

逻辑控制

要调整的内容

主要是作业.

1. 代码改错题.
2. 读代码, 写出运行结果题
3. 编程题.
4. 搞几个错误的代码让同学们调试.
5. **把调试单独拉出来讲**
6. 加一些编译错误的错误信息.
7. OJ 的输入输出格式处理代码要单独拎出来讲(不一定放到这里) (结合 Scanner)
8. 在后面的 IO 流的时候还要再回头来梳理 Scanner 的细节.

课件暂时不调整. 作业由郑老师和王磊老师负责.

本节目标

1. Java中程序的逻辑控制语句
2. Java中的输入输出方式
3. 掌握调试的基本方式.

1. 顺序结构

顺序结构比较简单. 像我们之前写过的代码就是顺序结构的, 按照代码书写的顺序一行一行执行.

```
System.out.println("aaa");  
System.out.println("bbb");  
System.out.println("ccc");
```

```
// 运行结果  
aaa  
bbb  
ccc
```

如果调整代码的书写顺序, 则执行顺序也发生变化

```
System.out.println("aaa");
System.out.println("ccc");
System.out.println("bbb");
```

// 运行结果

```
aaa
ccc
bbb
```

2. 分支结构

2.1 if 语句

基本语法形式1

```
if(布尔表达式){
    //条件满足时执行代码
}
```

基本语法形式2

```
if(布尔表达式){
    //条件满足时执行代码
}else{
    //条件不满足时执行代码
}
```

基本语法形式3 多分支的情况

```
if(布尔表达式){
    //条件满足时执行代码
}else if(布尔表达式){
    //条件满足时执行代码
}else{
    //条件都不满足时执行代码
}
```

代码示例1: 判定一个数字是奇数还是偶数

```
int num = 10;
if (num % 2 == 0) {
    System.out.println("num 是偶数");
} else {
    System.out.println("num 是奇数");
}
```

代码示例2: 判定一个数字是正数还是负数

```
int num = 10;
if (num > 0) {
    System.out.println("num 是正数");
} else if (num < 0) {
    System.out.println("num 是负数");
} else {
    System.out.println("num 是 0");
}
```

代码示例3: 判定某一年份是否是闰年

```
int year = 2000;
if (year % 100 == 0) {
    // 判定世纪闰年
    if (year % 400 == 0) {
        System.out.println("是闰年");
    } else {
        System.out.println("不是闰年");
    }
} else {
    // 普通闰年
    if (year % 4 == 0) {
        System.out.println("是闰年");
    } else {
        System.out.println("不是闰年");
    }
}
```

注意事项1 悬垂 else 问题

```
int x = 10;
int y = 10;
if (x == 10)
    if (y == 10)
        System.out.println("aaa");
else
    System.out.println("bbb");
```

if / else 语句中可以不加 大括号 . 但是也可以写语句(只能写一条语句). 此时 else 是和最接近的 if 匹配. 但是实际开发中我们 **不建议** 这么写. 最好加上大括号.

注意事项2 代码风格问题

```
// 风格1
int x = 10;
if (x == 10) {
    // 满足条件
} else {
    // 不满足条件
}
```

```
// 风格2
int x = 10;
if (x == 10)
{
    // 满足条件
}
else
{
    // 不满足条件
}
```

虽然两种方式都是合法的, 但是 Java 中更推荐使用风格1, { 放在 if / else 同一行.

注意事项3 分号问题

```
int x = 20;
if (x == 10); {
    System.out.println("hehe");
}

// 运行结果
hehe
```

此处多写了一个分号, 导致分号成为了 if 语句的语句体, 而 {} 中的代码已经成为了和一个 if 无关的代码块.

2.2 switch 语句

基本语法

```
switch(整数|枚举|字符|字符串){
    case 内容1 : {
        内容满足时执行语句;
        [break;]
    }
    case 内容2 : {
        内容满足时执行语句;
        [break;]
    }
    ...
    default:{
        内容都不满足时执行语句;
        [break;]
    }
}
```

代码示例: 根据 day 的值输出星期

```
int day = 1;
switch(day) {
    case 1:
```

```

        System.out.println("星期一");
        break;
    case 2:
        System.out.println("星期二");
        break;
    case 3:
        System.out.println("星期三");
        break;
    case 4:
        System.out.println("星期四");
        break;
    case 5:
        System.out.println("星期五");
        break;
    case 6:
        System.out.println("星期六");
        break;
    case 7:
        System.out.println("星期日");
        break;
    default:
        System.out.println("输入有误");
        break;
}

```

根据 switch 中值的不同, 会执行对应的 case 语句. 遇到 break 就会结束该 case 语句.

如果 switch 中的值没有匹配的 case, 就会执行 default 中的语句.

我们建议一个 switch 语句最好都要带上 default.

注意事项1 break 不要遗漏, 否则会失去 "多分支选择" 的效果

```

int day = 1;
switch(day) {
    case 1:
        System.out.println("星期一");
        // break;
    case 2:
        System.out.println("星期二");
        break;
}

// 运行结果
星期一
星期二

```

我们发现, 不写 break 的时候, case 语句会依次向下执行, 从而失去了多分支的效果.

注意事项2 switch 中的值只能是 整数 | 枚举 | 字符 | 字符串

```

double num = 1.0;
switch(num) {
    case 1.0:

```

```
        System.out.println("hehe");
        break;
    case 2.0:
        System.out.println("haha");
        break;
}
```

// 编译出错

Test.java:4: 错误: 不兼容的类型: 从double转换到int可能会有损失

```
        switch(num) {
            ^
```

1 个错误

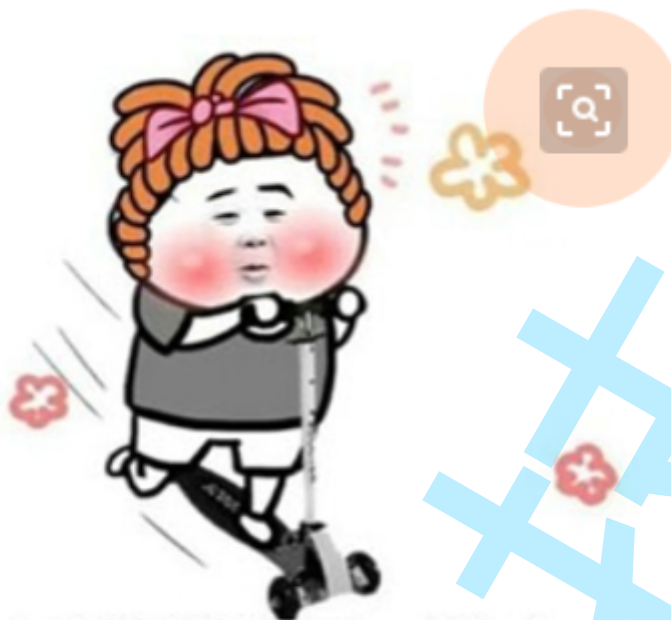
注意事项3 switch 不能表达复杂的条件

```
// 例如: 如果 num 的值在 10 到 20 之间, 就打印 hehe
// 这样的代码使用 if 很容易表达, 但是使用 switch 就无法表示.
if (num > 10 && num < 20) {
    System.out.println("hehe");
}
```

注意事项4 switch 虽然支持嵌套, 但是很丑~

```
int x = 1;
int y = 1;
switch(x) {
    case 1:
        switch(y) {
            case 1:
                System.out.println("hehe");
                break;
        }
        break;
    case 2:
        System.out.println("haha");
        break;
}
```

代码的美观程度也是一个重要的标准. 毕竟这是看脸的世界.



看脸的世界，脸小了怕你看不清

综上, 我们发现, switch 的使用局限性是比较大的

3. 循环结构

3.1 while 循环

基本语法格式:

```
while(循环条件){  
    循环语句;  
}
```

循环条件为 true, 则执行循环语句; 否则结束循环.

代码示例1: 打印 1 - 10 的数字

```
int num = 1;  
while (num <= 10) {  
    System.out.println(num);  
    num++;  
}
```

代码示例2: 计算 1 - 100 的和

```
int n = 1;
int result = 0;
while (n <= 100) {
    result += n;
    n++;
}
System.out.println(num);

// 执行结果
5050
```

代码示例3: 计算 5 的阶乘

```
int n = 1;
int result = 1;
while (n <= 5) {
    result *= n;
    n++;
}
System.out.println(num);

// 执行结果
120
```

代码示例4: 计算 $1! + 2! + 3! + 4! + 5!$

```
int num = 1;
int sum = 0;
// 外层循环负责求阶乘的和
while (num <= 5) {
    int factorResult = 1;
    int tmp = 1;
    // 里层循环负责完成求阶乘的细节.
    while (tmp <= num) {
        factorResult *= tmp;
        tmp++;
    }
    sum += factorResult;
    num++;
}
System.out.println("sum = " + sum);
```

这里我们发现, 当一个代码中带有双重循环的时候, 代码的复杂程度就大大提高了. 而比较复杂的代码就更容易出错.

后面我们会采用更简单的办法来解决这个问题.

注意事项

1. 和 if 类似, while 下面的语句可以不写 {}, 但是不写的时候只能支持一条语句. 建议还是加上 {}
2. 和 if 类似, while 后面的 { 建议和 while 写在同一行.

3. 和 if 类似, while 后面不要多写 分号, 否则可能导致循环不能正确执行.

```
int num = 1;
while (num <= 10); {
    System.out.println(num);
    num++;
}
```

// 执行结果

[无任何输出, 程序死循环]

此时 ; 为 while 的语句体(这是一个空语句), 实际的 {} 部分和循环无关. 此时循环条件 `num <= 10` 恒成立, 导致代码死循环了.

3.2 break

break 的功能是让循环提前结束.

代码示例: 找到 100 - 200 中第一个 3 的倍数

```
int num = 100;
while (num <= 200) {
    if (num % 3 == 0) {
        System.out.println("找到了 3 的倍数, 为:" + num);
        break;
    }
    num++;
}
```

// 执行结果

找到了 3 的倍数, 为:102

执行到 break 就会让循环结束.

3.3 continue

continue 的功能是跳过这次循环, 立即进入下次循环.

代码示例: 找到 100 - 200 中所有 3 的倍数

```
int num = 100;
while (num <= 200) {
    if (num % 3 != 0) {
        num++; // 这里的 ++ 不要忘记! 否则会死循环.
        continue;
    }
    System.out.println("找到了 3 的倍数, 为:" + num);
    num++;
}
```

执行到 continue 语句的时候, 就会立刻进入下次循环(判定循环条件), 从而不会执行到下方的打印语句.

3.4 for 循环

基本语法

```
for(表达式1;表达式2;表达式3){  
    循环体;  
}
```

- 表达式1: 用于初始化循环变量.
- 表达式2: 循环条件
- 表达式3: 更新循环变量.

相比于 while 循环, for 循环将这三个部分合并在一起, 写代码时不容易遗漏.

代码示例1: 打印 1 - 10 的数字

```
for (int i = 1; i <= 10; i++) {  
    System.out.println(i);  
}
```

代码示例2: 计算 1 - 100 的和

```
int sum = 0;  
for (int i = 1; i <= 100; i++) {  
    sum += i;  
}  
System.out.println("sum = " + sum);  
  
// 执行结果  
5050
```

代码示例3: 计算 5 的阶乘

```
int result = 0;  
for (int i = 1; i <= 5; i++) {  
    result *= i;  
}  
System.out.println("result = " + result);
```

代码示例4: 计算 $1! + 2! + 3! + 4! + 5!$

```
int sum = 0;
for (int i = 1; i <= 5; i++) {
    int tmp = 1;
    for (int j = 1; j <= i; j++) {
        tmp *= j;
    }
    sum += tmp;
}
System.out.println("sum = " + sum);
```

注意事项 (和while循环类似)

1. 和 if 类似, for 下面的语句可以不写 {}, 但是不写的时候只能支持一条语句. 建议还是加上 {}
2. 和 if 类似, for 后面的 { 建议和 while 写在同一行.
3. 和 if 类似, for 后面不要多写 分号, 否则可能导致循环不能正确执行.

3.5 do while 循环(选学)

基本语法

```
do{
    循环语句;
}while(循环条件);
```

先执行循环语句, 再判定循环条件.

代码示例: 打印 1 - 10

```
int num = 1;
do {
    System.out.println(num);
    num++;
} while (num <= 10)
```

注意事项

1. do while 循环最后的分号不要忘记
2. 一般 do while 很少用到, 更推荐使用 for 和 while.

4. 输入输出

4.1 输出到控制台

基本语法

```
System.out.println(msg);           // 输出一个字符串，带换行
System.out.print(msg);             // 输出一个字符串，不带换行
System.out.printf(format, msg);    // 格式化输出
```

- `println` 输出的内容自带 `\n`, `print` 不带 `\n`
- `printf` 的格式化输出方式和 C 语言的 `printf` 是基本一致的。

代码示例

```
System.out.println("hello world");

int x = 10;
System.out.printf("x = %d\n", x)
```

格式化字符串

转换符	类型	举例	
d	十进制整数	("%d", 100)	100
x	十六进制整数	("%x", 100)	64
o	八进制整数	("%o", 100)	144
f	定点浮点数	("%f", 100f)	100.000000
e	指数浮点数	("%e", 100f)	1.000000e+02
g	通用浮点数	("%g", 100f)	100.000
a	十六进制浮点数	("%a", 100)	0x1.9p6
s	字符串	("%s", 100)	100
c	字符	("%c", '1')	1
b	布尔值	("%b", 100)	true
h	散列码	("%h", 100)	64
%	百分号	("%.2f%%", 2/7f)	0.29%

这个表格没必要记住, 用到的时候根据需要查一下就行了。

4.2 从键盘输入

读入一个字符 (选学)

直接使用 `System.in.read` 可以读入一个字符. 但是需要搭配异常处理(后面会重点讲到).

```
System.out.print("Enter a Char:");
char i = (char) System.in.read();
System.out.println("your char is :"+i);
```

// 编译出错

```
Test.java:4: 错误: 未报告的异常错误IOException; 必须对其进行捕获或声明以便抛出
    char i = (char) System.in.read();
                        ^
1 个错误
```

正确写法

```
import java.io.IOException; // 需要导入 IOException 包

try {
    System.out.print("Enter a Char:");
    char i = (char) System.in.read();
    System.out.println("your char is :"+i);
} catch (IOException e) {
    System.out.println("exception");
}
```

这种方式比较麻烦, 我们不推荐使用.

使用 Scanner 读取字符串/整数/浮点数

```
import java.util.Scanner; // 需要导入 util 包

Scanner sc = new Scanner(System.in);
System.out.println("请输入你的姓名: ");
String name = sc.nextLine();
System.out.println("请输入你的年龄: ");
int age = sc.nextInt();
System.out.println("请输入你的工资: ");
float salary = sc.nextFloat();
System.out.println("你的信息如下: ");
System.out.println("姓名: "+name+"\n"+"年龄: "+age+"\n"+"工资: "+salary);
sc.close(); // 注意, 要记得调用关闭方法
```

// 执行结果

请输入你的姓名:

张三

请输入你的年龄:

18

请输入你的工资:

1000

你的信息如下:

姓名: 张三

年龄: 18

工资: 1000.0

使用 Scanner 循环读取 N 个数字

```
Scanner sc = new Scanner(System.in);
double sum = 0.0;
int num = 0;
while (sc.hasNextDouble()) {
    double tmp = sc.nextDouble();
    sum += tmp;
    num++;
}
System.out.println("sum = " + sum);
System.out.println("avg = " + sum / num);
sc.close();

// 执行结果
10
40.0
50.5
^Z
sum = 150.5
avg = 30.1
```

注意事项: 当循环输入多个数据的时候, 使用 ctrl + z 来结束输入 (Windows 上使用 ctrl + z, Linux / Mac 上使用 ctrl + d).

5. 猜数字游戏

游戏规则:

系统自动生成一个随机整数(1-100), 然后由用户输入一个猜测的数字. 如果输入的数字比该随机数小, 提示 "低了", 如果输入的数字比该随机数大, 提示 "高了", 如果输入的数字和随机数相等, 则提示 "猜对了".

参考代码

```
import java.util.Random;
import java.util.Scanner;;

class Test {
    public static void main(String[] args) {
        Random random = new Random(); // 默认随机种子是系统时间
        Scanner sc = new Scanner(System.in);
        int toGuess = random.nextInt(100);
        // System.out.println("toGuess: " + toGuess);
        while (true) {
            System.out.println("请输入要输入的数字: (1-100)");
            int num = sc.nextInt();
            if (num < toGuess) {
                System.out.println("低了");
            }
        }
    }
}
```

```
        } else if (num > toGuess) {
            System.out.println("高了");
        } else {
            System.out.println("猜对了");
            break;
        }
    }
    sc.close();
}
```

作业

1. 根据年龄, 来打印出当前年龄的人是少年(低于18), 青年(19-28), 中年(29-55), 老年(56以上)
2. 判定一个数字是否是素数
3. 打印 1 - 100 之间所有的素数
4. 输出 1000 - 2000 之间所有的闰年
5. 输出乘法口诀表
6. 求两个正整数的最大公约数
7. 计算 $1/1 - 1/2 + 1/3 - 1/4 + 1/5 - \dots + 1/99 - 1/100$ 的值。
8. 编写程序数一下 1 到 100 的所有整数中出现多少个数字9。
9. 求出0 ~ 999之间的所有“水仙花数”并输出。(“水仙花数”是指一个三位数, 其各位数字的立方和恰好等于该数本身, 如; $153 = 1 + 5 + 3^3$, 则153是一个“水仙花数”。)
10. 编写代码模拟三次密码输入的场景。最多能输入三次密码, 密码正确, 提示“登录成功”, 密码错误, 可以重新输入, 最多输入三次。三次均错, 则提示退出程序
11. 写一个函数返回参数二进制中 1 的个数 比如: 15 0000 1111 4 个 1
12. 获取一个数二进制序列中所有的偶数位和奇数位, 分别输出二进制序列。
13. 输出一个整数的每一位。
14. 完成猜数字游戏