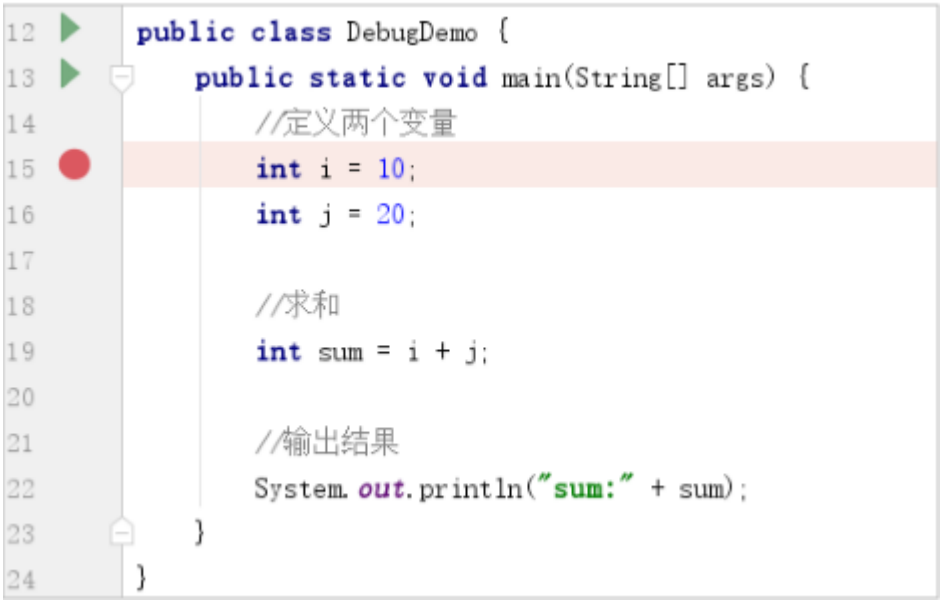
1.Debug模式

1.1什么是Debug模式【理解】

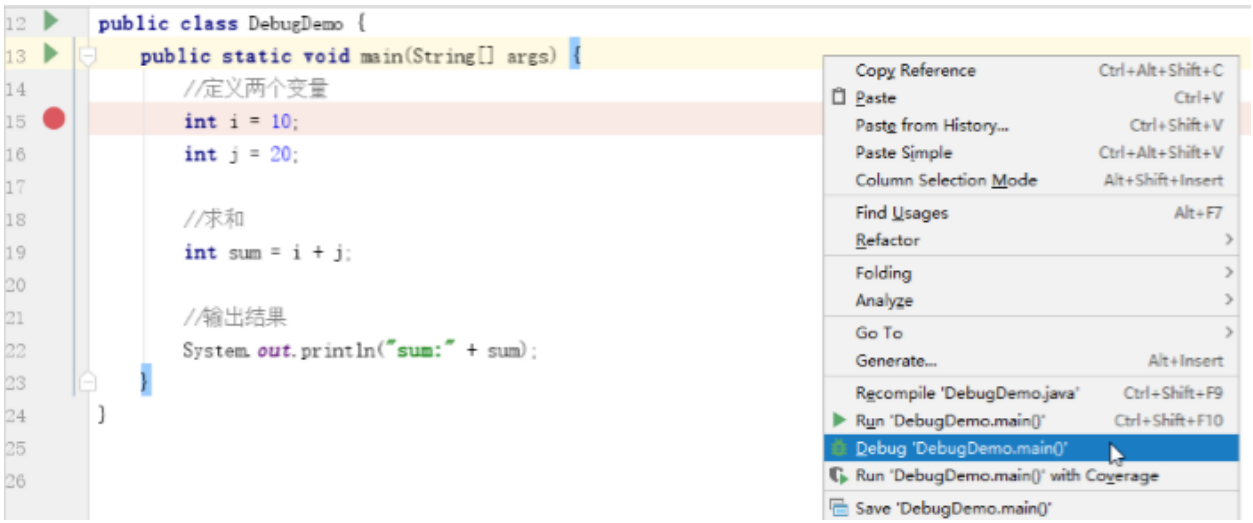
是供程序员使用的程序调试工具，它可以用于查看程序的执行流程，也可以用于追踪程序执行过程来调试程序。

1.2Debug模式操作流程【应用】

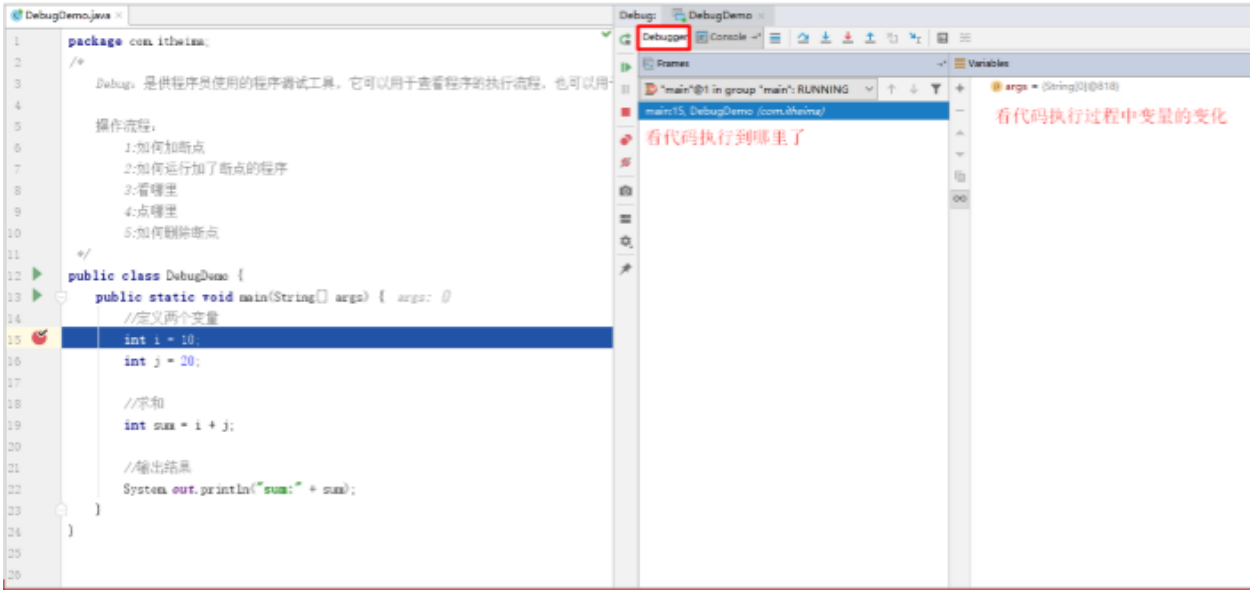
- 如何加断点
  - 选择要设置断点的代码行，在行号的区域后面单击鼠标左键即可



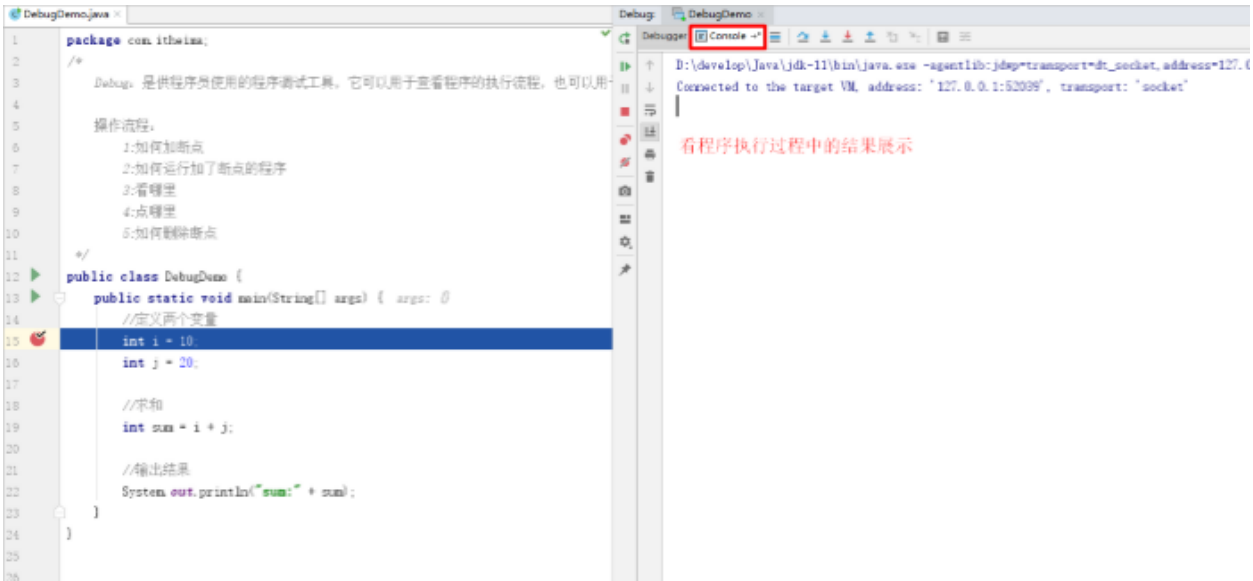
- 如何运行加了断点的程序
  - 在代码区域右键Debug执行



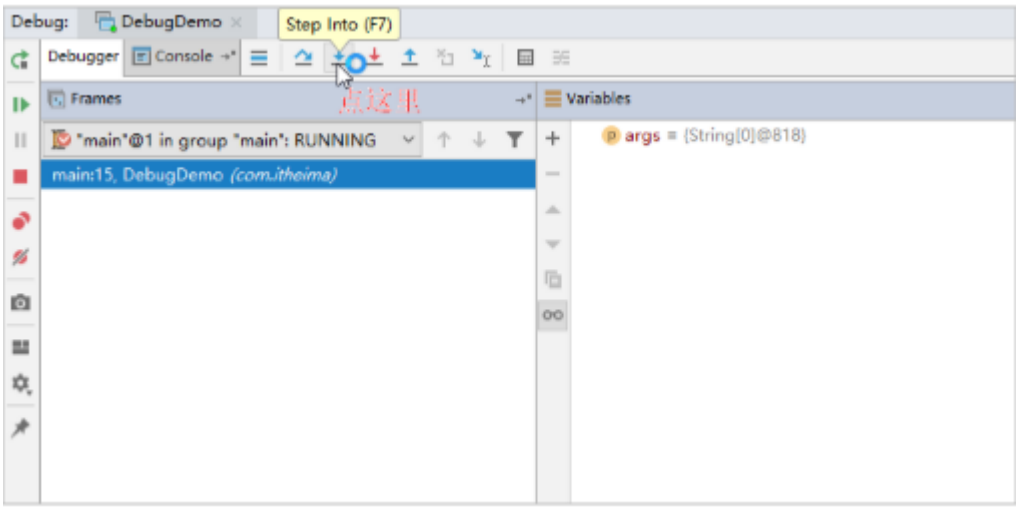
- 看哪里
  - 看Debugger窗口



- 看Console窗口

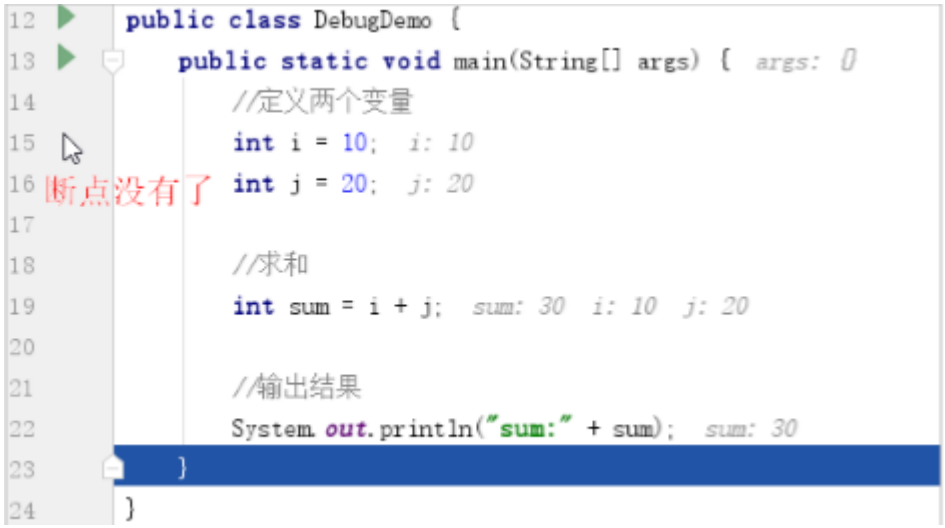


- 点哪里
  - 点Step Into (F7)这个箭头，也可以直接按F7

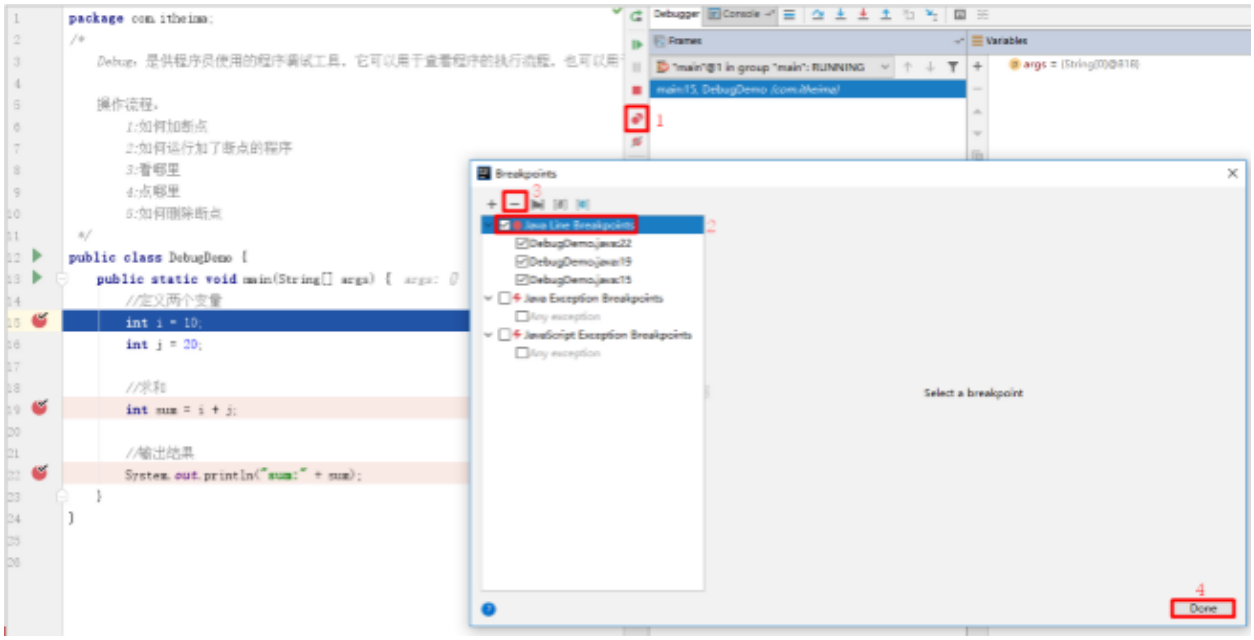


- 如何删除断点

- 选择要删除的断点，单击鼠标左键即可



- 如果是多个断点，可以每一个再点击一次。也可以一次性全部删除



2.基础练习

2.1减肥计划if版本【应用】

2.1.1案例需求

输入星期数，显示今天的减肥活动 周一：跑步 周二：游泳 周三：慢走 周四：动感单车 周五：拳击 周六：爬山 周日：好好吃一顿

2.1.2代码实现

```
/*
    思路：
        1:键盘录入一个星期数，用一个变量接收
        2:对星期数进行判断，这里用 if 语句实现
        3:在对应的语句控制中输出对应的减肥活动
```

```
 */
public class Test01 {
    public static void main(String[] args) {
        //键盘录入一个星期数，用一个变量接收
        Scanner sc = new Scanner(System.in);

        System.out.println("请输入一个星期数：");
        int week = sc.nextInt();

        //对星期数进行判断，这里用 if 语句实现
        if (week < 1 || week > 7) {
            System.out.println("你输入的星期数有误");
        } else if (week == 1) {
            System.out.println("跑步");
        } else if (week == 2) {
            System.out.println("游泳");
        } else if (week == 3) {
            System.out.println("慢走");
        } else if (week == 4) {
            System.out.println("动感单车");
        } else if (week == 5) {
            System.out.println("拳击");
        } else if (week == 6) {
            System.out.println("爬山");
        } else {
            System.out.println("好好吃一顿");
        }
    }
}
```

2.2减肥计划switch版本【应用】

2.2.1案例需求

输入星期数，显示今天的减肥活动 周一：跑步 周二：游泳 周三：慢走 周四：动感单车 周五：拳击 周六：爬山 周日：好好吃一顿

2.2.2代码实现

```
/*
    思路：
        1:键盘录入一个星期数，用一个变量接收
        2:对星期数进行判断，这里用 switch 语句实现
        3:在对应的语句控制中输出对应的减肥活动

    导包：
        1:手动导包  import java.util.Scanner;
        2:快捷键导包 Alt+Enter
        3:自动导包
 */
public class Test02 {
    public static void main(String[] args) {
        //键盘录入一个星期数，用一个变量接收
```

```
        Scanner sc = new Scanner(System.in);

        System.out.println("请输入一个星期数：");
        int week = sc.nextInt();

        //对星期数进行判断，这里用 switch 语句实现
        switch (week) {
            case 1:
                System.out.println("跑步");
                break;
            case 2:
                System.out.println("游泳");
                break;
            case 3:
                System.out.println("慢走");
                break;
            case 4:
                System.out.println("动感单车");
                break;
            case 5:
                System.out.println("拳击");
                break;
            case 6:
                System.out.println("爬山");
                break;
            case 7:
                System.out.println("好好吃一顿");
                break;
            default:
                System.out.println("你输入的星期数有误");
        }
    }
}
```

2.3逢七跳过【应用】

2.3.1案例需求

朋友聚会的时候可能会玩一个游戏：逢七过。 规则是：从任意一个数字开始报数，当你要报的数字包含7或者是7的倍数时都要说：过。 为了帮助大家更好的玩这个游戏，这里我们直接在控制台打印出1-100之间的满足逢七必过规则的数据。 这样，大家将来在玩游戏的时候，就知道哪些数据要说：过。

2.3.2代码实现

```
/*
    思路：
        1:数据在1-100之间，用for循环实现数据的获取
        2:根据规则，用if语句实现数据的判断：要么个位是7，要么十位是7，要么能够被7整除
        3:在控制台输出满足规则的数据
 */
public class Test03 {
    public static void main(String[] args) {
```

```
        //数据在1-100之间，用for循环实现数据的获取
        for(int x=1; x<=100; x++) {
            //根据规则，用if语句实现数据的判断：要么个位是7，要么十位是7，要么能够被7整除
            if(x%10==7 || x/10%10==7 || x%7==0) {
                //在控制台输出满足规则的数据
                System.out.println(x);
            }
        }
    }
}
```

2.4不死神兔【应用】

2.4.1案例需求

有一对兔子，从出生后第3个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子， 假如兔子都不死，问第二十个月的兔子对数为多少？

2.4.2代码实现

```
/*
   思路：
        1:为了存储多个月的兔子对数，定义一个数组，用动态初始化完成数组元素的初始化，长度为20
        2:因为第1个月，第2个月兔子的对数是已知的，都是1，所以数组的第1个元素，第2个元素值也都是1
        3:用循环实现计算每个月的兔子对数
        4:输出数组中最后一个元素的值，就是第20个月的兔子对数
 */
public class Test04 {
    public static void main(String[] args) {
        //为了存储多个月的兔子对数，定义一个数组，用动态初始化完成数组元素的初始化，长度为20
        int[] arr = new int[20];

        //因为第1个月，第2个月兔子的对数是已知的，都是1，所以数组的第1个元素，第2个元素值也都是1
        arr[0] = 1;
        arr[1] = 1;

        //用循环实现计算每个月的兔子对数
        for(int x=2; x<arr.length; x++) {
            arr[x] = arr[x-2] + arr[x-1];
        }

        //输出数组中最后一个元素的值，就是第20个月的兔子对数
        System.out.println("第二十个月兔子的对数是：" + arr[19]);
    }
}
```

2.5百钱白鸡【应用】

2.5.1案例需求

我国古代数学家张丘建在《算经》一书中提出的数学问题：鸡翁一值钱五，鸡母一值钱三，鸡雏三值钱一。 百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？

2.5.2代码实现

```
/*
    思路：
        1:第1层循环，用于表示鸡翁的范围，初始化表达式的变量定义为 x=0，判断条件是x<=20
        2:第2层循环，用于表示鸡母的范围，初始化表达式的变量定义为 y=0，判断条件是y<=33
        3:这个时候，用于表示鸡雏的变量 z = 100 - x - y
        4:判断表达式 z%3==0 和表达式 5*x + 3*y + z/3 = 100 是否同时成立，如果成立，输出对应的x，y，z 的值，就是对应的鸡翁，鸡母，鸡雏的值
 */
public class Test05 {
    public static void main(String[] args) {
        //第1层循环，用于表示鸡翁的范围，初始化表达式的变量定义为 x=0，判断条件是x<=20
        for(int x=0; x<=20; x++) {
            //第2层循环，用于表示鸡母的范围，初始化表达式的变量定义为 y=0，判断条件是y<=33
            for(int y=0; y<=33; y++) {
                //这个时候，用于表示鸡雏的变量 z = 100 - x - y
                int z = 100 - x - y;

                //判断表达式 z%3==0 和表达式 5*x + 3*y + z/3 = 100 是否同时成立
                if(z%3==0 && 5*x+3*y+z/3==100) {
                    System.out.println(x+","+y+","+z);
                }
            }
        }
    }
}
```

2.6数组元素求和【应用】

2.6.1案例需求

有这样的一个数组，元素是{68,27,95,88,171,996,51,210}。求出该数组中满足要求的元素和， 要求是：求和的元素个位和十位都不能是7，并且只能是偶数

2.6.2代码实现

```
/*
   思路：
        1:定义一个数组，用静态初始化完成数组元素的初始化
        2:定义一个求和变量，初始值是0
        3:遍历数组，获取到数组中的每一个元素
        4:判断该元素是否满足条件，如果满足条件就累加
        5:输出求和变量的值
 */
public class Test06 {
    public static void main(String[] args) {
        //定义一个数组，用静态初始化完成数组元素的初始化
        int[] arr = {68, 27, 95, 88, 171, 996, 51, 210};

        //定义一个求和变量，初始值是0
        int sum = 0;
```

```
        //遍历数组，获取到数组中的每一个元素
        for(int x=0; x<arr.length; x++) {
            //判断该元素是否满足条件，如果满足条件就累加
            if(arr[x]%10!=7 && arr[x]/10%10!=7 && arr[x]%2==0) {
                sum += arr[x];
            }
        }

        //输出求和变量的值
        System.out.println("sum:" + sum);
    }
}
```

2.7判断两个数组是否相同【应用】

2.7.1案例需求

定义一个方法，用于比较两个数组的内容是否相同

2.7.2代码实现

```
/*
    思路：
        1:定义两个数组，分别使用静态初始化完成数组元素的初始化
        2:定义一个方法，用于比较两个数组的内容是否相同
        3:比较两个数组的内容是否相同，按照下面的步骤实现就可以了
            首先比较数组长度，如果长度不相同，数组内容肯定不相同，返回false
            其次遍历，比较两个数组中的每一个元素，只要有元素不相同，返回false
            最后循环遍历结束后，返回true
        4:调用方法，用变量接收
        5:输出结果
 */
public class Test07 {
    public static void main(String[] args) {
        //定义两个数组，分别使用静态初始化完成数组元素的初始化
        int[] arr = {11, 22, 33, 44, 55};
        //int[] arr2 = {11, 22, 33, 44, 55};
        int[] arr2 = {11, 22, 33, 44, 5};


        //调用方法，用变量接收
        boolean flag = compare(arr,arr2);
        //输出结果
        System.out.println(flag);
    }

    //定义一个方法，用于比较两个数组的内容是否相同
    /*
        两个明确：
            返回值类型：boolean
            参数：int[] arr, int[] arr2
     */
```

```
    public static boolean compare(int[] arr, int[] arr2) {
        //首先比较数组长度，如果长度不相同，数组内容肯定不相同，返回false
        if(arr.length != arr2.length) {
            return false;
        }

        //其次遍历，比较两个数组中的每一个元素，只要有元素不相同，返回false
        for(int x=0; x<arr.length; x++) {
            if(arr[x] != arr2[x]) {
                return false;
            }
        }

        //最后循环遍历结束后，返回true
        return true;
    }
}
```

2.8查找元素在数组中出现的索引位置【应用】

2.8.1案例需求

已知一个数组 arr = {19, 28, 37, 46, 50}; 键盘录入一个数据，查找该数据在数组中的索引。

并在控制台输出找到的索引值。如果没有查找到，则输出-1

2.8.2代码实现

```
/*
    思路：
        1:定义一个数组，用静态初始化完成数组元素的初始化
        2:键盘录入要查找的数据，用一个变量接收
        3:定义一个索引变量，初始值为-1
        4:遍历数组，获取到数组中的每一个元素
        5:拿键盘录入的数据和数组中的每一个元素进行比较，如果值相同，就把该值对应的索引赋值给索引变量,并
结束循环
        6:输出索引变量
 */
public class Test08 {
    public static void main(String[] args) {
        //定义一个数组，用静态初始化完成数组元素的初始化
        int[] arr = {19, 28, 37, 46, 50};

        //键盘录入要查找的数据，用一个变量接收
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入要查找的数据：");
        int number = sc.nextInt();

        //调用方法
        int index = getIndex(arr, number);

        //输出索引变量
        System.out.println("index: " + index);
```

```
    }

    //查找指定的数据在数组中的索引
    /*
        两个明确：
            返回值类型：int
            参数：int[] arr, int number
     */
    public static int getIndex(int[] arr, int number) {
        //定义一个索引变量，初始值为-1
        int index = -1;

        //遍历数组，获取到数组中的每一个元素
        for(int x=0; x<arr.length; x++) {
            //拿键盘录入的数据和数组中的每一个元素进行比较，如果值相同，就把该值对应的索引赋值给索引变量,并结束循环
            if(arr[x] == number) {
                index = x;
                break;
            }
        }

        //返回索引
        return index;
    }
}
```

2.9数组元素反转【应用】

2.9.1案例需求

已知一个数组 arr = {19, 28, 37, 46, 50}; 用程序实现把数组中的元素值交换， 交换后的数组 arr = {50, 46, 37, 28, 19}; 并在控制台输出交换后的数组元素。

2.9.2代码实现

```
/*
    思路：
        1:定义一个数组，用静态初始化完成数组元素的初始化
        2:循环遍历数组，这一次初始化语句定义两个索引变量，判断条件是开始索引小于等于结束索引
        3:变量交换
        4:遍历数组
 */
public class Test09 {
    public static void main(String[] args) {
        //定义一个数组，用静态初始化完成数组元素的初始化
        int[] arr = {19, 28, 37, 46, 50};

        //调用反转的方法
        reverse(arr);

        //遍历数组
        printArray(arr);
```

```
    }

    /*
        两个明确：
            返回值类型：void
            参数：int[] arr
     */
    public static void reverse(int[] arr) {
        //循环遍历数组，这一次初始化语句定义两个索引变量，判断条件是开始索引小于等于结束索引
        for (int start = 0, end = arr.length - 1; start <= end; start++, end--) {
            //变量交换
            int temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;
        }
    }

    /*
        两个明确：
            返回值类型：void
            参数：int[] arr
     */
    public static void printArray(int[] arr) {
        System.out.print("[");

        for (int x = 0; x < arr.length; x++) {
            if (x == arr.length - 1) {
                System.out.print(arr[x]);
            } else {
                System.out.print(arr[x] + ", ");
            }
        }

        System.out.println("]");
    }
}
```

2.10评委打分【应用】

2.10.1案例需求

在编程竞赛中，有6个评委为参赛的选手打分，分数为0-100的整数分。 选手的最后得分为：去掉一个最高分和一个最低分后 的4个评委平均值 (不考虑小数部分)。

2.10.2代码实现

```
/*
    思路：
        1:定义一个数组，用动态初始化完成数组元素的初始化，长度为6
        2:键盘录入评委分数
        3:由于是6个评委打分，所以，接收评委分数的操作，用循环改进
        4:定义方法实现获取数组中的最高分(数组最大值)，调用方法
        5:定义方法实现获取数组中的最低分(数组最小值) ，调用方法
```

```
        6:定义方法实现获取数组中的所有元素的和(数组元素求和) ，调用方法
        7:按照计算规则进行计算得到平均分
        8:输出平均分
 */
public class Test10 {
    public static void main(String[] args) {
        //定义一个数组，用动态初始化完成数组元素的初始化，长度为6
        int[] arr = new int[6];

        //键盘录入评委分数
        Scanner sc = new Scanner(System.in);

        //由于是6个评委打分，所以，接收评委分数的操作，用循环改进
        for(int x=0; x<arr.length; x++) {
            System.out.println("请输入第" + (x + 1) + "个评委的打分：");
            arr[x] = sc.nextInt();
        }

        //printArray(arr);

        //定义方法实现获取数组中的最高分(数组最大值)，调用方法
        int max = getMax(arr);

        //定义方法实现获取数组中的最低分(数组最小值) ，调用方法
        int min = getMin(arr);

        //定义方法实现获取数组中的所有元素的和(数组元素求和) ，调用方法
        int sum = getSum(arr);

        //按照计算规则进行计算得到平均分
        int avg = (sum - max - min) / (arr.length - 2);

        //输出平均分
        System.out.println("选手的最终得分是：" + avg);

    }

    /*
        两个明确：
            返回值类型：int
            参数：int[] arr
     */
    public static int getSum(int[] arr) {
        int sum = 0;

        for(int x=0; x<arr.length; x++) {
            sum += arr[x];
        }

        return sum;
    }

    /*
```

```
        两个明确：
            返回值类型：int
            参数：int[] arr
     */
    public static int getMin(int[] arr) {
        int min = arr[0];

        for(int x=1; x<arr.length; x++) {
            if(arr[x] < min) {
                min = arr[x];
            }
        }

        return min;
    }

    /*
        两个明确：
            返回值类型：int
            参数：int[] arr
     */
    public static int getMax(int[] arr) {
        int max = arr[0];

        for(int x=1; x<arr.length; x++) {
            if(arr[x] > max) {
                max = arr[x];
            }
        }

        return max;
    }

    //遍历数组
    public static void printArray(int[] arr) {
        System.out.print("[");

        for (int x = 0; x < arr.length; x++) {
            if (x == arr.length - 1) {
                System.out.print(arr[x]);
            } else {
                System.out.print(arr[x] + ", ");
            }
        }

        System.out.println("]");
    }
}
```