从给定天数到基准日期的“距离”。然后对 7 取模就得到结果。
2. 第一个问题，我们只需要得到这一年的 1 月 1 日是星期几，就可以依次打印出这一年的日历，只需要注意在月末换行即可。

4.2 获取星期几解决方案

我们以 2004 年 4 月 22 日举例

1. 由于历法变化，1582 年 10 月以前的历法与现在不同，为了方便起见，我们取 1583 年 1 月 1 日作为基准日期。在 `getDaysBeforeThisDayInPastYear` 函数中我们计算出到从 1583 年 1 月 1 日到 2003 年 12 月 31 日之间的天数。

```
int getDaysBeforeThisDayInPastYear(int year){
    int days = 0;
    for (int i=1583; i<year; i++){
        if (isLeapYear(i)) days+=366;
        else days+=365;
    } // days in past years
    return days;
}
```

2. 然后在 `getDaysBeforeThisDayInThisYear` 函数中获取到从 2004 年 1 月 1 日到 2004 年 4 月 22 日之间的天数。

```
int getDaysBeforeThisDayInThisYear(int year, int month, int day){
    byte leap_plus=0;
    if (isLeapYear(year)) leap_plus=1;
    // if this year is leap year, add one to leap_plus
    switch(month){
        case 1: return day;
        case 2: return 31+day;
        case 3: return 59+day+leap_plus;
        case 4: return 90+day+leap_plus;
        case 5: return 120+day+leap_plus;
        case 6: return 151+day+leap_plus;
        case 7: return 181+day+leap_plus;
        case 8: return 212+day+leap_plus;
        case 9: return 243+day+leap_plus;
        case 10: return 273+day+leap_plus;
        case 11: return 304+day+leap_plus;
        case 12: return 334+day+leap_plus;
    } // switch month
    return -1;
}
```


3. 把上述两步中的结果加起来，就得到了所谓的“距离”。进而就可以计算出今天是星期几。

```
int getDayOfWeek(int year, int month, int day){
    int days = getDaysBeforeThisDayInThisYear(year, month, day);
    days += getDaysBeforeThisDayInPastYear(year);
    return (days+4)%7+1;
}
```

4. 人机交互性设计见 4.4

4.3 打印日历解决方案

由于月份和月份之间星期几并不会被中断，所以一个星期会把上个月和这个月联系起来，因此我考虑以星期为单位来输出日历，而不是以月份为单位。

1. 打印出一行也就是一个星期，如果是月初还要打印出对应的表头。

```
void printOneLine(int first_number, int pos_next_line, int month){
    for(int i=0; i<7; i++){
        System.out.print((first_number+i)+"\t");
        if (pos_next_line-1==i) {
            System.out.println();
            System.out.println("\n\t\t "+month2Words(month+1)+":\nMon.\tTue.\tWed.\tTur.\tFri.\tSat.\tSun.");
            for(int j=0; j<pos_next_line; j++)
                System.out.print("\t");
            for(int j=1; j<=7-pos_next_line; j++)
                System.out.print(j+"\t");
            break;
        } // If there is a \n
    } // print loop
    System.out.println();
}
```

2. 打印出整个日历。(这里操作其实很复杂，做完以后思考，其实不如按照月份为单位来打印更方便。而且这样没有办法仅仅打印指定的月份。)

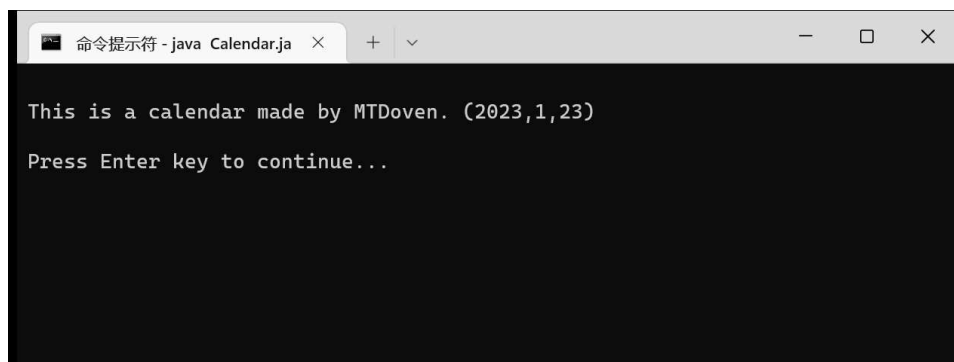
```
void printCalendars(int first_day_of_week, boolean is_leap_year){
    int day=0;
    int pos_next_line = 0;
    int month = 1;
    System.out.println("\n\t\t January:\nMon.\tTue.\tWed.\tTur.\tFri.\tSat.\tSun.");
    for(int i=1; i<first_day_of_week; i++)
        System.out.print("\t");
    for(day=1; day<=8-first_day_of_week; day++)
        System.out.print(day+"\t");
    System.out.println();
    // day=5
    while(month<=12){
        printOneLine(day, pos_next_line, month);
        day += 7;
        if (day+6 == month2Days(month, is_leap_year)){...}
        else if (day+7 <= month2Days(month, is_leap_year)){...}
        else {...}
    }
    System.out.print("\n\n");
}
```

4.4 人机交互性

0. 人机交互 main 函数部分:

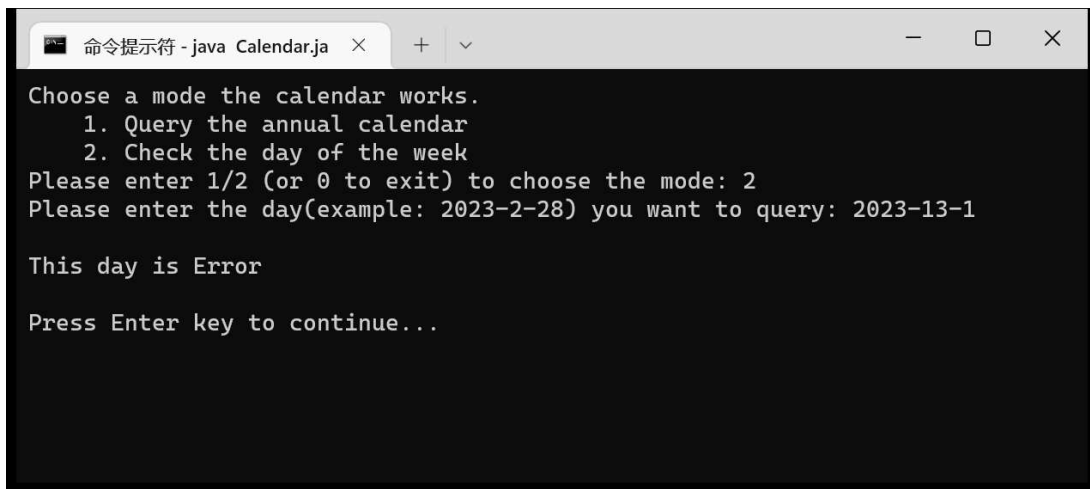
```
public static void main(String args[]){
    System.out.println("\033c");
    Scanner receiver = new Scanner(System.in);
    Calendar c = new Calendar();
    System.out.println("This is a calendar made by MTDoven. (2023,1,23)");
    while (true) {
        c.pressEnterToContinue();
        System.out.print("\033c");
        ////////////////////////////////////////////////// 0. Press any key to continue //////////////////////////////////
        System.out.print("Choose a mode the calendar works.\n"+
            "    1. Query the annual calendar\n    2. Check the day of the week\n"+
            "Please enter 1/2 (or 0 to exit) to choose the mode: ");
        int mode = receiver.nextInt();
        ////////////////////////////////////////////////// 1. Query the annual calendar //////////////////////////////////
        if (mode==1){
            System.out.print("Please enter the year you want to query: ");
            int year = receiver.nextInt();
            int week_of_day_the_first_day = c.getDayOfWeek(year, month: 1, day: 1);
            c.printCalendars(week_of_day_the_first_day, c.isLeapYear(year));
        } // 1. Query the annual calendar
        ////////////////////////////////////////////////// 2. Check the day of the week //////////////////////////////////
        else if (mode==2){
            System.out.print("Please enter the day(example: 2023-2-28) you want to query: ");
            String[] day = receiver.next().split( regex: "-|/");
            int day_of_week = c.getDayOfWeek(Integer.parseInt(day[0]),
                Integer.parseInt(day[1]),Integer.parseInt(day[2]));
            String[] weekday2words = {"Error","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"};
            System.out.println("\nThis day is "+weekday2words [day_of_week]);
        } else break;// 2. Check the day of the week
        //////////////////////////////////////////////////
    } // loop
    System.out.println("\033c");
    System.out.println("Thank you for using !!!");
    c.pressEnterToContinue();
} // static main
////////////////////////////////////////////////
```

1. 为了避免不必要的字符问题, 而且为了通用性, 整个程序采用英文交互。



```
命令提示符 - java Calendar.ja
This is a calendar made by MTDoven. (2023,1,23)
Press Enter key to continue...
```

2. 如果输入内容或各式错误，程序可以给出提示 Error，而不会给出错误结果。

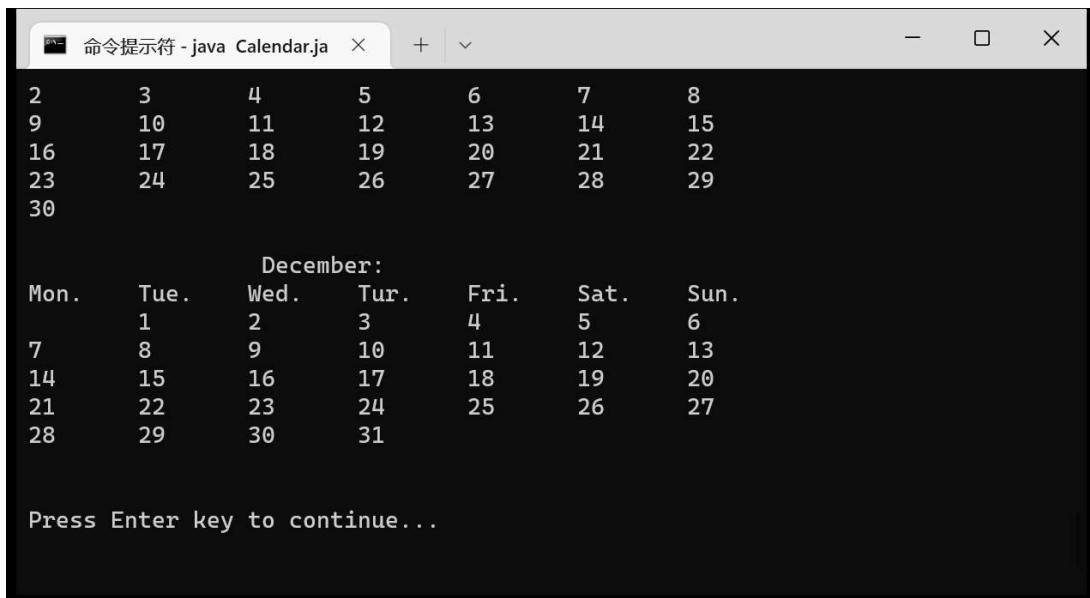


```
命令提示符 - java Calendar.java x + v
Choose a mode the calendar works.
  1. Query the annual calendar
  2. Check the day of the week
Please enter 1/2 (or 0 to exit) to choose the mode: 2
Please enter the day(example: 2023-2-28) you want to query: 2023-13-1

This day is Error

Press Enter key to continue...
```

3. 虽然是命令程序，但通过打印"\033c"实现清屏，从而显示这确实是一个程序，而不是在黑框框里敲代码的感觉。

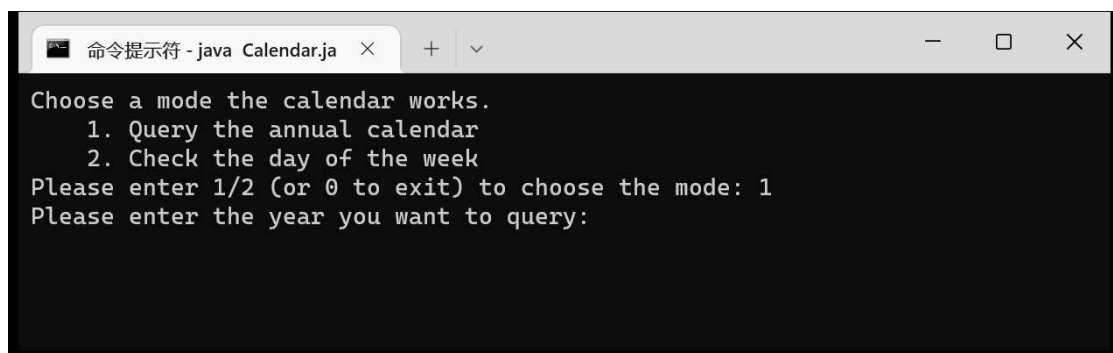


```
命令提示符 - java Calendar.java x + v
2      3      4      5      6      7      8
9      10     11     12     13     14     15
16     17     18     19     20     21     22
23     24     25     26     27     28     29
30

          December:
Mon.   Tue.   Wed.   Tur.   Fri.   Sat.   Sun.
      1      2      3      4      5      6
7      8      9      10     11     12     13
14     15     16     17     18     19     20
21     22     23     24     25     26     27
28     29     30     31

Press Enter key to continue...
```

4. 程序通过引导式交互，每一步都会给出对应的提示下一步应该做什么。



```
命令提示符 - java Calendar.java x + v
Choose a mode the calendar works.
  1. Query the annual calendar
  2. Check the day of the week
Please enter 1/2 (or 0 to exit) to choose the mode: 1
Please enter the year you want to query:
```

4.5 任务总结

1. 把一个大问题一步步分解成容易完成的小问题，然后把它们串联起来，在面向过程编程中这种 Top-Down 的编程思想尤为重要。
2. 这个程序仅仅经历过一小部分测试，而且依靠个人测试，偶然性很大。
3. 代码风格尤为重要。通过适当的注释帮助自己和别人理清思路，可以大大提高开发效率。
4. 在做日历的时候，我考虑过能否把阴历也给加入进去，但是我发现阴历的历法并不那么规律，如果要显示阴历的日期，需要在程序中写死很多数据，没有多大意义。
5. 其实对于日历这种信息量并不大的数据，完全可以计算出每一天的信息，然后存储到数据库中再来调用，而不是每次都去计算。

5. 总体总结

要学的东西还很多，要走的路还很长；
革命尚未成功，同志仍须努力。