语言基础第五天：

回顾：

1. Scanner接收用户输入的数据：----------共3步
2. 分支结构：
   - if...else if：多条路
   - switch...case：多条路
     - 优点：效率高、结构清晰
     - 缺点：只能对整数判断相等
     - break：跳出switch
3. 循环：反复多次执行一段相同或相似的代码
4. 循环三要素：
   - 循环变量的初始化
   - 循环的条件(以循环变量为基础)
   - 循环变量的改变(向着循环的结束变)

   > 循环变量：在整个循环过程中所反复改变的那个数
5. 循环结构：
   - while结构：先判断后执行，有可能一次都不走
   - do...while结构：先执行后判断，至少走一次

   > 第1要素与第3要素的代码相同时，使用do...while

精华笔记：

1. 循环结构：
   - for结构：应用率高、与次数相关的循环

```
1)语法：
  //    1    2    3
  for(要素1;要素2;要素3){
      语句块/循环体----------------反复执行的语句   4
  }
2)执行过程：
  1243243243243243...2
```

2. 三种循环结构如何选择：
   - 先看循环是否与次数相关：
     - 若相关----------直接上for
     - 若无关，再看第1要素与第3相互的代码是否相同：
       - 若相同----------直接上do...while
       - 若不同----------直接上while

3. break：跳出循环

   continue：跳过循环体中剩余语句而进入下一次循环

4. 嵌套循环：
   - 循环中套循环，常常多行多列时使用，一般外层控制行，内层控制列
   - 执行规则：外层循环走一次，内层循环走所有次
   - 建议：嵌套层数越少越好，能用一层就不用两层，能用两层就不用三层
   - break只能跳出当前一层循环
5. 数组：
   - 是一种数据类型(引用类型)
   - 相同数据类型元素的集合
   - 定义：
   - 初始化：----------初始化的是数组中的元素
   - 访问：----------访问的是数组中的元素
     - 通过(数组名.length)可以获取数组的长度(元素个数)
     - 通过下标/索引来访问数组中的元素，下标从0开始，最大到(数组的长度-1)
   - 遍历/迭代：从头到尾挨个走一遍

笔记：

1. 循环结构：
   - for结构：应用率高、与次数相关的循环

```
1)语法：
  //    1    2    3
  for(要素1;要素2;要素3){
      语句块/循环体----------反复执行的语句   4
  }
2)执行过程：
  1243243243243243...2
```

```
for(int times=0;times<5;times++){
    System.out.println("行动是成功的阶梯");
}

//特殊的：for中的循环变量num的作用域---仅在当前for中
for(int num=1;num<=9;num++){
    System.out.println(num+"*9="+num*9);
}

for(int num=1;num<=9;num+=2){
    System.out.println(num+"*9="+num*9);
}
/*
    执行过程：
       num=1  true  1*9=9
       num=3  true  3*9=27
       num=5  true  5*9=45
       num=7  true  7*9=63
       num=9  true  9*9=81
```

```
        num=11 false for循环结束
*/

//演示for的特殊语法格式:
int num=1;
for(;num<=9;num++){
    System.out.println(num+"*9="+num*9);
}

for(int num=1;num<=9;){
    System.out.println(num+"*9="+num*9);
    num++;
}

for(;;){ //没有条件的循环就是一个死循环
    System.out.println("我爱Java");
}

for(int i=1,j=5;i<=5;i+=2,j-=2){
}
/*
   i=1,j=5
   i=3,j=3
   i=5,j=1
   i=7,j=-1
*/
```

```
//随机加法运算器
package day05;
import java.util.Scanner;
//随机加法运算器
public class Addition {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int score = 0; //总分
        for(int i=1;i<=10;i++){ //10次     (1)25+65=?
            int a = (int)(Math.random()*100); //加数a--0到99的随机数
            int b = (int)(Math.random()*100); //加数b--0到99的随机数
            int result = a+b; //存正确答案
            System.out.println("("+i+")"+a+"+"+b+"=?"); //1)出题
            System.out.println("算吧!");
            int answer = scan.nextInt(); //2)答题
            if(answer==-1){ //3)判题
                break;
            }
            if(answer==result){
                System.out.println("答对了");
                score += 10; //答对1题，加10分
            }else{
                System.out.println("答错了");
            }
        }
        System.out.println("总分为:"+score);
    }
}
```

2. 三种循环结构如何选择：
    - 先看循环是否与次数相关：
        - 若相关----------直接上for
        - 若无关，再看第1要素与第3相互的代码是否相同：
            - 若相同----------直接上do...while
            - 若不同----------直接上while
3. break：跳出循环

```
for(int num=1;num<=9;num++){
    if(num==4){ //在某种特定条件下，提前结束循环
        break; //跳出循环
    }
    System.out.println(num+"*9="+num*9);
}

/*
   执行过程：
      num=1  1*9=9
      num=2  2*9=18
      num=3  3*9=27
      num=4
*/
```

continue：跳过循环体中剩余语句而进入下一次循环

```
//输出9的乘法表，只要不能被3整除
for(int num=1;num<=9;num++){
    if(num%3!=0){
        System.out.println(num+"*9="+num*9);
    }
}

//输出9的乘法表，跳过能被3整除的
for(int num=1;num<=9;num++){
    if(num%3==0){
        continue; //跳过循环体中剩余语句而进入下一次循环
    }
    System.out.println(num+"*9="+num*9);
}
/*
  num=1  1*9=9
  num=2  2*9=18
  num=3
  num=4  4*9=36
  num=5  5*9=45
  num=6
  num=7  7*9=63
  num=8  8*9=72
  num=9
  num=10 false
*/
```

4. 嵌套循环：
    - 循环中套循环，常常多行多列时使用，一般外层控制行，内层控制列

- 执行规则：外层循环走一次，内层循环走所有次
- 建议：嵌套层数越少越好，能用一层就不用两层，能用两层就不用三层
- break只能跳出当前一层循环

```
//九九乘法表
public class MultiTable {
    public static void main(String[] args) {
        for(int num=1;num<=9;num++){ //控制行
            for(int i=1;i<=num;i++){ //控制列
                System.out.print(i+"*"+num+"="+i*num+"\t");
            }
            System.out.println(); //换行
        }
        /*
          执行过程:
            num=3
              i=1  1*3=3
              i=2  2*3=6
              i=3  3*3=9
              i=4  false
              换行
            num=2
              i=1  1*2=2
              i=2  2*2=4
              i=3  false
              换行
            num=1
              i=1  1*1=1
              i=2  false
              换行
         */
    }
}
```

5. 数组：
   - 是一种数据类型(引用类型)
   - 相同数据类型元素的集合
   - 定义：

```
//声明整型数组arr，包含10个元素，每个元素都是int型，默认值为0
int[] arr = new int[10];
```

   - 初始化：----------初始化的是数组中的元素

```
int[] arr1 = new int[3]; //0,0,0
int[] arr2 = {2,5,8}; //2,5,8
int[] arr3 = new int[]{2,5,8}; //2,5,8
int[] arr4;
//arr4 = {2,5,8}; //编译错误，此方式只能声明同时初始化
arr4 = new int[]{2,5,8}; //正确
```

   - 访问：----------访问的是数组中的元素
     - 通过(数组名.length)可以获取数组的长度(元素个数)

```
int[] arr = new int[3];
System.out.println(arr.length); //3，输出数组的长度
```

- 通过下标/索引来访问数组中的元素，下标从0开始，最大到(数组的长度-1)

```
int[] arr = new int[3];
System.out.println(arr[0]); //0，输出第1个元素的值
arr[0] = 100; //给第1个元素赋值为100
arr[1] = 200; //给第2个元素赋值为200
arr[2] = 300; //给第3个元素赋值为300
arr[3] = 400; //运行时会发生数组下标越界异常
System.out.println(arr[arr.length-1]); //输出最后一个元素的值
```

- 遍历/迭代：从头到尾挨个走一遍

```
int[] arr = new int[10];
for(int i=0;i<arr.length;i++){ //遍历arr数组
    arr[i] = (int)(Math.random()*100); //给每个元素赋值为0到99的之间的随机数
    System.out.println(arr[i]); //输出每个元素的值
}
```

补充：

1. 变量的重名问题：
   - 作用域重叠时，变量不能同名
2. \t：水平制表位，固定占8位
3. 数组元素的默认值：

```
byte,short,int,long,char----------------0
float,double----------------------------0.0
boolean---------------------------------false
```

4. 异常：
   - ArrayIndexOutOfBoundsException：数组下标越界异常
     - 数组下标为0到(数组长度-1)，若超出这个范围则发生数组下标越界异常
5. 明日单词：

```
1)copy:复制
2)arraycopy/copyOf:数组复制
3)max:最大值
4)min:最小值
5)sort:顺序、排序
6)method:方法
7)public static:公开静态的
8)void:空，没有返回结果的
9)return:返回
10)say:说
11)sayHi/sayHello:问好
12)getNum:获取数字
```

```
13)plus:加法
14)test:测试
```