流程控制与数组

流程控制

1.顺序结构 顺序结构就是程序从上到下一次逐行执行，没有任何判断和跳转

2.分支结构 Java中有两种分支结构：if语句和switch语句

- if语句 if语句有三种形式：
    - `if(logic expression) {statement...}`
    - `if(logic expression) {statement...} esle{statement...}`
    - `if(logic expression) {statement...} else if(logic expression) {statement...} ... else{statement...}`

  if语句的条件执行体要么是一个花括号括起来的代码块，要么是以分号结束的一行语句

- switch语句 switch语句后面的控制表达式的数据类型只能是byte、short、char、int四种整型，枚举类型和String类型，不能是boolean类型。 switch形式： `switch(expression) { case conditon1: {statement(s);break;} case conditon2:{statement(s);break;} ... case conditonN: {statement(s);break;} default:{statement(s)} }`
    - 执行顺序是先计算expression的值，然后依次与condition匹配，匹配成功则执行，如果所有condition都不匹配，则执行default后的代码块
    - case标签后的代码块可以省略花括号
    - case后的代码块必须加break语句，否则后面的代码会不经匹配全部执行

3.循环结构

- while循环 while语句形式： `[init_statement] while(test_expression){ statement; [iteration_statement] }`
- do while循环 do while循环语句形式： `[init_statement] do{ statement; [iteration_statement] }while(test_statement);`
- for循环 for循环语句形式： `for([init_statement];[test_statement];[iteration_statement]){ statement }`
    - for循环初始化表达式中可以声明多个变量，但是他们应该具有相同的数据类型
    - for循环圆括号中只有两个分号是必须的，初始化语句、循环条件、迭代语句都可以省略
    - 如果把一个循环放在另一个循环中，就可以形成循环嵌套

4.控制循环结构

- 使用break结束循环 break语句用于完全结束一个循环，跳出循环体 break语句可以结束当前循环，也可以结束当前循环的外层循环，此时需要在break后紧跟一个有意义的标签
- 使用continue忽略本次循环 continue语句用于忽略本次循环，开始下一次循环 continue语句可以忽略当前的本次循环，也可以忽略当前的外层循环的本次循环，此时需要在continue后紧跟一个有意义的标签
- 使用return结束方法 return语句直接结束当前方法

数组类型

1.理解数组 数组是一种引用类型，其既可以存储基本数据类型，也可以存储引用数据类型，只要所有元素具有相同的类型即可 数组一旦初始化完成，其所占用的内存空间将被固定下来，长度固定，无论数组元素是否清空

2.定义数组 定义数组格式：`type[] arrayName;` 数组只是一个引用类型变量，定义了数组只是定义了一个引用变量，其还未分配内存空间，需要初始化后才能使用

3.初始化数组 初始化数组两种方式：静态初始化、动态初始化

- 静态初始化 格式1：`arrayName = new type[]{element1,element2,element3,...}` 格式2：`arrayName = {element1,element2,element3...}` 格式3：`type[] arrayName = {element1,element2,element3...}`
- 动态初始化 格式1：`arrayName = new type[length]` 格式2：`type[] arrayName = new type[length]` 这里的type必须与定义数组时的type类型相同，或者是其子类 系统为数组元素分配初始值的规则是：0（byte、short、int、long）、0.0（float、double）、false（boolean）、'\u0000'（char）、null（引用类型）

4.使用数组 可以使用[]来访问数组元素，索引范围为0-arrayName.length

5.foreach循环 foreach循环能够自动遍历数组和集合的所有元素 格式：`for(variableName:array|collection){ //variableName自动迭代访问每个元素 }` foreach循环利用临时变量迭代数组元素，因此在佛reach循环中不要对数组进行赋值

6.深入数组

- 内存中的数组
  - 实际的数组对象存储在堆内存中，如果引用该数组对象的数组引用变量是一个局部变量，那么它被存储在栈内存中
  - 如果需要访问数组对象本身，只能通过数组引用变量来访问它
  - 只要类型相互兼容，就可以让一个数组变量指向另一个实际的数组
- 多维数组
  - 多维数组动态初始化：可以只指定最左边维的大小，也可以指定每一维的大小
  - 多维数组静态初始化：必须指定多个低维数组作为多维数组的初始化值