语言基础第五天：

回顾：

1. Scanner接收用户输入的数据：3步，先背下来
2. 分支结构(下)：
    - switch...case结构：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
    - 循环条件(以循环变量为基础)
    - 循环变量的改变
5. 循环结构：
    - while结构：先判断后执行，有可能一次都不执行
    - do...while结构：先执行后判断，至少执行一次

        > 若要素1与要素3的代码相同，首选do...while

    - for结构：应用率最高，与次数相关的

精华笔记：

1. for的练习：
2. break：跳出循环

   continue：跳过循环体中剩余语句而进入下一次循环
3. 嵌套循环：
    - 循环中套循环，常常多行多列时使用，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默认只能跳出当前一层循环
4. 数组(上)：
    - 是一种引用数据类型
    - 相同数据类型元素的集合
    - 定义：
    - 初始化：初始化数组中的元素
    - 访问：
        - 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

- 排序：

笔记：

1. for练习：

```
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+14=?
            int a = (int)(Math.random()*100); //加数a(0到99)
            int b = (int)(Math.random()*100); //加数b(0到99)
            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==result){ //3)判题
                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. 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);
}
/*
   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，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
*/

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

3. 嵌套循环：
   - 循环中套循环，常常多行多列时使用，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默认只能跳出当前一层循环

```
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
            */
        }
    }
```

4. 数组(上):
    - 是一种引用数据类型
    - 相同数据类型元素的集合
    - 定义:

```
//声明整型数组a，包含5个元素，每个元素都是int类型，默认值为0
int[] a = new int[5];
//声明浮点型数组d，包含10个元素，每个元素都是double类型，默认值为0.0
double[] d = new double[10];
//声明布尔型数组b，包含26个元素，每个元素都是boolean类型，默认值为false
boolean[] b = new boolean[26];
```

    - 初始化：初始化数组中的元素

```
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:
        - 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("数组的长度:"+arr.length); //3
```

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

```
int[] arr = new int[3];
System.out.println("数组的长度:"+arr.length); //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]); //300，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的随机数
    System.out.println(arr[i]); //输出每个元素的值
}
```

```
public class MaxOfArray {
    public static void main(String[] args) {
        int[] arr = new int[10];
        for(int i=0;i<arr.length;i++){
            arr[i] = (int)(Math.random()*100);
            System.out.println(arr[i]);
        }

        //                  0, 1, 2, 3
        //假设:int[] arr = {12,56,89,8};
        //max=12/56/89
        int max = arr[0]; //假设第1个元素为最大值
        for(int i=1;i<arr.length;i++){ //遍历剩余元素
            if(arr[i]>max){   //若剩余元素大于max
                max = arr[i]; //将max修改为较大的
            }
        }
        System.out.println("最大值为:"+max);

    }
}
```

- 排序:

```
package day05;
import java.util.Arrays;
//数组的演示
public class ArrayDemo {
    public static void main(String[] args) {
        //5)数组的排序:
        Random rand = new Random(); //随机数对象
        int[] arr = new int[10];
        for(int i=0;i<arr.length;i++){
            arr[i] = rand.nextInt(100); //0到99的随机整数
            System.out.println(arr[i]);
        }

        Arrays.sort(arr); //对arr数组做升序排列

        System.out.println("排序后:");
        for(int i=0;i<arr.length;i++){
            System.out.println(arr[i]);
        }

        System.out.println("倒着输出:");
        for(int i=arr.length-1;i>=0;i--){ //数据还是升序的，只是倒着展示
            System.out.println(arr[i]);
        }
        System.out.println("第1个元素为:"+arr[0]);
    }
}
```

补充：

1. \t：水平制表位，固定占8位
2. 默认值：

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

3. ArrayIndexOutOfBoundsException：数组下标越界异常

> 数组下标一定在0到(数组长度-1)之间，若超出范围，在运行时会发生数组下标越界异常

4. 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
```