3.1 switch语句格式

```
格式：
switch(表达式){
    case 值1：
                语句体1;
                break;
    case 值2：
                语句体2;
                break;
    ...
    default：
                语句体n+1;
                [break;]
}
```

格式说明：

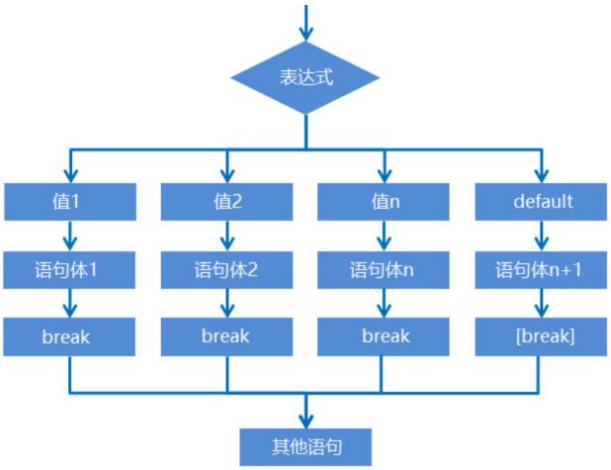
- 表达式：取值为byte、short、int、char，JDK5以后可以是枚举，JDK7以后可以是String。
- case：后面跟的是要和表达式进行比较的值。
- break：表示中断，结束的意思，用来结束switch语句。
- default：表示所有情况都不匹配的时候，就执行该处的内容，和if语句的else相似。

执行流程：

① 首先计算表达式的值。

② 依次和case后面的值进行比较，如果有对应的值，就会执行相应的语句，在执行的过程中，遇到break就会结束。

③ 如果所有的case后面的值和表达式的值都不匹配，就会执行default里面的语句体，然后程序结束掉。



注意事项：在switch语句中，如果case控制的语句体后面不写break，将出现**穿透**现象，在不判断下一个case值的情况下，向下运行，直到遇到break，或者整体switch语句结束

编写 switch 语句时，切勿忘记 break

1. for循环语句

1.1 循环结构

循环结构的组成：

初始化语句：　用于表示循环开启时的起始状态，简单说就是循环开始的时候什么样

条件判断语句：用于表示循环反复执行的条件，简单说就是判断循环是否能一直执行下去

循环体语句：　用于表示循环反复执行的内容，简单说就是循环反复执行的事情

条件控制语句：用于表示循环执行中每次变化的内容，简单说就是控制循环是否能执行下去

循环结构对应的语法：

初始化语句： 这里可以是一条或者多条语句，这些语句可以完成一些初始化操作

条件判断语句：这里使用一个结果值为boolean类型的表达式，这个表达式能决定是否执行循环体。例如：a<3

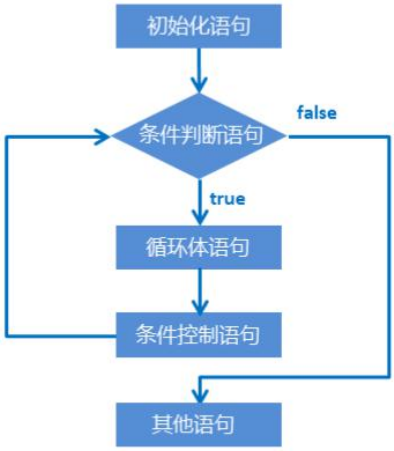
循环体语句： 这里可以是任意语句，这些语句将反复执行

条件控制语句：这里通常是使用一条语句来改变变量的值，从而达到控制循环是否继续向下执行的效果。常见i++,i--这样的操作

1.2 for循环语句格式

格式：

```
for（初始化语句;条件判断语句;条件控制语句）{
    循环体语句;
}
```



执行流程：

① 执行初始化语句

② 执行条件判断语句，看其结果是true还是false

如果是false，循环结束

如果是true，继续执行

③ 执行循环体语句

④ 执行条件控制语句

⑤ 回到②继续

```
public class Text1{
    public static void main(String[] args){
        for(int i=100;i<1000;i++) {
            int ge = i%10;
            int shi = i/10%10;
            int bai = i/10/10%10;
            if(ge*ge*ge + shi*shi*shi + bai*bai*bai == i){
                System.out.println(i);
            }
        }
    }
}
```

For 语句里，初始语句体为 int i = number

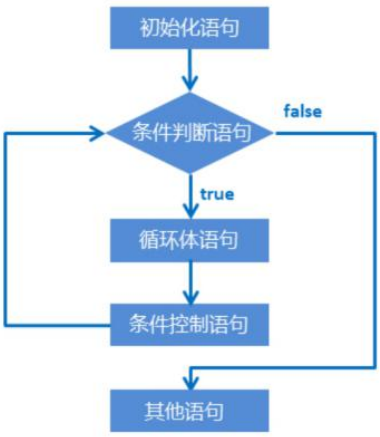
2.1 while循环语句格式

基本格式：

```
while（条件判断语句）{
    循环体语句;
}
```

完整格式：

```
初始化语句;
while（条件判断语句）{
    循环体语句;
    条件控制语句;
}
```



执行流程：

① 执行初始化语句

② 执行条件判断语句，看其结果是true还是false

如果是false，循环结束

如果是true，继续执行

③ 执行循环体语句

④ 执行条件控制语句

⑤ 回到②继续

3.1 do...while循环语句格式

基本格式：

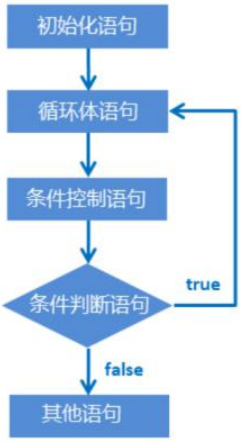
```
do {
    循环体语句;
}while(条件判断语句);
```

完整格式：

```
初始化语句;
do {
    循环体语句;
    条件控制语句;
}while(条件判断语句);
```

执行流程：

① 执行初始化语句
② 执行循环体语句
③ 执行条件控制语句
④ 执行条件判断语句，看其结果是true还是false
　　如果是false，循环结束
　　如果是true，继续执行
⑤ 回到②继续



3.2 三种循环的区别

三种循环的区别：

- for循环和while循环先判断条件是否成立，然后决定是否执行循环体（先判断后执行）
- do...while循环先执行一次循环体，然后判断条件是否成立，是否继续执行循环体（先执行后判断）

for和while的区别：

- 条件控制语句所控制的自增变量，因为归属for循环的语法结构中，在for循环结束后，就不能再次被访问到了
- 条件控制语句所控制的自增变量，对于while循环来说不归属其语法结构中，在while循环结束后，该变量还可以继续使用

死循环格式：

```
for(;;) { }
while(true) { }
do{ }while(true);
```

while的死循环格式是最常用的
命令提示符窗口中Ctrl+C可以结束死循环

4.1 跳转控制语句概述

- continue　用在循环中，基于条件控制，跳过某次循环体内容的执行，继续下一次的执行
- break　　用在循环中，基于条件控制，终止循环体内容的执行，也就是说结束当前的整个循环

5.1 循环嵌套概述

语句结构：

- 顺序语句　以分号结尾，表示一句话的结束
- 分支语句　一对大括号表示if的整体结构，整体描述一个完整的if语句
  一对大括号表示switch的整体结构，整体描述一个完整的switch语句
- 循环语句　一对大括号表示for的整体结构，整体描述一个完整的for语句
  一对大括号表示while的整体结构，整体描述一个完整的while语句
  do..while以分号结尾，整体描述一个完整的do..while语句

任何语句对外都可以看成是一句话，一句代码
分支语句中包含分支语句称为分支嵌套
循环语句中包含循环语句称为**循环嵌套**

```
for ( ;; ) {
    for( ;; ) {
        ....
    }
}
```

```
if ( 条件 ) {
    Syso...
}
```

等同于

```
if ( 条件 ) {
    if ( 条件 ) {
    }else {
    }
}
```

6.1Random的作用和使用步骤

作用：用于产生一个随机数

使用步骤：

① 导包

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

导包的动作必须出现在类定义的上面

② 创建对象

```
Random r = new Random();
```

上面这个格式里面，r是变量名，可以变，其他的都不允许变

③ 获取随机数

```
int number = r.nextInt(10);//获取数据的范围：[0,10) 包括0,不包括10
```

上面这个格式里面，number是变量名，可以变，数字10可以变。其他的都不允许变