Java流程控制语句2

- What?Why?How?

为什么需要循环

- 1、张浩Java考试成绩未达到自己的目标。为了表明自己勤奋学习的决心，他决定写一百遍“好好学习，天天向上！”

100条

```
System.out.println("第1遍写：好好学习，天天向上！");
System.out.println(“第2遍写：好好学习，天天向上！");
……
System.out.println(“第100遍写：好好学习，天天向上！");
```

为什么需要循环

- 没有使用循环结构

```
System.out.println("第1遍写：好好学习，天天向上！");
System.out.println("第2遍写：好好学习，天天向上！");
System.out.println("第3遍写：好好学习，天天向上！");
System.out.println("第4遍写：好好学习，天天向上！");

……
System.out.println("第9999遍写：好好学习，天天向上！");
System.out.println("第10000遍写：好好学习，天天向上！");
```

使用while循环

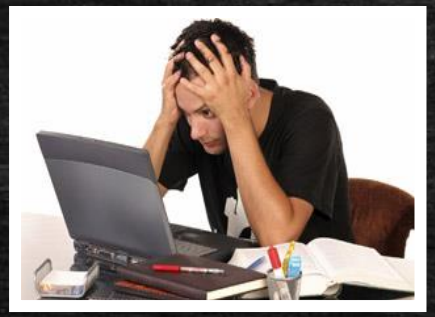
```
int i = 1;
while ( i <= 100 ){

    System.out.println("第" +i+ "遍写：好好学习，天天向上！");
    i ++;
}
```

什么是循环

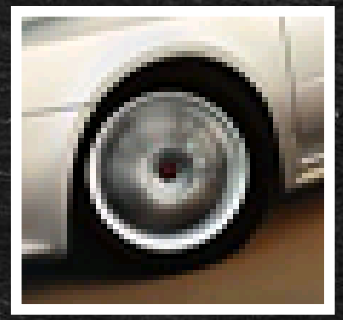
- 生活中的循环

打印50份试卷

10000米赛跑



锲而不舍地学习

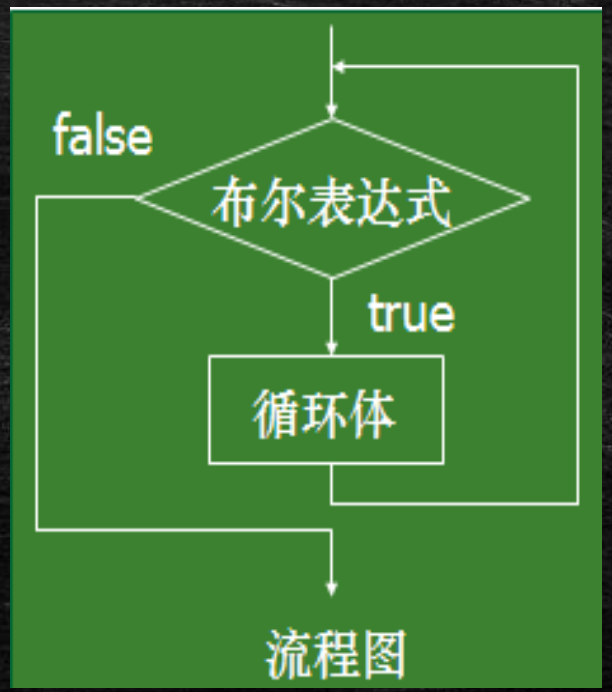


旋转的车轮

- 循环结构的特点

while循环

- 在循环刚开始时，会计算一次“布尔表达式”的值，若条件为真，执行循环体。而对于后来每一次额外的循环，都会在开始前重新计算一次。
- 语句中应有使循环趋向于结束的语句，否则会出现无限循环——"死"循环。



while循环

```
public class WhileTest {
        public static void main(String[] args) {
                int  i = 0;
                int  sum = 0;
                while (i <= 100) {
                        sum += i;   //sum = sum+i;
                        i++;
                }
                System.out.println("Sum= " + sum);
        }
}
```

循环结构都由如下四个结构组成：
初始化、条件判断、循环体、迭代

上机练习1—求100以内的偶数和

- 需求说明：
  - 编程实现：计算100以内（包括100）的偶数之和
  - 观察每一次循环中变量值的变化
- 实现思路：
  - 1、声明整型变量num和sum
  - 2、循环条件：num<=100
  - 3、循环操作：累加求和

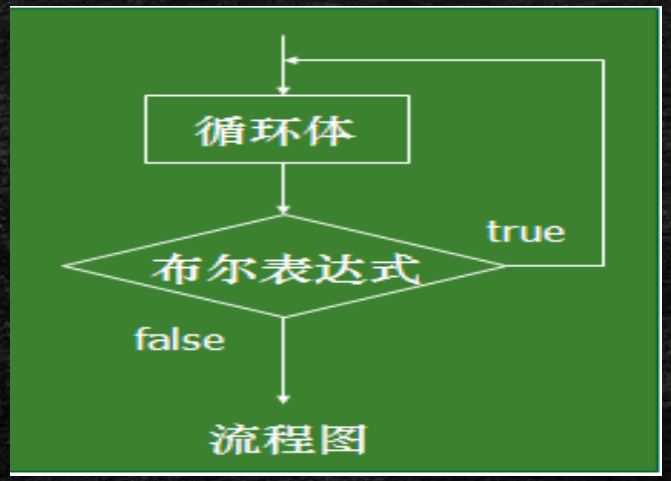
上机练习2—购物结算

- **需求说明：**
  - 循环输入商品编号和购买数量
  - 当输入n时结账
  - 结账时计算应付金额并找零

```
*****************************************
请选择购买的商品编号：
1.T 恤          2.网球鞋          3.网球拍
*****************************************

请输入商品编号：1
请输入购买数量：2
T 恤￥245.0      数量 2  合计 ￥490.0
是否继续（y/n）y

请输入商品编号：2
请输入购买数量：1
网球鞋￥570.0    数量 1  合计 ￥570.0
是否继续（y/n）y

请输入商品编号：3
请输入购买数量：2
网球拍￥320.0    数量 2  合计 ￥640.0
是否继续（y/n）n

折扣：0.8
应付金额：1360.0
实付金额：1200
您输入的金额小于应付金额，请重新输入：1400
找钱：40.0
```

do-while循环

- do-while:
  - 先执行，后判断。
- while:
  - 先判断，后执行。




```
int a = 0;
while(a<0){
        System.out.println(a);
        a++;
}
System.out.println("-----");
a=0;
do{
        System.out.println(a);
        a++;
} while (a<0);
```

While和dowhile的区别：
Dowhile总是保证循环体会被至少执行一次！这是他们的主要差别

为什么使用for循环

- 回顾问题：输出100次“好好学习！”

使用while循环结构

```
int i=0;
while(i<100){
   System.out.println("好好学习！");
   i++;
}
```

特点：循环次数固定

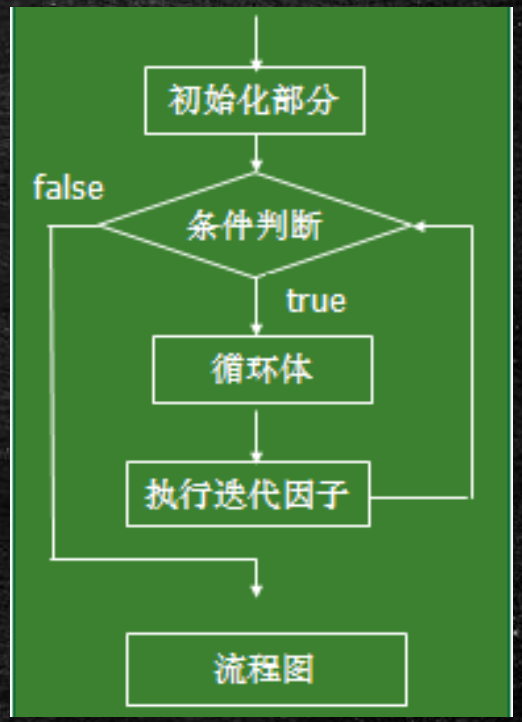
使用for循环结构

```
for(int i=0;i<100;i++){
   System.out.println("好好学习！");
}
```

for比while更简洁

for循环

- for循环语句是支持迭代的一种通用结构，是最有效、最灵活的循环结构
- 语法形式
  - for (初始表达式;布尔表达式;步进) {
    循环体;
  - }
- 注意事项
  - for循环在执行条件测试后，先执行程序部分，再执行步进。
  - 在for语句的初始化部分声明的变量，其作用域为整个for循环体
  - “初始化”和“循环条件表达式”部分可以使用逗号来执行多个操作
  - 如果三个部分都为空语句（分号不能省），相当于一个无限循环



上机练习3—计算顾客年龄比例

- 计算各层次的顾客比例

- **需求说明：**
  - 商场对顾客的年龄层次进行调查

```
请输入第1位顾客的年龄： 20
请输入第2位顾客的年龄： 34
请输入第3位顾客的年龄： 54
请输入第4位顾客的年龄： 12
请输入第5位顾客的年龄： 34
请输入第6位顾客的年龄： 56
请输入第7位顾客的年龄： 65
请输入第8位顾客的年龄： 12
请输入第9位顾客的年龄： 34
请输入第10位顾客的年龄： 32
30岁以下的比例是： 30.0%
30岁以上的比例是： 70.0%
```

课堂练习

- 用while和for循环分别计算100以内奇数和偶数的和，并输出。
- 用while和for循环输出1-1000之间能被5整除的数，且每行输出3个。
- 使用循环分别实现将10进值整数变成二进制数
- 编程求：1！+2！+……+10!

跳转语句---break和continue

- 在任何循环语句的主体部分，均可用break控制循环的流程。break用于强行退出循环，不执行循环中剩余的语句。(break语句还可用于多支语句switch中)
- continue 语句用在循环语句体中，用于终止某次循环过程，即跳过循环体中尚未执行的语句，接着进行下一次是否执行循环的判定。

生成0-100随机数，直到生成88为止，停止循环！

```
int total = 0;
 System.out.println(“Begin”);
 while(true) {
    total++;
    int i = (int)Math.round(100 *  Math.random());
    if(i == 88) break;
 }
 System.out.println("Game over, used " + total + " times.");
```

把100~150之间不能被3整除的数输出：

```
for (int i = 100; i < 150; i++) {
    if (i % 3 == 0)

            continue;

            System.out.println(i);
}
```

为什么需要break语句

- 回顾break用于switch语句
- 描述4000米长跑比赛

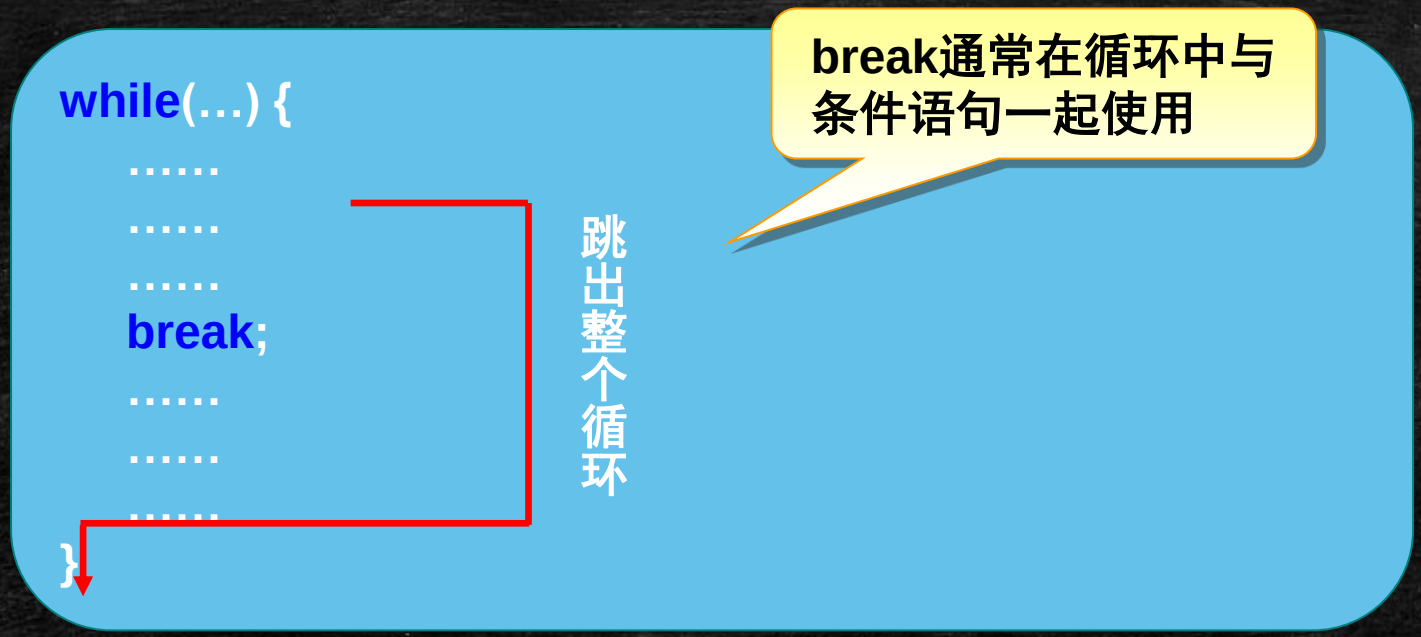
```
int i = 2;
switch(i){
        case 1:
            System.out.println("星期一");
            break;
        case 2:
            System.out.println("星期二");
            break;
 }

//其他语句
```

```
for (int i = 0; i<10; i++) {
        if ( 不能坚持 ) {
                break;     //退出比赛
        }

}
```

遇到break，立即跳出switch语句

什么是break语句

- break：改变程序控制流
  - 用于do-while、while、for中时，可跳出循环而执行循环后面的语句



如何使用break语句2-1

3、循环录入某学生5门课的成绩并计算平均分，如果某分数录入为负，停止录入并提示录入错误

循环录入成绩，判断录入正确性：录入错误，使用break语句立刻跳出循环；否则，累加求和

```
输入学生姓名：张浩
请输入第1门课的成绩：  23
请输入第2门课的成绩：  4
请输入第3门课的成绩：  -90
抱歉，分数录入错误，请重新进行录入！


输入学生姓名：张浩
请输入第1门课的成绩：  43
请输入第2门课的成绩：  56
请输入第3门课的成绩：  76
请输入第4门课的成绩：  89
请输入第5门课的成绩：  76
张浩的平均分是：68
```

如何使用break语句2-2

```
……
 for(int i = 0; i < 5; i++){       //循环5次录入5门课成绩
        System.out.print("请输入第" + (i+1) + "门课的成绩：  ");
        score = input.nextInt();
        if(score < 0){       //输入负数
                isNegative = true;
                break;
        }
       sum = sum + score;            //累加求和
}
…循环外的语句…
```

对录入的分数进行判断，如果小于0，标记出错状态，并立即跳出整个for循环

小结2

- 1~10之间的整数相加，得到累加值大于20的当前数

提示
1、使用循环进行累加，从1到10
2、判断累加值是否大于20
3、如果大于20，则跳出循环，并打印当前值

为什么需要continue语句

- 4、循环录入Java课的学生成绩，统计分数大于等于80分的学生比例

```
输入班级总人数: 5
请输入第1位学生的成绩:  43
请输入第2位学生的成绩:  45
请输入第3位学生的成绩:  67
请输入第4位学生的成绩:  8
请输入第5位学生的成绩:  98
80分以上的学生人数是:  1
80分以上的学生所占的比例为: 20.0%
```

1、通过循环，获得分数大于等于80分的学生人数num

2、判断：如果成绩<80，不执行num++，直接进入下一次循环

什么是continue语句

- continue ： 只能用在循环里
- continue 作用：跳过循环体中剩余的语句而执行下一次循环

```
while(...) {
    ......
    ......
    ......
    continue;
    ......
    ......
}
```

继续下一次循环

通常与条件语句一起使用，加速循环

```
for(int i = 0; i<10;i++){
    跑400米;
    if（！口渴）{
      continue; //不喝水，继续跑
    }
    接过水壶，喝水;
}
```

如何使用continue语句

```
for (int i = 0; i < total; i++) {
        System.out.print("请输入第" + (i + 1) + "位学生的成绩： ");
        score = input.nextInt();
        if (score < 80) {
                continue;
        }
        num++;
}
System.out.println("80分以上的学生人数是： " + num);
double rate = (double) num / total * 100;
System.out.println("80分以上的学生所占的比例为：" + rate + "%");
```

对录入的分数进行判断，如果小于80，跳出本次循环，执行下一次循环

对比break和continue

- 使用场合
  - break可用于switch结构和循环结构中
  - continue只能用于循环结构中
- 作用（循环结构中）
  - break语句终止某个循环，程序跳转到循环块外的下一条语句。
  - continue跳出本次循环，进入下一次循环

上机练习4—循环录入会员信息

- 需求说明：
  - 循环录入3位会员的信息
  - 会员号合法，显示录入信息；
  - 否则显示录入失败
- 实现思路：
  - 1、循环录入3位会员信息。
  - 2、会员号无效，
  - 利用continue
  - 实现程序跳转

```
MyShopping管理系统 > 客户信息管理 > 添加客户信息

请输入会员号（<4位整数>）：123
请输入会员生日（月/日<用两位整数表示>）：09/23
请输入会员积分：30
客户号123是无效会员号!
录入信息失败

请输入会员号（<4位整数>）：1234
请输入会员生日（月/日<用两位整数表示>）：09/11
请输入会员积分：23
您录入的会员信息是：
1234 09/11 23

请输入会员号（<4位整数>）：2345
请输入会员生日（月/日<用两位整数表示>）：08/01
请输入会员积分：26
您录入的会员信息是：
2345 08/01 26

程序结束!
```

跳转语句---return

- return语句从当前方法退出，返回到调用该方法的语句处，并从该语句的下条语句处继续执行程序。
- 返回语句的两种格式（具体到方法时详细讲解）
  - 1、return expression
    - 返回一个值给调用该方法的语句。
    - 返回值的数据类型必须和方法声明中的返回值类型一致或是精度低于声明的数据类型。
  - 2、return
    - 当方法声明中用void声明返回类型为空时，应使用这种返回类型，它不返回任何值。

跳转语句总结

- break
  - switch语句
  - 循环语句
- continue
  - 循环语句
- return
  - 任何语句中，结束当前方法，和循环其实没有什么关系

多重循环

- 三种循环方式
  - while
  - do-while
  - for

- 多重循环（循环嵌套）
  - 一个循环体内又包含另一个完整的循环结构
  - 任何两种循环都可以相互嵌套
  - 可以任意层次循环，但是一般不超过3层

- 多重循环执行过程
  - 外层循环变量变化一次，内层循环变量要变化一遍

```
for(循环条件1){
    //循环操作1
    for(循环条件2){
        //循环操作2
    }
}
```

```
while(循环条件1){
    //循环操作1
    for(循环条件2){
        //循环操作2
    }
}
```

多重循环

- 打印矩形(4*5)
- 打印平行四边形()
- 打印等腰三角形
- 打印菱形
- 等边三角形可以实现吗?

多重循环

- 多重循环中使用continue
  - 示例
    - 3个班级各4名学员参赛，计算每个班级参赛学员平均分，统计成绩大于85分学员数
  - 思路
    - 外层循环控制班级
    - 内层循环控制某个班级的每名学生
    - 成绩不大于85,continue
    - 成绩大于85，学员数加1

课堂作业

- 百钱买百鸡
- 斐波那契数列
- 输出九九乘法表：

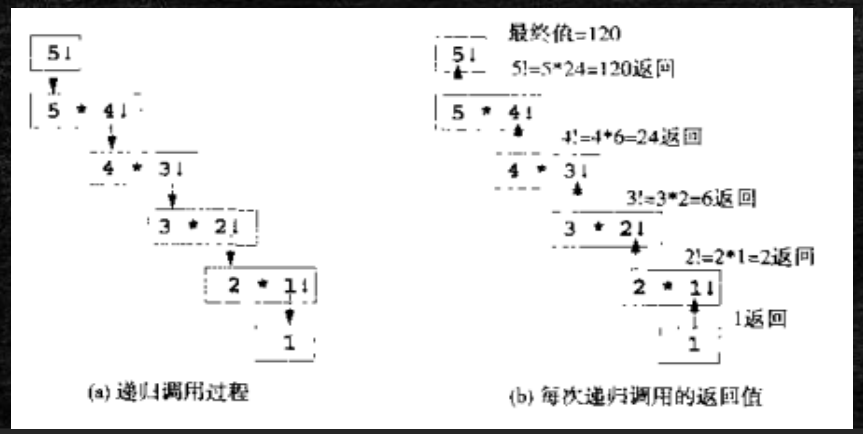
递归算法

- 什么是递归（recursion）
  - 程序调用自身的编程技巧称为递归。
  - 一个过程或函数在其定义或说明中有直接或间接调用自身的一种方法

- 递归问题的特点
  - 一个问题可被分解为若干层简单的子问题
  - 子问题和其上层问题的解决方案一致
  - 外层问题的解决依赖于子问题的解决

递归结构

- **递归结构包括两个部分：**
  - **递归结束条件**。解答：什么时候不调用自身方法。如果没有条件，将陷入死循环。
  - **递归体**。解答：什么时候需要调用自身方法。
- 递归示例
  - 使用递归求n!
  - 使用实现斐波那契数列




```
public class A {
	static long  factorial(int n){
		if(n==1){
			return 1;
		}else{
			return n*factorial(n-1);
		}
	}
	public static void main(String[] args) {
		long d1 = System.currentTimeMillis();
		System.out.println("阶乘的结果: ",factorial(10));
		long d2 = System.currentTimeMillis();
		System.out.println("递归费时： ",d2-d1);
		 //耗时：32ms
	}
}
```

递归算法

- 递归的优点
  - 简单的程序

- 递归的缺点
  - 但是递归调用会占用大量的系统堆栈，内存耗用多，
  - 在递归调用层次多时速度要比循环慢的多

- 递归的使用场合
  - 任何可用递归解决的问题也能使用迭代解决。
  - 当递归方法可以更加自然地反映问题，并且易于理解和调试，并且不强调效率问题时，可以采用递归；
  - 在要求高性能的情况下尽量避免使用递归，递归既花时间又耗内存。

总结

- 选择结构
  - if语句　单、双、多分支选择结构，等值、不等值判断均可
  - switch语句　只有多分支选择结构 只针对等值判断

- 循环结构
  - while循环　　先判断再循环　　适合循环次数不固定情况
  - do-while循环　　先循环再判断　　适合循环次数不固定情况
  - for循环　　适合循环次数固定情况

- 循环跳转语句
  - break　跳出本层循环，跳出外层循环需要结合标签或符号位实现
  - continue　提前结束本次循环
  - return　结束当前方法

总结

- 多重循环
  - 任何两种循环都可以相互嵌套
  - 外层循环变量变化一次，内层循环变量要变化一遍
- 递归
  - 程序调用自身的编程技巧称为递归。
  - 递归简单，但是内存耗用多，速度要比循环慢
  - 任何可用递归解决的问题也能使用循环解决，反之不见得