4. 程序逻辑控制

【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. 完成猜数字游戏

1. 顺序结构

顺序结构比较简单，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

1. **语法格式1**

```
if(布尔表达式){
    // 语句
}
```

如果布尔表达式结果为true，执行if中的语句，否则不执行。

比如：小明，如果这次考试考到90分或以上，给你奖励一个鸡腿。

```
int score = 92;
if(score >= 90){
    System.out.println("吃个大鸡腿!!!");
}
```

2. 语法格式2

```
if(布尔表达式){
    // 语句1
}else{
    // 语句2
}
```

如果布尔表达式结果为true，则执行if中语句，否则执行else中语句。

比如：小明，如果这次考到90分以上，给你奖励一个大鸡腿，否则奖你一个大嘴巴子。

```
int score = 92;
if(score >= 90){
    System.out.println("吃个大鸡腿!!!");
}else{
    System.out.println("挨大嘴巴子!!!");
}
```

3. 语法格式3

```
if(布尔表达式1){
    // 语句1
}else if(布尔表达式2){
    // 语句2
}else{
```

```
    // 语句3
}
```

表达式1成立，执行语句1，否则表达式2成立，执行语句2，否则执行语句3

比如：考虑到学生自尊，不公开分数排名，因此：

- 分数在 [90, 100] 之间的，为优秀
- 分数在 [80，90) 之前的，为良好
- 分数在 [70，80) 之间的，为中等
- 分数在 [60，70) 之间的，为及格
- 分数在 [0，60) 之间的，为不及格
- 错误数据

按照上述办法通知学生成绩。

```
if(score >= 90){
    System.out.println("优秀");
}else if(score >= 80 && score < 90){
    System.out.println("良好");
}else if(score >= 70 && score < 80){
    System.out.println("中等");
}else if(score >= 60 && score < 70){
    System.out.println("及格");
}else if(score >= 0 && score < 60){
    System.out.println("不及格");
}else{
    System.out.println("错误数据");
}
```

【练习】

1. 判断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");
}
```

2. 判断一个数字是正数，负数，还是零

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

3. 判断一个年份是否为闰年

```java
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");
    }
}
```

4. **代码风格**

```java
// 风格1
int x = 10;
if (x == 10) {
    // 语句1
} else {
    // 语句2
}
// ------------------------ 推荐风格1--------------------
// 风格2
int x = 10;
```

```java
if (x == 10)
{
    // 语句1
}
else
{
    // 语句2
}
```

- 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. 代码跟紧凑。

5. **分号问题**

```java
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.

6. **悬垂 else 问题**

```java
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.

2.2 switch 语句

1. **基本语法**

```
switch(表达式){
        case 常量值1:{
                语句1;
                [break;]
                }
        case 常量值2:{
                语句2;
                [break;]
                }
        ...
        default:{
                内容都不满足时执行语句;
                [break;]
                }
}
```

2. **执行流程：**
   a. 先计算表达式的值
   b. 和case依次比较，一旦有响应的匹配就执行该项下的语句，直到遇到break时结束
   c. 当表达式的值没有与所列项匹配时，执行default
3. **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;
    }
```

【注意事项】

- 多个case后的常量值不可以重复
- switch的括号内只能是以下类型的表达式：
  - 基本类型：byte、char、short、int，注意不能是long类型
  - 引用类型：String常量串、枚举类型

```
//错误代码示例:
double num = 1.0;
switch(num) {
    case 1.0:
        System.out.println("hehe");
        break;
    case 2.0:
        System.out.println("haha");
        break;
}
```

- 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
```

- 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");
}
```

- switch 虽然支持嵌套, 但是很丑，一般不推荐~

```
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;
}
```

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

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

代码示例2: 计算 1 - 100 的和

```java
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

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

// 执
```

代码示例4: 计算 1! + 2! + 3! + 4! + 5!

```java
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, 当一个代码中带有多重循环的时候, 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;布尔表达式2;表达式3){
        表达式4;
}
```

- 表达式1: 用于初始化循环变量初始值设置，在循环最开始时执行，且只执行一次
- 表达式2: 循环条件，满则循环继续，否则循环结束
- 表达式3: 循环变量更新方式

【执行过程】

表达式1、表达式2、表达式4、表达式3 -》表达式2、表达式4、表达式3-》表达式2、表达式4、表达式3 -》...

【代码示例】

1. 打印 1 - 10 的数字

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

2. 计算 1 - 100 的和

```java
int sum = 0;
for (int i = 1; i <= 100; i++) {
    sum += i;
}
System.out.println("sum = " + sum);

// 执行结果
5050
```

3. 计算 5 的阶乘

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

4. 计算 1! + 2! + 3! + 4! + 5!

```java
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.
4. 和while循环一样，结束单趟循环用continue，结束整个循环用break
5. 表达式2如果缺失，代表该循环为死循环。

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，循环条件成立则继续执行，否则循环结束。

例如：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. 小试牛刀

1. 判定一个数字是否是素数
2. 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确好等于该数本身，如：`153＝1^3＋5^3＋3^3`，则153是一个“水仙花数”。)

完