程序流程控制

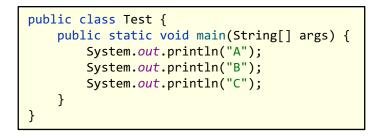




程序执行的几种常见形式



程序默认执行流程





判断条件来选择某段程序执行

if switch



重复执行某段程序多次

for while do-while



> 分支结构

- ♦ if
- switch
- ◆ switch的注意事项
- ◆ switch的穿透性
- > 循环结构
- ▶ 跳转关键字: break、continue
- > 案例技术:随机数Random类



- > 分支结构
- > 循环结构
 - ◆ for循环
 - ◆ for循环案例1
 - ◆ for循环案例2
 - ◆ for循环案例3
 - ◆ while循环
 - ◆ while循环案例
 - ◆ do-while循环
 - ◆ 死循环
 - ◆ 循环嵌套
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类



> 分支结构

- ♦ if
- switch
- ◆ switch的注意事项
- ◆ switch的穿透性
- > 循环结构
- ▶ 跳转关键字: break、continue
- > 案例技术:随机数Random类



If分支

● 根据判定的结果(真或假)决定执行某个分支的代码

If分支的作用







If分支有三种格式

```
格式1:
if (条件表达式) {
语句体;
}
```

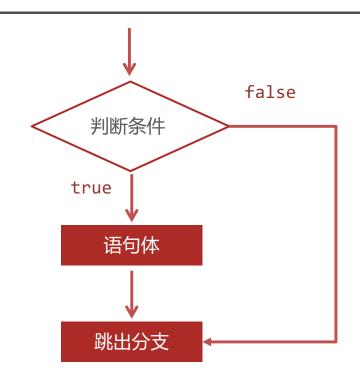
```
格式2:
if (条件表达式) {
    语句体1;
} else {
    语句体2;
}
```

```
格式3:
if (条件表达式1) {
 语句体1;
} else if (条件表达式2) {
 语句体2;
} else if (条件表达式3) {
 语句体3;
else {
 语句体n+1;
```



if 第一种格式

```
格式:
if (条件) {
    语句体;
}
...
```



执行流程:

① 首先判断条件表达式的结果,如果为true执行语句体,为 false 就不执行语句体。

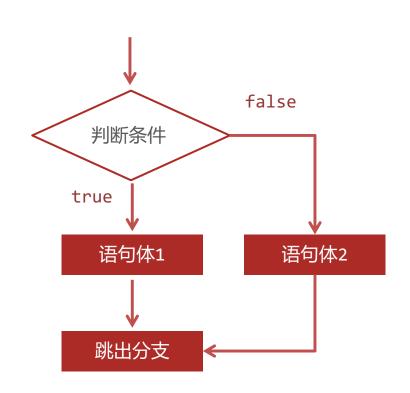
注意事项:

● if 语句中,如果大括号控制的只有一行代码,则大括号可以省略不写。



if 第二种格式

```
格式:
if (条件) {
    语句体1;
} else {
    语句体2;
}
```



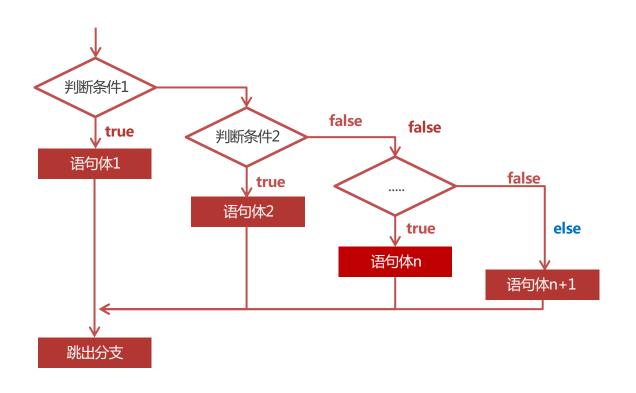
执行流程:

① 首先判断条件表达式的结果,如果为true执行语句体1,为 false 就执行语句体2。



if 第三种格式

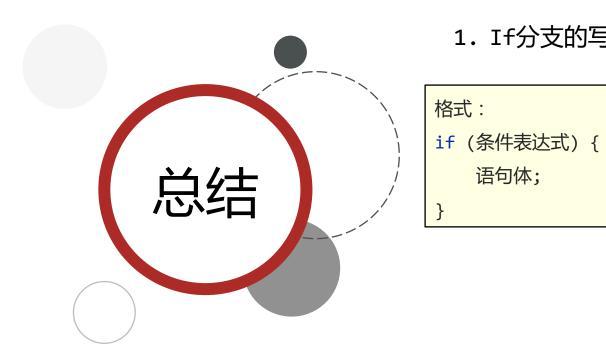
```
格式:
if (条件1) {
 语句体1;
} else if (条件2) {
  语句体2;
} else if (条件3) {
  语句体4;
else {
  语句体n+1;
```



执行流程:

- ① 先判断条件1的值,如果为true则执行语句体1,分支结束;如果为false则判断条件2的值
- ② 如果值为true就执行语句体2,分支结束;如果为false则判断条件3的值
- ③ ...
- ④ 如果没有任何条件为true,就执行else分支的语句体n+1。





1. If分支的写法有几种,各有什么特点?

语句体;

```
格式:
                   格式:
                   if (条件表达式1) {
if (条件表达式) {
   语句体1;
                    语句体1;
} else {
                   } else if (条件表达式2) {
   语句体2;
                    语句体2;
                   } else if (条件表达式3) {
                    语句体4;
                   else {
                    语句体n+1;
```





考试奖励

需求:键盘录入考试成绩,根据成绩所在的区间,程序打印出不同的奖励机制







考试奖励

需求:键盘录入考试成绩,根据成绩所在的区间,程序打印出不同的奖励机制

分析:

① 键盘录入考试成绩

③ 为每种判断设置对应的条件

② 由于奖励种类较多,属于多种判断,采用 if...else...if格式实现

④ 为每种判断设置对应的奖励

```
System.out.println("山地自行车一辆");
System.out.println("游乐场玩一次");
System.out.println("变形金刚玩具一个");
System.out.println("胖揍一顿");
```

注意事项:正确数据、边界数据、错误数据





密码校验

需求: 键盘录入用户密码, 如果密码为 111111, 程序输出密码正确, 否则输出密码有误

分析:

- ① 使用Scanner录入用户输入的密码,并使用变量接受
- ② 使用 if...else 组织程序逻辑





> 分支结构

- ◆ If分支
- ◆ switch分支
- ◆ switch使用的注意事项
- ◆ switch穿透性
- > 循环结构
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类



switch分支

● 也是匹配条件去执行分支,适合做值匹配的分支选择,结构清晰,格式良好

```
switch(表达式){
   case 值1:
      执行代码...;
      break;
   case 值2:
      执行代码...;
      break;
   case 值n-1:
      执行代码...:
      break;
   default:
      执行代码n;
```

执行流程:

- ① 先执行表达式的值,拿着这个值去与case后的值进行匹配。
- 匹配哪个case的值为true就执行哪个case, 遇到break就跳出switch分支。
- ③ 如果case后的值都不匹配则执行default代码。

switch案例

● 周一: 埋头苦干, 解决bug

● 周二:请求大牛程序员帮忙

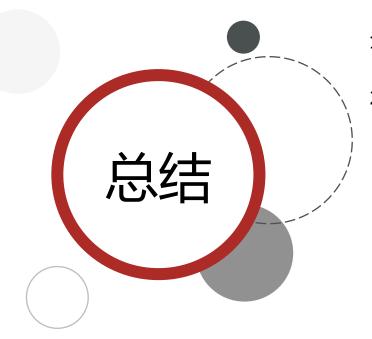
● 周三:今晚啤酒、龙虾、小烧烤 周日:郁郁寡欢、准备上班。

● 周四: 主动帮助新来的女程序解决bug

周五:今晚吃鸡

周六:与王婆介绍的小芳相亲





- 1. switch分支的格式、执行流程是怎么样的?
- 2. if、switch分支各自适合做什么业务场景?
 - if其实在功能上远远强大于switch。
 - if适合做区间匹配。
 - switch适合做:值匹配的分支选择、代码优雅。

```
switch(表达式){
   case 值1:
       执行代码1;
       break;
   case 值2:
       执行代码2;
       break;
   case 值n-1:
       执行代码n-1;
       break;
   default:
       执行代码n;
       // break;
```



> 分支结构

- ◆ If分支
- ◆ switch分支
- ◆ switch的注意事项
- ◆ switch穿透性
- > 循环结构
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类



switch分支注意事项:

- ① 表达式类型只能是byte、short、int、char , JDK5开始支持枚举 , JDK7开始支持String、不支持double、float、long。
- ② case给出的值不允许重复,且只能是字面量,不能是变量。
- ③ 不要忘记写break,否则会出现穿透现象。

```
switch(表达式){
   case 值1:
      执行代码...;
       break:
   case 值2:
      执行代码...;
       break;
   case 值n-1:
      执行代码...;
      break;
   default:
      执行代码n;
```



> 分支结构

- ◆ If分支
- ◆ switch分支
- ◆ switch的注意事项
- ◆ switch穿透性的应用场景
- > 循环结构
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类



switch的穿透性

● 如果代码执行到没有写break的case块,执行完后将直接进入下一个case块执行代码(而且不会进行任何匹配), 直到遇到break才跳出分支,这就是switch的穿透性。

switch穿透性案例(月份天数查看器)

需求:用户输入月份可以展示该月份的天数。

- 1、3、5、7、8、10、12月份是31天
- 2月份是闰年为29天、非闰年为28天。
- 4、6、9、11月份 是30天

```
switch(表达式){
   case 值1:
      执行代码1;
   case 值2:
      执行代码2;
   case 值n-1:
      执行代码n-1;
      break;
   default:
      执行代码n;
      // break;
```





- 1. 什么情况下会出现switch穿透现象?
 - case中没有写break。
- 2. switch穿透性能解决什么问题?
 - 存在多个case分支的功能代码是一样时,可以用穿透性把流程集中到同一处处理,这样可以简化代码。



- > 分支结构
- > 循环结构
 - ◆ for循环
 - ◆ for循环案例1
 - ◆ for循环案例2
 - ◆ for循环案例3
 - ◆ while循环
 - ◆ while循环案例
 - ◆ do-while循环
 - ◆ 死循环
 - ◆ 循环嵌套
- > 跳转关键字: break、continue
- 》 案例技术:随机数Random类



for 循环

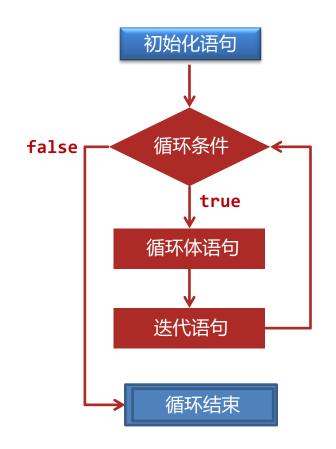
● 控制一段代码反复执行很多次。

```
格式:
for (初始化语句;循环条件;迭代语句) {
 循环体语句(重复执行的代码);
}
```

示例

```
//输出3次HelloWorld

for (int i = 0; i < 3; i++) {
    System.out.println("Hello World");
}
```





for 循环案例详细流程说明

```
//输出3次HelloWorld

for (int i = 0; i < 3; i++) {
    System.out.println("Hello World");
}
```

执行的流程:

- 循环一开始,执行int i = 0 一次。
- ② 然后判断循环条件:0 < 3 返回true ,进入到循环体中执行输出 :helloWorld ,然后执行迭代语句i++ ,此时i=1了。
- ③ 然后判断循环条件:1 < 3返回true,进入到循环体中执行输出 :helloWorld ,然后执行迭代语句i++,此时i=2了。
- ④ 然后判断循环条件:2 < 3 返回true,进入到循环体中执行输出:helloWorld,然后执行迭代语句i++,此时i=3了。
- ⑤ 然后判断循环条件:3 < 3 返回false, 循环立即结束!!





1. for循环格式和执行流程是什么样的?

```
for (初始化语句 ;循环条件;迭代语句) {循环体语句;
}
```

```
//输出3次HelloWorld

for (int i = 0; i < 3; i++) {
    System.out.println("Hello World");
}
```



- > 分支结构
- > 循环结构
 - ◆ for循环
 - ◆ for循环案例1
 - ◆ for循环案例2
 - ◆ for循环案例3
 - ◆ while循环
 - ◆ while循环案例
 - ◆ do-while循环
 - ◆ 死循环
 - ◆ 循环嵌套
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类





求和

需求: 求1-5之间的数据和, 并把求和结果在控制台输出。

分析:

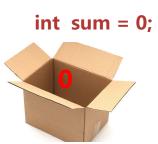
① 定义for循环,使其能够依次产生:1、2、3、4、5。

```
for (int i = 1; i <= 5; i++) {
}
```

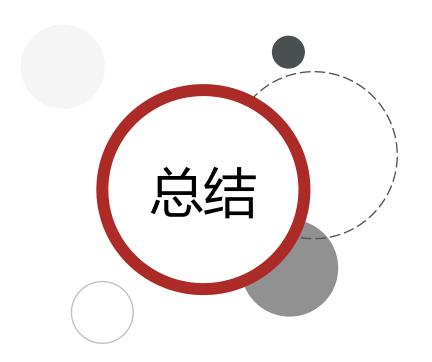
② 在循环外定义一个整数变量sum用于求和,循环每产生一个数,就累加到sum中去

```
int sum = 0;
for (int i = 1; i <= 5; i++) {
    sum += i;
}
```

③ 循环结束后,输出求和变量即是结果。







- 1. 如何实现元素求1-5的和?
 - 使用循环控制输出1-5

```
for (int i = 1; i <= 5; i++) {
}</pre>
```

● 在循环外定义变量sum累加数据。



- > 分支结构
- > 循环结构
 - ◆ for循环
 - ◆ for循环案例1
 - ◆ for循环案例2
 - ◆ for循环案例3
 - ◆ while循环
 - ◆ while循环案例
 - ◆ do-while循环
 - ◆ 死循环
 - ◆ 循环嵌套
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类





求奇数和

需求: 求1-10之间的奇数和, 并把求和结果在控制台输出。

方式一:

- ① 定义for循环,使其能够依次产生:1、2、3、4、5...10。
- ② 循环每产生一个数据,都通过if判断其是否是奇数

```
for (int i = 1; i <= 10; i++) {
    if (i % 2 == 1) {
        // i = 1 3 5 7...
    }
}</pre>
```

③ 在循环外定义一个整数变量sum,在if分支内来累加产生的奇数数据。







求奇数和

需求: 求1-10之间的奇数和, 并把求和结果在控制台输出。

方式二:

① 定义for循环,使其能够依次产生:1、3、5、7、9。

```
for (int i = 1; i <= 10; i+=2) {
}</pre>
```

② 在循环外定义一个整数变量sum,循环每产生一个奇数就累加到sum







1. 如何实现求奇数和

● 方式一:在for循环中,通过if筛选出奇数

```
for (int i = 1; i <= 10; i++) {
    if (i % 2 == 1) {
        // i = 1 3 5 7...
    }
}</pre>
```

● 方式二:直接使用for循环找出奇数。

```
for (int i = 1; i <= 10; i+=2) {
}</pre>
```



- > 分支结构
- > 循环结构
 - ◆ for循环
 - ◆ for循环案例1
 - ◆ for循环案例2
 - ◆ for循环案例3
 - ◆ while循环
 - ◆ while循环案例
 - ◆ do-while循环
 - ◆ 死循环
 - ◆ 循环嵌套
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类



国 案例

水仙花数

需求:在控制台输出所有的"水仙花数",水仙花数必须满足如下2个要求:

- 1. 水仙花数是一个三位数
- 2. 水仙花数的个位、十位、百位的数字立方和等于原数

分析:

- ① 定义一个for循环从"100一直到999"。
- ② 每次访问到数据后,提取该数据的:个位、十位、百位数字。
- ③ 使用if判断:个位、十位、百位的数字立方和是否等于原数,等于则输出该数据。

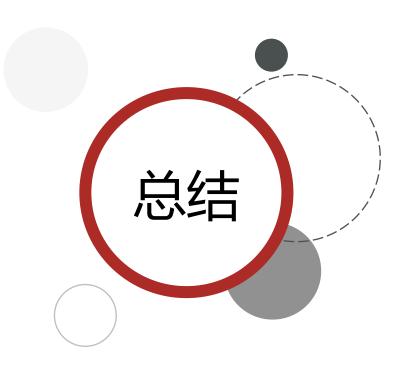
153 370 371 407





- 1. 如果还要知道水仙花数的个数怎么办?
 - 在循环外定义一个变量count用于记录水仙花数。
 - 每输出水仙花数时,让count++。





1. 如何找出水仙花数?

- 定义一个for循环从 "100一直到999"。
- 每次访问到数据后,提取该数据的:个位、十位、百位数字。
- 看各个数的立方和是否等于原数,等于则输出原数据。
- 2. 如何计算出水仙花的个数?
 - 在循环外定义一个变量count用于记录水仙花数。
 - 每输出水仙花数时,让count++。



- > 分支结构
- > 循环结构
 - ◆ for循环
 - ◆ for循环案例1
 - ◆ for循环案例2
 - ◆ for循环案例3
 - ◆ while循环
 - ◆ while循环案例
 - ◆ do-while循环
 - ◆ 死循环
 - ◆ 循环嵌套
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类

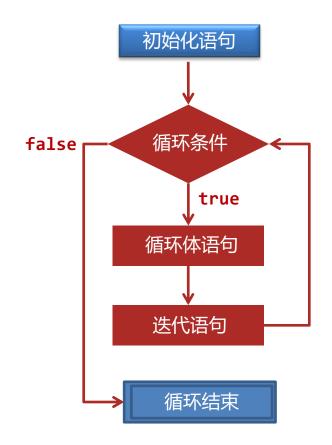


while 循环格式与执行流程

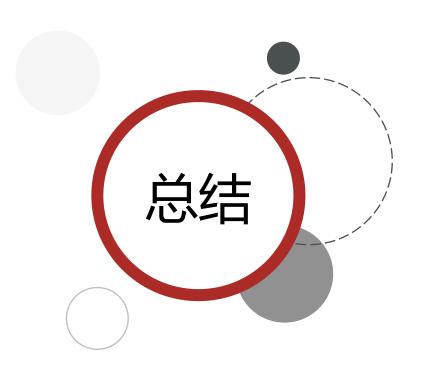
```
初始化语句;
while (循环条件) {
    循环体语句(<mark>被重复执行的代码</mark>);
    迭代语句;
}
```

示例

```
int i = 0;
while (i < 3) {
    System.out.println("Hello World");
    i++;
}</pre>
```







1. while循环的格式,执行流程是怎么样的?

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

```
int i = 0;
while (i < 3) {
    System.out.println("Hello World");
    i++;
}</pre>
```

- 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。



- > 分支结构
- > 循环结构
 - ◆ for循环
 - ◆ for循环案例1
 - ◆ for循环案例2
 - ◆ for循环案例3
 - ◆ while循环
 - ◆ while循环案例
 - ◆ do-while循环
 - ◆ 死循环
 - ◆ 循环嵌套
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类





珠穆朗玛峰(世界最高峰8848.86米)



需求:

世界最高山峰是珠穆朗玛峰(8848.86米=8848860毫米), 假如我有一张足够大的纸, 它的厚度是0.1毫米。请问, 折叠多少次, 可以折成珠穆朗玛峰的高度。

思路:

● 这种不清楚要循环多少次的情况可以选择用while实现。





珠穆朗玛峰(世界最高峰8848.86米)



分析步骤

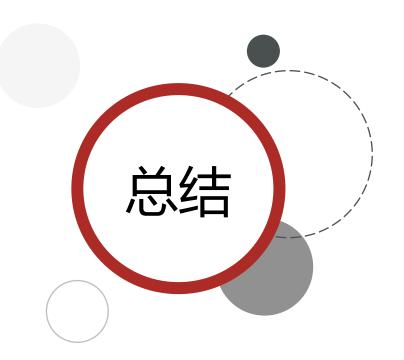
① 定义变量存储珠穆朗玛峰的高度、纸张的高度。

```
double peakHeight = 8848860; // 山峰高度
double paperThickness = 0.1; // 纸张厚度
```

② 使用while循环,循环条件是(纸张厚度<山峰高度),内部控制纸张折叠,每折叠一次,纸张厚度为原来两倍,循环外定义计数变量,每折叠依次让该变量+1

```
int count = 0;
while (paperThickness < peakHeight){
   paperThickness *= 2;
   count++;
}</pre>
```





1. 怎么解决此案例?

- ① 定义变量存储珠穆朗玛峰的高度、纸张的高度。
- ② 使用while循环,循环条件是(纸张厚度<山峰高度),内部控制纸张折叠,每折叠 一次,纸张厚度为原来两倍,循环外定义计数变量,每折叠依次让该变量+1
- 2. for和while使用总结
 - ① 其实while能做的for都能实现
 - ② 但是如果一开始不知道循环次数的情况下,建议使用while循环解决更专业。



- > 分支结构
- > 循环结构
 - ◆ for循环
 - ◆ for循环案例1
 - ◆ for循环案例2
 - ◆ for循环案例3
 - ◆ while循环
 - ◆ while循环案例
 - ◆ do-while循环、三种循环小结
 - ◆ 死循环
 - ◆ 循环嵌套
- 跳转关键字: break、continue
- > 案例技术:随机数Random类

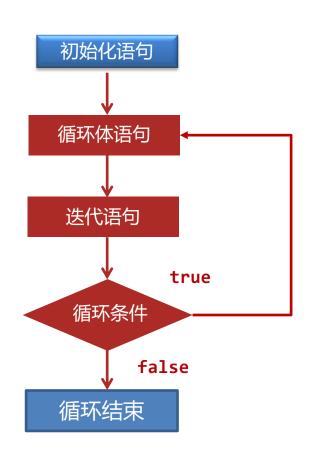


do-while循环

● 先执行再判断循环条件。

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





do-while循环的特点:一定会先执行一次循环体。



三种循环的区别小结

- for循环 和 while循环(先判断后执行)
- do...while (第一次先执行后判断)
- for循环和while循环的执行流程是一模一样的。
- 如果已知循环次数建议使用for循环,如果不清楚要循环多少次建议使用while循环。
- for循环中,控制循环的变量只在循环中可以使用。While循环中,控制循环的变量在循环后还可以继续使用。

```
int i = 0;
while (i < 3) {
    System.out.println("Hello World");
    i++;
}
System.out.println(i);</pre>
```

```
for (int i = 0; i < 3; i++ ) {
    System.out.println("Hello World");
}
System.out.println(i);</pre>
```



- > 分支结构
- > 循环结构
 - ◆ for循环
 - ◆ for循环案例1
 - ◆ for循环案例2
 - ◆ for循环案例3
 - ◆ while循环
 - ◆ while循环案例
 - ◆ do-while循环
 - ◆ 死循环
 - ◆ 循环嵌套
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类



死循环

● 一直循环的执行下去,如果没有干预不会停止下来。

写法

```
for(;;) {
    System.out.println("Hello World");
// 经典做法
while(true) {
    System.out.println("Hello World");
do {
    System.out.println("Hello World");
} while (true);
```





密码验证

需求:系统密码是520,请用户不断的输入密码验证,验证不对输出:密码错误,验证成功输出:欢迎进入系统,并停止程序。

分析:

- ① 定义一个整型变量记录正确的密码:520
- ② 使用while死循环,让用户不断输入数据,与正确密码比对:验证不成功输出:密码错误、验证成功输出:欢迎进入系统,并使用break结束当前循环的执行。





1. 死循环可以怎么写?

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



- > 分支结构
- > 循环结构
 - ◆ for循环
 - ◆ for循环案例1
 - ◆ for循环案例2
 - ◆ for循环案例3
 - ◆ while循环
 - ◆ while循环案例
 - ◆ do-while循环
 - ◆ 死循环
 - ◆ 循环嵌套
- 跳转关键字: break、continue
- > 案例技术:随机数Random类



循环嵌套

● 循环中又包含循环

```
for(int i = 0; i < 3; i++) {
    for(int j = 0; j < 5; j++) {
        System.out.println("我爱你");
    }
}
```

嵌套循环的特点

● 外部循环每循环一次,内部循环全部执行完一次。





循环嵌套

需求:在控制台使用 * 打印出4行5列的矩形

```
*****
*****
****
```

```
for(int i = 1; i <= 4; i++) {
    System.out.println("*****");
}</pre>
```





- > 分支结构
- > 循环结构
- 跳转关键字: break、continue
- 》 案例技术:随机数Random类



跳转控制语句介绍

● break : 跳出并结束当前所在循环的执行。

● continue: 用于跳出当前循环的当次执行,进入下一次循环。

注意事项

break: 只能用于结束所在循环,或者结束所在switch分支的执行。

continue : 只能在循环中进行使用。



- > 分支结构
- > 循环结构
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类
 - ◆ Random的使用
 - ◆ 猜数字游戏



Random随机数技术

● 作用:用于在程序中获取随机数的技术。

使用步骤:

①:导包:告诉程序去JDK的哪个包中找随机数技术

②:写一行代码代表得到随机数对象

③:调用随机数的功能获取0-9的随机数

注意:

● nextInt(n) 功能只能生成: 0 至 n-1之间的随机数,不包含 n。

```
package com.itheima.random;
import java.util.Random;
public class Test {
    public static void main(String[] args) {
        Random r = new Random();
        int number = r.nextInt(10);
        System.out.println("随机生成了: " + number);
    }
}
```



Random生成随机数的特点

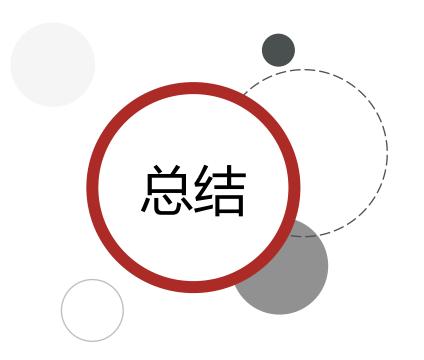
● nextInt(n)功能只能生成:0-(n-1)之间的随机数。

Random生成区间随机数的技巧:减加法。

● 例如:要生成1-10之间的随机数,程序怎么实现?

$$\boxed{1-10} \longrightarrow -1 \longrightarrow \boxed{(0-9)+1}$$





- 1. Random随机数类生成需要几步,具体是什么样的?
 - 导包: import java.util.Random;
 - Random r = new Random();
 - int number = r.nextInt(10);
- 2. Random随机数如何生成 65 91之间的随机数?
 - \bullet 65 91 => (0 26) + 65
 - int number = r.nextInt(27) + 65;



- > 分支结构
- > 循环结构
- > 跳转关键字: break、continue
- > 案例技术:随机数Random类
 - ◆ Random的使用
 - ◆ 猜数字游戏



軍 案例

猜数字游戏

需求:

● 随机生成一个1-100之间的数据,提示用户猜测,猜大提示过大,猜小提示过小,直到猜中结束游戏

0

分析:

- ① 随机生成一个1-100之间的数据
- ② 使用死循环让用户不断提示用户猜测,猜大提示过大,猜小提示过小,猜中结束游戏。



传智教育旗下高端IT教育品牌