一、程序流程

1.0 顺序结构[默认流程]

```
public class Test {
public static void main(String[] args) {
    System.out.println("A");
    System.out.println("B");
    System.out.println("C");
    }
}
//如果你没有写其他的结构，按照代码的先后顺序，依次执行程序中大多数的代码都是这样执行的。
```

2.0 分支结构

- 两种：if、switch
  - if其实在功能上远远强大于switch，效率较于switch要低
  - if适合做区间匹配。
  - switch适合做值匹配，代码优雅,效率高

3.0 循环结构

- 三种：for、while、do while

二、分支

1.0 switch

- 概述

```
值匹配，性能好，功能没有if强大（if可以范围匹配）
```

- 代码体现

```
        public static void main(String[] args) {
        //switch天然与枚举结合进行信息分类，对入参类型进行校验，但只适合信息标志
        //如果需要记录一些信息的就使用常量
        choose(CaseType.FIFTH);
    }
    private static void choose(CaseType caseType) {
```

```
        //这里可以是byte short int char String 枚举 不支持小数和long（小数不精确不利于比对，long太大了，没有可能有这么多的标志位）
        switch (caseType) {
            //这里只能是字面量或枚举或常量，不能重复
            case FIFTH:
                System.out.println("1");
                //不加beak会发生switch穿透，有时候会有用到switch穿透
                break;
            case SECOND:
                System.out.println("2");
                break;
                break;
                System.out.println("4");
                break;
            case FIRST:
                System.out.println("5");
                break;
        }
    }
```

- switch配合枚举类使用
  - 定义枚举类

```
public enum Season{
    SPRING,SUMMER,AUTUMN,WINTER;
}
```

  - 枚举类反编译

```
Compiled from "Season. java"
    public final class Season extends java.lang. Enum<Season> {
        public static final Season SPRING=new Season ();
        public static final Season SUMMER = new Season ();
        public static final Season AUTUMN = new Season ();
        public static final Season WINTER = new Season ();
        public static Season [] values ();
        public static Season valueOf (java.lang.String);
    }
```

  - 枚举类java代码形式

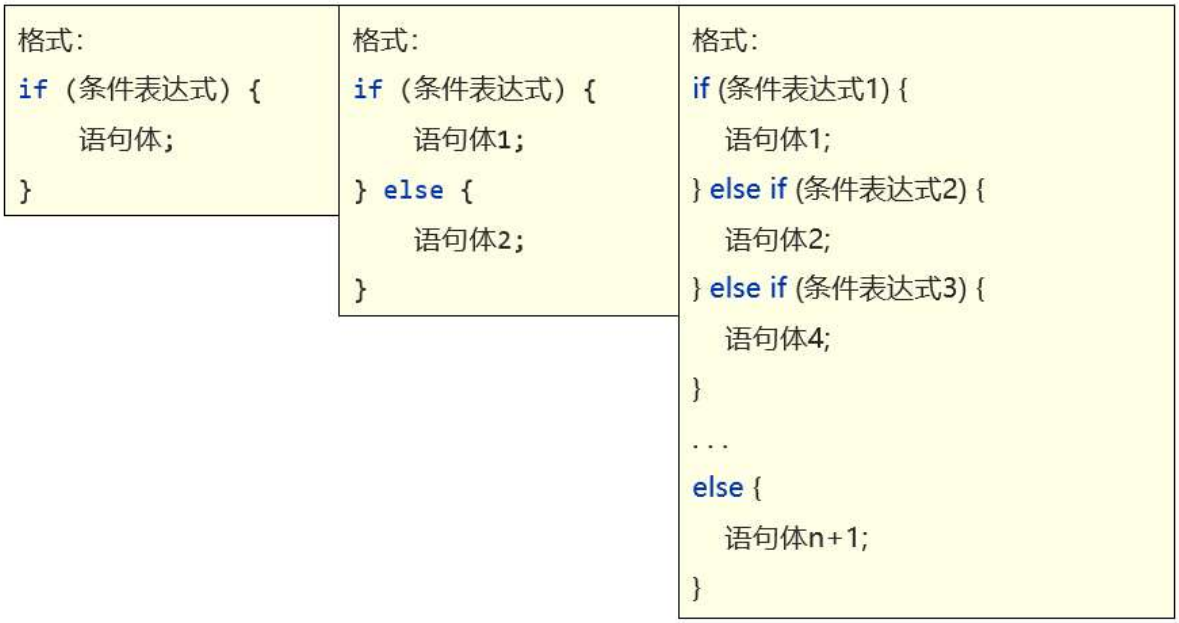
```
public final class Season extends Enum{
      public static final Season SPRING;
      public static final Season SUMMER;
      public static final Season AUTUMN;
      public static final Season WINTER;

    public Season(String name,int orgin){
        super(name,orgin);
    }
    static{
        SPRING=new Season("SPRING",0);
        SUMMER=new Season("SUMMER",1);
        AUTUMN=new Season("AUTUMN",2);
        WINTER=new Season("WINTER",3);
```

```
    }
    .........
}
```

- 枚举特性

```
1. 继承了java.lang下的Enum类的类就是枚举类
2. 枚举类是最终类不可以继承
3. 枚举类的构造器都是私有的，即是不能对外创建对象；
4. 枚举类相当于多例模式,可以通过枚举实现单例
5. 枚举类第一行默认是罗列所有对象名称,修饰符是public static final
6. 每一个名称都代表一个该枚举类的实例，且类型为常量
7. 选择枚举类型作为信息的标志与分类比选择常量更为严谨
```

2.0 if



三、循环

1.0 for

- 格式

```
for(初始语句;循环条件;迭代语句){
    //循环逻辑
}
```

- 注意

```
1. 初始化语句、循环条件、迭代语句如果有多个，用逗号隔开 但分号不能省略
2. 用break打破循环和switch
3. 用continue结束本次循环
```

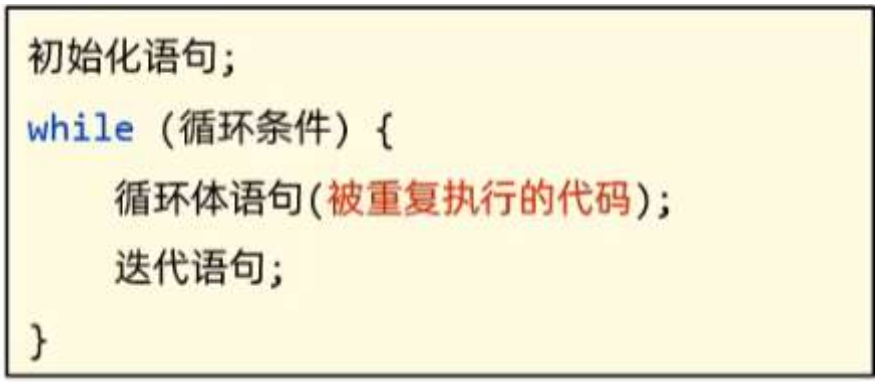
- 代码实现

```
    @Test
    public void testFor() {
        int sum = 0;
        for (int i = 2; i <= 102; i += 2) {
            sum += i;
        }
        System.out.println("2-102求和：" + sum);
    }
```

- 水仙花数

```
    /*
     * 水仙花（每一位上的数的三次方相加等于原来的数）
     * */
    @Test
    public static void testFor_() {
        final Scanner sc = new Scanner(System.in);
        System.out.println("请输入数字");
        final String s = sc.nextLine();
        final String[] strings = s.split(",");
        for (String s1 : strings) {
            //当字符串为不符合数字特征的字符时，数字转换异常
            //java.lang.NumberFormatException（数字转换异常）
            //静态方法valueOf底层调用的是，静态方法parseInt，使用valueOf的目的是方便记忆
            yes(Integer.valueOf(s1));
            // Integer.parseInt(s);//将字符串转为整数
java.lang.NumberFormatException（数字转换异常）
        }
    }

    private static void yes(Integer valueOf) {
        if (valueOf / 100 != 0) {
            //通过取模得到的是末尾数值，通过除法控制末位
            final int bai = valueOf / 100;//百位
            final int ge = valueOf % 10;//个位
            final int shi = valueOf / 10 % 10;//十位
            if (bai * bai * bai + ge * ge * ge + shi * shi * shi == valueOf) {
                System.out.println(valueOf + "是水仙花");
            } else {
                System.out.println(valueOf + "不是水仙花");
            }
        }
    }
================================================================================
请输入数字
153,370,371,111
153是水仙花
370是水仙花
371是水仙花
111不是水仙花
```

2.0 while



2、什么时候用for循环，什么时候用while循环？
功能上是完全一样的，for能解决的while也能解决，反之亦然。
使用规范是：知道循环几次：使用for；不知道循环几次建议使用：while

- 代码实现(折叠纸张达到珠穆朗玛峰高度)

```
@Test
    public void test_() {
        final double mountain = 8848860;
        double paper = 0.1;

        //当不知道次数是用while，确定次数时用for
    /*    while (paper < mountain) {
            paper *= 2;
        }*/

        //不专业
        for (; paper < mountain; ) {
            paper *= 2;
        }
        System.out.println(paper);
    }
```

3.0 do...while

```
初始化语句;
    do{
        循环语句;
        迭代语句;
    }while（循环条件）;
```

（第一次先执行后判断）

4.0 死循环

```
    @Test
    public void test() {
  /*      for (; ; ) {
            System.out.println("1");
        }*/

    /*    while (true) {
```

```
            System.out.println("1");
        }*/

        do {
            System.out.println("2");
        } while (true);
    }
```

5.0 跳转语句

```
break :
        只能用于结束所在循环，
        或者结束所在switch分支的执行。
        不能用break打破if；
        break OUT;用于打破外层循环；将OUT放在外层循环语句的上一行
continue :
        只能在循环中进行使用。
```

六、随机数Random类

```
    @Test
    public void test1() {
        final Random random = new Random();
        for (int i = 0; i < 100; i++) {
            //加减法
            //比如42-55
            //直接 r.nextInt(x)+42  , 然后55-42=13 ,  包前不包后 x=14
            //r.nextInt(14)+42
            final int r = random.nextInt(10) + 1;//1-----9+1
            System.out.println(r);
        }
    }
```