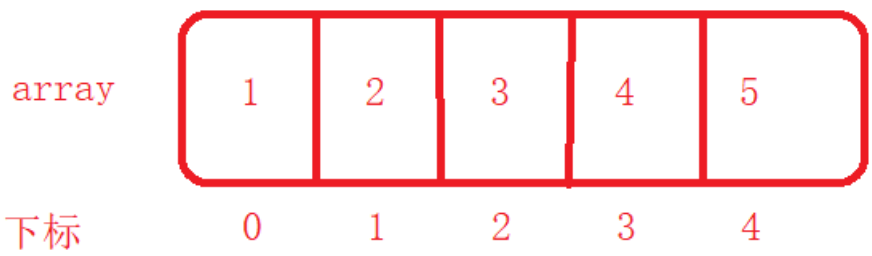
第十五章 数组与枚举

1. 数组的概述

1. 数组是什么
    a. 变量只能存储一个数据，数组可以存储多个数据
    b. 数组也是存储数据的容器
    c. 数组也是一种数据结构
2. 数字的注意事项
    a. 数组的容量大小一旦指定，不可以改变
    b. 数组有两部分组成
        i. 数组名称，内存中开辟空间的容器名称
        ii. 下标：下标从0开始，连续的
        iii. 数组定位一个元素，通过数值名称+下标
3. 数组的优,缺点
    a. 优点：查询和修改比较快，查询的时间复杂度是O(1)
    b. 缺点：删除和插入比较慢，因为数组长度不可变，需要创建新数组把老数组数据放入，实现删除和插入的效果
    c. 总结；如果频繁的查询和修改，建议使用数组，如果频繁的删除和插入不建议使用
4. 数组是引用类型，不属于基本类型

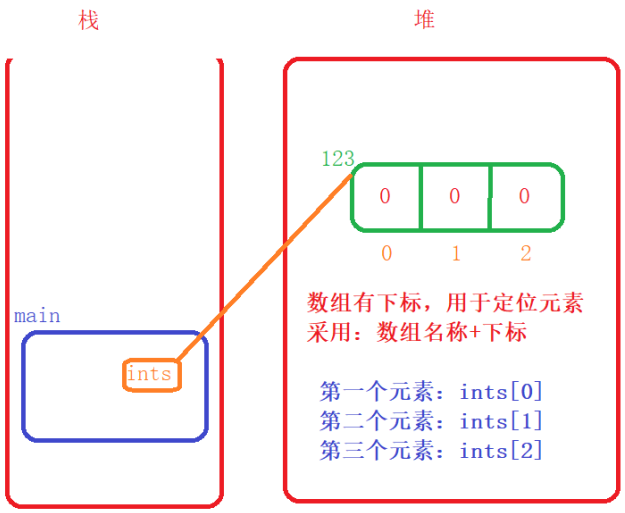


2. 数组的内存分析

代码块

```java
package com.powernode.array19;

public class Test {
    public static void main(String[] args) {
        /**
         * 1.数组的动态初始化
         *     1.语法：数据类型 [] 数组名称 = new 数据类型[长度]
         *     2.数据类型：基本类型和引用类型
         *     3.长度：存储元素的个数
         * 2.注意：动态初始化必须为数组指定长度
         */
        //Array initializer expected 期望数组初始化项
        int[] ints = new int[3];
        System.out.println(ints);

        System.out.println(ints[0]);
        System.out.println(ints[1]);
        System.out.println(ints[2]);
    }
}
```


```java
package com.powernode.array19;

public class Test {
    public static void main(String[] args) {
        /**
         * 1.数组的动态初始化
         *     1.语法：数据类型 [] 数组名称 = new 数据类型[长度]
         *     2.数据类型：基本类型和引用类型
         *     3.长度：存储元素的个数
         * 2.注意：动态初始化必须为数组指定长度
         */
        //Array initializer expected 期望数组初始化项
        int[] ints = new int[3];
        System.out.println(ints);

        System.out.println(ints[0]);
        System.out.println(ints[1]);
        System.out.println(ints[2]);
    }
}
```

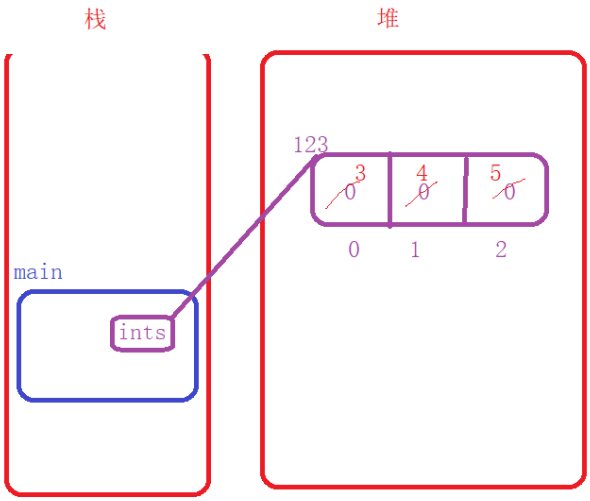
3. 数组存取数据

代码块

```java
package com.powernode.array19;

public class Test02 {
    public static void main(String[] args) {
```

```
        //1.语法：数据类型 [] 数组名称 = new 数据类型[长度]
        int[] ints = new int[3];
        //int ints1 [] = new int[3];不推荐使用
        //2.向数组中存数据
        ints[0] = 3;
        ints[1] = 4;
        ints[2] = 5;
        //3.从数组中存数据
        System.out.println("ints[0] = " + ints[0]);
        System.out.println("ints[1] = " + ints[1]);
        System.out.println("ints[2] = " + ints[2]);
    }
}
```

```
package com.powernode.array19;

public class Test02 {
    public static void main(String[] args) {
        //1.语法：数据类型 [] 数组名称 = new 数据类型[长度]
        int[] ints = new int[3];
        //int ints1 [] = new int[3];不推荐使用
        //2.向数组中存数据
        ints[0] = 3;
        ints[1] = 4;
        ints[2] = 5;
        //3.从数组中存数据
        System.out.println("ints[0] = " + ints[0]);
        System.out.println("ints[1] = " + ints[1]);
        System.out.println("ints[2] = " + ints[2]);
    }
}
```



4. 数组异常

4.1 常见本态异常

- *ArrayIndexOutOfBoundsException*
- *NullPointerException*

代码块

```
package com.powernode.array19;

public class Test03 {
    public static void main(String[] args) {
        //1.语法：数据类型 [] 数组名称 = new 数据类型[长度]
        int[] ints = new int[3];
        //int ints1 [] = new int[3];不推荐使用
```

```
        //2.向数组中存数据
        ints[0] = 3;
        ints[1] = 4;
        ints[2] = 5;
        /**
         * 数组下标越界异常:
         *   1.ArrayIndexOutOfBoundsException:数组越界（用非法索引访问数组时抛出的异常）
         *   2.什么原因造成的，超出了下标的取值范围
         *     1.new int[3] ,下标的取值范围[0,2],包含0和2
         *     2.ints[3] 不在取值范围[0,2]内，所以报错
         *   3.出现了错误，JVM终止工作，出现异常后的代码无法执行
         */
        //ints[3] = 6;

        //3.从数组中存数据
        System.out.println("ints[0] = " + ints[0]);
        System.out.println("ints[1] = " + ints[1]);
        System.out.println("ints[2] = " + ints[2]);
        System.out.println("-----------------");
        int[] array = new int[3];
        array = null;
        //.NullPointerException
        array[1] = 2;

    }
}
```

4.2 多态数组存储异常

代码块

```
package com.powernode.array19;
class Animal{}
class Dog extends Animal{}
class Cat extends Animal{}
public class Test04 {
    public static void main(String[] args) {
        int[] ints = new int[3];
        ints [0] = 1;
        //多态数组
        Animal[] animals = new Dog[3];
        animals[0] = new Dog();
        //ArrayStoreException:数组存储异常
        animals[1] = new Cat();
    }
```

```
15  }
```

5. 数组静态初始化

代码块

```
1   package com.powernode.array20;
2
3   public class Test01 {
4       public static void main(String[] args) {
5           //1.数组静态初始化
6           int[] ints = {1, 2, 3};
7           System.out.println("ints[0] = " + ints[0]);
8           System.out.println("ints[1] = " + ints[1]);
9           System.out.println("ints[2] = " + ints[2]);
10          /**
11           * 2.什么情况下，使用动态初始化，什么情况下使用动态初始化
12           *     1.一般情况下确定了值，使用静态初始化
13           *     2.一般情况下确定了长度，不确定值，使用动态初始化
14           */
15          char[] chars = {'a', 'b', 'c'};
16          char[] chars1 = new char[3];
17          chars1[0] = 'a';
18          chars1[1] = 'a';
19          chars1[2] = 'a';
20          //3.引用类型数组
21          String[] strings = {"abc", "def", "xyz"};
22
23          Object[] objects = new String[3];
24          objects[0] = "abc";
25      }
26
27  }
```

6. 数组遍历

6.1 普通for循环遍历

代码块

```
1   package com.powernode.array20;
2
3   public class Test02 {
4       public static void main(String[] args) {
```

```
        int[] ints = {1, 2, 3,8};
        System.out.println("ints[0] = " + ints[0]);
        System.out.println("ints[1] = " + ints[1]);
        System.out.println("ints[2] = " + ints[2]);
        System.out.println("------------");
        /**
         * 1.数组对象有一个隐含的属性：length
         * 2.通过length可以获得数组的长度（可存储元素的个数）
         */
        System.out.println("ints.length = " + ints.length);
        for (int i = 0; i < ints.length; i++) {
            System.out.println(ints[i]);
        }

    }

}
```

6.2 增强for循环遍历

代码块

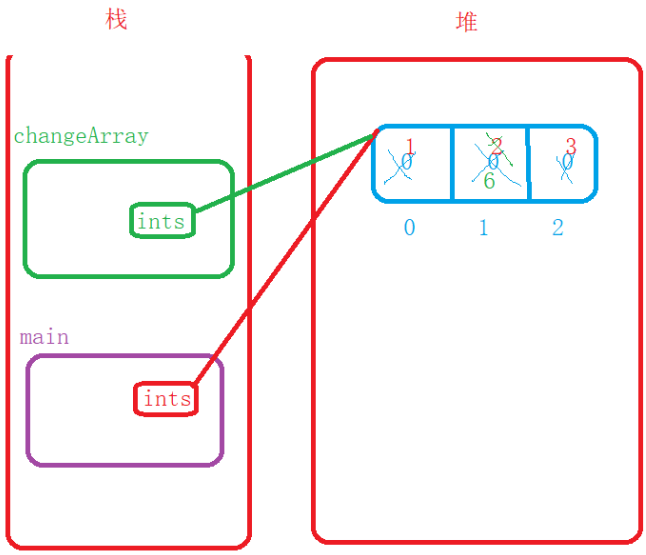
```
package com.powernode.array20;

public class Test03 {
    public static void main(String[] args) {
        int[] ints = {1, 2, 3};

        /**
         * 1.语法：for(数据类型 变量名称:数组名称){}
         * 2.执行过程：通过数组名称，拿到数组中的每个元素依次给变量名称赋值
         */
        for(int i:ints){
            System.out.println(i);
        }

    }

}
```

7. 数组的地址传递

代码块

```java
package com.powernode.array21;

public class Test01 {
    public static void main(String[] args) {
        int[] ints = {1, 2, 3};
        changeArray(ints);
        System.out.println(ints[1]);//6
    }

    public static void changeArray(int[] ints) {
        ints[1] = 6;
    }
}
```


```java
package com.powernode.array21;

public class Test01 {
    public static void main(String[] args) {
        int[] ints = {1, 2, 3};
        changeArray(ints);
        System.out.println(ints[1]);//6
    }

    public static void changeArray(int[] ints) {
        ints[1] = 6;
    }
}
```

8. 数组的扩容

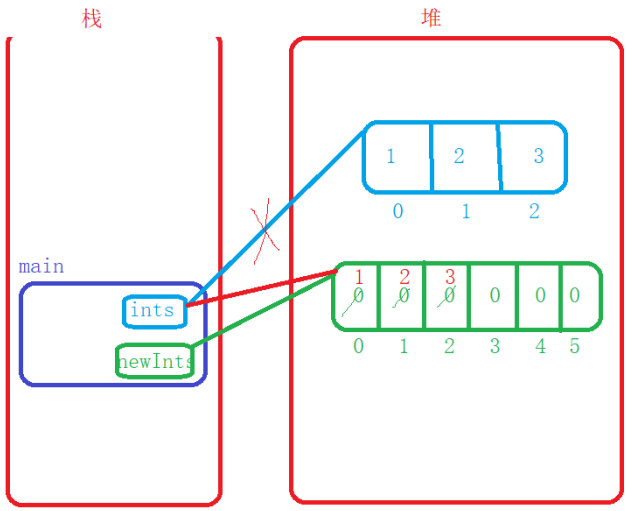
8.1 手动扩容

代码块

```java
package com.powernode.array21;

public class Test02 {
    public static void main(String[] args) {
        int[] ints = {1, 2, 3};
        /**
         * 1.长度不够用了，对数组进行扩容
         * 2.扩容的思路
```

```
 9              *     1.创建一个新的数组，长度是老数组的2倍
10              *     2.把老数组的数据取出来，放入新数组
11              */
12             //1.创建一个新的数组，长度是老数组的2倍
13             int[] newInts = new int[ints.length << 2];//扩容2倍
14             //2.把老数组的数据取出来，放入新数组
15             for (int i = 0; i < ints.length; i++) {
16                 newInts[i] = ints[i];
17             }
18             //把老数组的引用指向新数组的地址
19             ints = newInts;
20         }
21
22
23     }
```

```
package com.powernode.array21;

public class Test02 {
    public static void main(String[] args) {
        int[] ints = {1, 2, 3};
        /**
         * 1.长度不够用了，对数组进行扩容
         * 2.扩容的思路
         *     1.创建一个新的数组，长度是老数组的2倍
         *     2.把老数组的数据取出来，放入新数组
         */
        //1.创建一个新的数组，长度是老数组的2倍
        int[] newInts = new int[ints.length << 2];//扩容2倍
        //2.把老数组的数据取出来，放入新数组
        for (int i = 0; i < ints.length; i++) {
            newInts[i] = ints[i];
        }
        //把老数组的引用指向新数组的地址
        ints = newInts;
    }


}
```



8.2 JDK方法扩容

代码块

```
1   package com.powernode.array21;
2
3   public class Test03 {
4       public static void main(String[] args) {
5           int[] ints = {1, 2, 3};
6           /**
7            * 1.System.arraycopy():JDK提供的方法
8            * 2.学习方法的方法
9            *     1.是否为静态
```

```
10            *    2.返回类型
11            *    3.方法参数:
12            *       1.Object src : 源数组
13            *       2.int  srcPos: 源数组的起始位置
14            *       3.Object dest: 目标数组
15            *       4.int destPos: 目标数组的起始位置
16            *       5.int length: 需要赋值的元素个数
17            */
18           int[] newInts = new int[ints.length << 1];
19           //
20           System.arraycopy(ints,0,newInts,0,ints.length);
21           ints = newInts;
22           for (int anInt : ints) {
23               System.out.println(anInt);
24           }
25       }
26
27
28   }
```

8.3 手动封装arraycopy

代码块

```
1   package com.powernode.array21;
2
3   public class Test04 {
4       public static void main(String[] args) {
5           int[] ints = {1, 2, 3};
6           /**
7            * 1.System.arraycopy():JDK提供的方法
8            * 2.学习方法的方法
9            *    1.是否为静态
10           *    2.返回类型
11           *    3.方法参数:
12           *       1.Object src : 源数组
13           *       2.int  srcPos: 源数组的起始位置
14           *       3.Object dest: 目标数组
15           *       4.int destPos: 目标数组的起始位置
16           *       5.int length: 需要赋值的元素个数
17           */
18          int[] newInts = new int[ints.length << 1];
19          //
20          //System.arraycopy(ints,0,newInts,0,ints.length);
21          System01.arraycopy(ints,0,newInts,0,3);
22
```

```
23          for (int anInt : newInts) {
24              System.out.println(anInt);
25          }
26      }
27
28
29  }
30  class System01{
31      public static void arraycopy01(Object src,Object desc,int length){
32          //检查源数组和目标数组是否为null
33          if (src == null || desc == null) {
34              System.out.println("源数组和目标数组都不能为null");
35              return;
36          }
37          //检查src 和 desc 是否为数组类型
38          if (!src.getClass().isArray() || !desc.getClass().isArray()) {
39              System.out.println("源数组和目标数组都必须是数组");
40              return;
41          }
42          //判断源数组和目标数组是否为int[]
43          if (src instanceof int [] && desc instanceof int []) {
44              int [] ints = (int[]) src;
45              int[] newInts = (int[]) desc;
46              for (int i = 0; i < length; i++) {
47                  newInts[i] = ints[i];
48              }
49          }
50
51      }
52
53      /**
54       * @param src : 源数组
55       * @param srcPos :  起始位置
56       * @param desc :  目标数组
57       * @param destPos: 起始位置
58       * @param length: copy元素的个数
59       */
60      public static void arraycopy(Object src,int srcPos,Object desc,int
    destPos,int length){
61          //检查源数组和目标数组是否为null
62          if (src == null || desc == null) {
63              System.out.println("源数组和目标数组都不能为null");
64              return;
65          }
66          //检查src 和 desc 是否为数组类型
67          if (!src.getClass().isArray() || !desc.getClass().isArray()) {
68              System.out.println("源数组和目标数组都必须是数组");
```

```
            return;
        }
        //判断源数组和目标数组是否为int[]
        if (src instanceof int [] && desc instanceof int []) {
            int [] ints = (int[]) src;
            int[] newInts = (int[]) desc;
            for (int i = 0; i < length; i++) {
                newInts[destPos++] = ints[srcPos++];
            }
        }

    }
}
```

9. 向数组中插入数据

9.1 向数组中插入数据 （main）

代码块

```
package com.powernode.array01;

public class Test01 {
    public static void main(String[] args) {
        int[] ints = {1, 2, 3, 4, 5};
        /**
         * 1.向数组下标为3的元素插入6
         * 2.插入后的结果为：{1, 2, 3, 6, 4, 5};
         * 3.实施步骤：
         *     1.创建一个新的数组，新数组的长度是老数组的长度+1
         *           {0, 0, 0, 0, 0, 0};
         *     2.把老数组的前3个元素放入 新数组
         *           {1, 2, 3, 0, 0, 0};
         *     3.在新数组中下标为3的元素插入6
         *           {1, 2, 3, 6, 0, 0};
         *     4.再把老数组的后两位元素放入新数组
         *           {1, 2, 3, 6, 4, 5};
         */
        //1.创建一个新的数组，新数组的长度是老数组的长度+1
        int[] newInts = new int[ints.length + 1];//{0, 0, 0, 0, 0, 0};
        //2.把老数组的前3个元素放入 新数组
        System.arraycopy(ints,0,newInts,0,3);//{1, 2, 3, 0, 0, 0};
        //3.在新数组中下标为3的元素插入6
        newInts[3] = 6;//{1, 2, 3, 6, 0, 0};
```

```
        //4.再把老数组的后两位元素放入新数组
        System.arraycopy(ints,3,newInts,4,2);

        for (int anInt : newInts) {
            System.out.print(anInt + "\t");
        }


    }
}
```

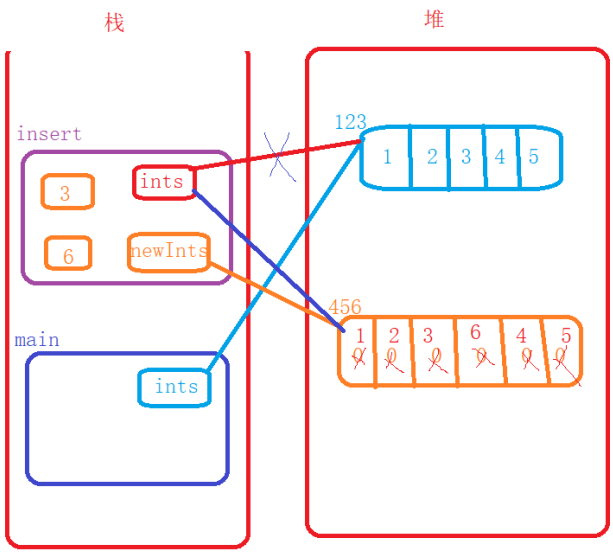
9.2 封装方法（内存分析）

代码块

```
package com.powernode.array01;

public class Test02 {
    public static void main(String[] args) {
        int[] ints = {1, 2, 3, 4, 5};
        insert(ints,3,6);
        for (int anInt : ints) {
            System.out.print(anInt + "\t");
        }
    }

    /**
     * @param ints : 源数组
     * @param index: 需要插入的下标
     * @param num: 需要插入的数据
     */
    public static void insert(int [] ints,int index,int num){
        /**
         * 1.向数组下标为3的元素插入6
         * 2.插入后的结果为：{1, 2, 3, 6, 4, 5};
         * 3.实施步骤：
         *     1.创建一个新的数组，新数组的长度是老数组的长度+1
         *           {0, 0, 0, 0, 0, 0};
         *     2.把老数组的前3个元素放入 新数组
         *           {1, 2, 3, 0, 0, 0};
         *     3.在新数组中下标为3的元素插入6
         *           {1, 2, 3, 6, 0, 0};
         *     4.再把老数组的后两位元素放入新数组
         *           {1, 2, 3, 6, 4, 5};
```

```
30              */
31             //1.创建一个新的数组，新数组的长度是老数组的长度+1
32             int[] newInts = new int[ints.length + 1];//{0, 0, 0, 0, 0, 0};
33             //2.把老数组的前3个元素放入 新数组
34             System.arraycopy(ints,0,newInts,0,3);//{1, 2, 3, 0, 0, 0};
35             //3.在新数组中下标为3的元素插入6
36             newInts[3] = 6;//{1, 2, 3, 6, 0, 0};
37             //4.再把老数组的后两位元素放入新数组
38             System.arraycopy(ints,3,newInts,4,2);
39
40             ints = newInts;
41         }
42     }
```

```
public class Test02 {
    public static void main(String[] args) {
        int[] ints = {1, 2, 3, 4, 5};
        insert(ints,3,6);
        for (int anInt : ints) {
            System.out.print(anInt + "\t");
        }
    }
    /**
     * @param ints : 源数组
     * @param index: 需要插入的下标
     * @param num: 需要插入的数据
     */
    public static void insert(int [] ints,int index,int num){
        //1.创建一个新的数组，新数组的长度是老数组的长度+1
        int[] newInts = new int[ints.length + 1];//{0, 0, 0, 0, 0, 0};
        //2.把老数组的前3个元素放入 新数组
        System.arraycopy(ints,0,newInts,0,3);//{1, 2, 3, 0, 0, 0};
        //3.在新数组中下标为3的元素插入6
        newInts[3] = 6;//{1, 2, 3, 6, 0, 0};
        //4.再把老数组的后两位元素放入新数组
        System.arraycopy(ints,3,newInts,4,2);
        ints = newInts;
    }
```



9.3 插入方法完善（return数组）

代码块

```
1   package com.powernode.array01;
2
3   public class Test03 {
4       public static void main(String[] args) {
5           int[] ints = {1, 2, 3, 4, 5};
6           ints = insert(ints,3,6);
7           for (int anInt : ints) {
8               System.out.print(anInt + "\t");
9           }
10      }
11
```

```java
    /**
     * @param ints : 源数组
     * @param index: 需要插入的下标
     * @param num: 需要插入的数据
     */
    public static int [] insert(int [] ints,int index,int num){
        /**
         * 1.向数组下标为3的元素插入6
         * 2.插入后的结果为：{1, 2, 3, 6, 4, 5};
         * 3.实施步骤：
         *     1.创建一个新的数组，新数组的长度是老数组的长度+1
         *           {0, 0, 0, 0, 0, 0};
         *     2.把老数组的前3个元素放入 新数组
         *           {1, 2, 3, 0, 0, 0};
         *     3.在新数组中下标为3的元素插入6
         *           {1, 2, 3, 6, 0, 0};
         *     4.再把老数组的后两位元素放入新数组
         *           {1, 2, 3, 6, 4, 5};
         */
        //1.创建一个新的数组，新数组的长度是老数组的长度+1
        int[] newInts = new int[ints.length + 1];//{0, 0, 0, 0, 0, 0};
        //2.把老数组的前3个元素放入 新数组
        System.arraycopy(ints,0,newInts,0,3);//{1, 2, 3, 0, 0, 0};
        //3.在新数组中下标为3的元素插入6
        newInts[3] = 6;//{1, 2, 3, 6, 0, 0};
        //4.再把老数组的后两位元素放入新数组
        System.arraycopy(ints,3,newInts,4,2);

        return newInts;
    }
}
```

9.4 插入方法完善（最终）

代码块

```java
package com.powernode.array01;

public class Test04 {
    public static void main(String[] args) {
        int[] ints = {1, 2, 3, 4, 5};
        ints = insert(ints,3,6);
        for (int anInt : ints) {
            System.out.print(anInt + "\t");
        }
    }
```

```
11
12      /**
13       * @param ints : 源数组
14       * @param index: 需要插入的下标
15       * @param num: 需要插入的数据
16       */
17      public static int [] insert(int [] ints,int index,int num){
18          /**
19           * -  int[] ints = {1, 2, 3, 4, 5};
20           * 1.向数组下标为3的元素插入6
21           * 2.插入后的结果为：{1, 2, 3, 6, 4, 5};
22           * 3.实施步骤：
23           *     1.创建一个新的数组，新数组的长度是老数组的长度+1
24           *           {0, 0, 0, 0, 0, 0};
25           *     2.把老数组的前3个元素放入 新数组
26           *           {1, 2, 3, 0, 0, 0};
27           *     3.在新数组中下标为3的元素插入6
28           *           {1, 2, 3, 6, 0, 0};
29           *     4.再把老数组的后两位元素放入新数组
30           *           {1, 2, 3, 6, 4, 5};
31           */
32          //1.创建一个新的数组，新数组的长度是老数组的长度+1
33          int[] newInts = new int[ints.length + 1];//{0, 0, 0, 0, 0, 0};
34          //2.把老数组的前3个元素放入 新数组
35          System.arraycopy(ints,0,newInts,0,index);//{1, 2, 3, 0, 0, 0};
36          //3.在新数组中下标为3的元素插入6
37          newInts[index] = num;//{1, 2, 3, 6, 0, 0};
38          //4.再把老数组的后两位元素放入新数组
39          /**
40           * {1, 2, 3, 4, 5}
41           * 0   1  2  3  4
42           * 算法：
43           *    1.向数组 3 的位置插入 后面还有 2 个元素 ：  length(5) - 3 = 2
44           *    2.向数组 4 的位置插入 后面还有 1 个元素 ：  length(5) - 4 = 1
45           *    3.总结：
46           *       剩余的元素 = 数组长度 - 插入的位置
47           *
48           *
49           */
50          System.arraycopy(ints,index,newInts,index + 1,ints.length - index);
51
52          return newInts;
53      }
54  }
```

9.5 使用JDK提供的方法

代码块

```
package com.powernode.array01;

public class Test05 {
    public static void main(String[] args) {
        int[] ints = {1, 2, 3, 4, 5};
        ints = insert(ints,1,6);
        for (int anInt : ints) {
            System.out.print(anInt + "\t");
        }
    }

    /**
     * @param ints : 源数组
     * @param index: 需要插入的下标
     * @param num: 需要插入的数据
     */
    public static int [] insert(int [] ints,int index,int num){
        /**
         * -  int[] ints = {1, 2, 3, 4, 5};
         * 1.向数组下标为3的元素插入6
         * 2.插入后的结果为：{1, 2, 3, 6, 4, 5};
         * 3.实施步骤：
         *     1.创建一个新的数组，新数组的长度是老数组的长度+1
         *           {0, 0, 0, 0, 0, 0};
         *     2.把老数组的前3个元素放入 新数组
         *           {1, 2, 3, 0, 0, 0};
         *     3.在新数组中下标为3的元素插入6
         *           {1, 2, 3, 6, 0, 0};
         *     4.再把老数组的后两位元素放入新数组
         *           {1, 2, 3, 6, 4, 5};
         */
        //1.创建一个新的数组，新数组的长度是老数组的长度+1
        int[] newInts = new int[ints.length + 1];//{0, 0, 0, 0, 0, 0};
        //2.把老数组的前3个元素放入 新数组
        //System.arraycopy(ints,0,newInts,0,index);//{1, 2, 3, 0, 0, 0};
        for (int i = 0; i < index; i++) {
            newInts[i] = ints[i];
        }
        //3.在新数组中下标为3的元素插入6
        newInts[index] = num;//{1, 2, 3, 6, 0, 0};
        //4.再把老数组的后两位元素放入新数组
        /**
         * {1, 2, 3, 4, 5}
         * 0   1  2  3  4
         * 算法：
```

```
46              *    1.向数组 3 的位置插入 后面还有 2 个元素 :  length(5) - 3 = 2
47              *    2.向数组 4 的位置插入 后面还有 1 个元素 :  length(5) - 4 = 1
48              *    3.总结:
49              *        剩余的元素 = 数组长度 - 插入的位置
50              *
51              *
52              */
53             //System.arraycopy(ints,index,newInts,index + 1,ints.length - index);
54             for (int i = index; i < ints.length; i++) {
55                 newInts[i + 1] = ints[i];
56             }
57             return newInts;
58         }
59     }
```

作业

1. 动态数组 存储10个数，使用普通for和增强for输出
2. 静态数组，存储{1,6,2,4,3,0}，使用普通for和增强for输出
3. 数组扩容
4. 数组插入
5. 封装数组插入的方法
6. 尝试删除数组中元素（明天讲）

10. 删除数组中元素

10.1 方式一（分开copy）

代码块

```
1   package com.powernode.array02;
2
3   public class Test01 {
4       public static void main(String[] args) {
5           int[] ints = {1, 2, 3, 4, 5, 6};
6           /**
7            * 1.删除下标为 3 的元素 4
8            * 2.删除后的结果为: {1, 2, 3, 5, 6};
9            * 3.删除步骤:
10           *    1.创建一个新的数组，长度为老数组长度 - 1
```

```
         *       {0, 0, 0, 0, 0};
         *    2.把数组的前3个元素放入新数组
         *       {1, 2, 3, 0, 0};
         *    3.把数组的后2个元素放入新数组
         *       {1, 2, 3, 5, 6};
         */
        //1.创建一个新的数组，长度为老数组长度 - 1
        int[] newInts = new int[ints.length - 1];
        //2.把数组的前3个元素放入新数组
        System.arraycopy(ints,0,newInts,0,3);
        //3.把数组的后2个元素放入新数组
        System.arraycopy(ints,4,newInts,3,2);
        for (int newInt : newInts) {
            System.out.print(newInt + "\t");
        }
    }
}
```

10.2 封装方法

代码块

```
package com.powernode.array02;

public class Test02 {
    public static void main(String[] args) {
        int[] ints = {1, 2, 3, 4, 5, 6};
        /**
         * 1.删除下标为 3 的元素 4
         * 2.删除后的结果为：{1, 2, 3, 5, 6};
         * 3.删除步骤：
         *    1.创建一个新的数组，长度为老数组长度 - 1
         *       {0, 0, 0, 0, 0};
         *    2.把数组的前3个元素放入新数组
         *       {1, 2, 3, 0, 0};
         *    3.把数组的后2个元素放入新数组
         *       {1, 2, 3, 5, 6};
         */
        ints = deleteElementByIndex(ints, 3);
        for (int num : ints) {
            System.out.print(num + "\t");
        }
    }

    /**
     *
```

```
25          * @param ints : 源数组
26          * @param index: 需要删除的索引
27          * @return
28          */
29         public static int [] deleteElementByIndex(int [] ints,int index){
30             //1.创建一个新的数组，长度为老数组长度 - 1
31             int[] newInts = new int[ints.length - 1];
32             //2.把数组的前3个元素放入新数组
33             System.arraycopy(ints,0,newInts,0,index);
34             //3.把数组的后2个元素放入新数组
35             System.arraycopy(ints,index + 1,newInts,index,ints.length - index - 1);
36             /**
37              * 老数组：{1, 2, 3, 4, 5, 6};
38              *  下标：  0   1  2  3  4  5
39              *  1.删除下标为 2 的元素：后面还有  3 个元素 ：  6 - 2 - 1 = 3
40              *  2.删除下标为 3 的元素：后面还有  2 个元素 ：  6 - 3 - 1 = 2
41              *  3.删除下标为 4 的元素：后面还有  1 个元素 ：  6 - 4 - 1 = 1
42              *  总结：ints.length - index - 1
43              */
44             return newInts;
45         }
46     }
```

10.3 方式二（移位删除）

代码块

```
1   package com.powernode.array02;
2
3   import java.util.Arrays;
4
5   public class Test03 {
6       public static void main(String[] args) {
7
8           /**
9            * 1.删除下标为 3 的元素 4
10           * 2.删除后的结果为：{1, 2, 3, 5, 6};
11           * 3.删除步骤：
12           *    1.把老数组的后两位往前挪一位
13           *       {1, 2, 3, 5, 6, 6};
14           *    2.把老数组的前5位，复制到新数组中：{1, 2, 3, 5, 6};
15           */
16          int[] ints = {1, 2, 3, 4, 5, 6};
17          //1.把老数组的后两位往前挪一位
18          System.arraycopy(ints,4,ints,3,2);
19          //2.把老数组的前5位，复制到新数组中：{1, 2, 3, 5, 6};
```

```
        /**
         * 学习方法的方法:
         *   1.是否为静态 :  staitc
         *   2.返回类型 :  int[]
         *   3.参数列表:
         *      int[] original :  源数组
         *      int newLength :  需要复制的元素个数（新数组的长度）
         */
        ints = Arrays.copyOf(ints, 5);
        for (int anInt : ints) {
            System.out.print(anInt + "\t");
        }
    }


}
```

10.4 封装方法

代码块

```
package com.powernode.array02;

import java.util.Arrays;

public class Test04 {
    public static void main(String[] args) {

        /**
         * 1.删除下标为 3 的元素 4
         * 2.删除后的结果为：{1, 2, 3, 5, 6};
         * 3.删除步骤:
         *    1.把老数组的后两位往前挪一位
         *       {1, 2, 3, 5, 6, 6};
         *    2.把老数组的前5位，复制到新数组中：{1, 2, 3, 5, 6};
         */
        int[] ints = {1, 2, 3, 4, 5, 6};
        ints = deleteElementByIndex(ints, 3);
        for (int anInt : ints) {
            System.out.print(anInt + "\t");
        }
    }
    public static int [] deleteElementByIndex(int [] ints,int index){
        //1.把老数组的后两位往前挪一位
        System.arraycopy(ints,index + 1,ints,index,ints.length - index - 1);
        //2.把老数组的前5位，复制到新数组中：{1, 2, 3, 5, 6};
```

```
        /**
         * 学习方法的方法:
         *   1.是否为静态 :  staitc
         *   2.返回类型 :  int[]
         *   3.参数列表:
         *      int[] original :  源数组
         *      int newLength :  需要复制的元素个数（新数组的长度）
         */
        ints = Arrays.copyOf(ints, ints.length - 1);
        return ints;
    }

}
```

11. 可变参数

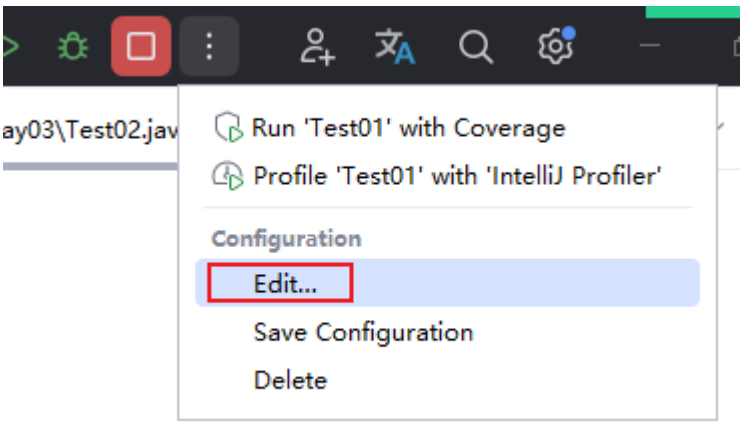
代码块

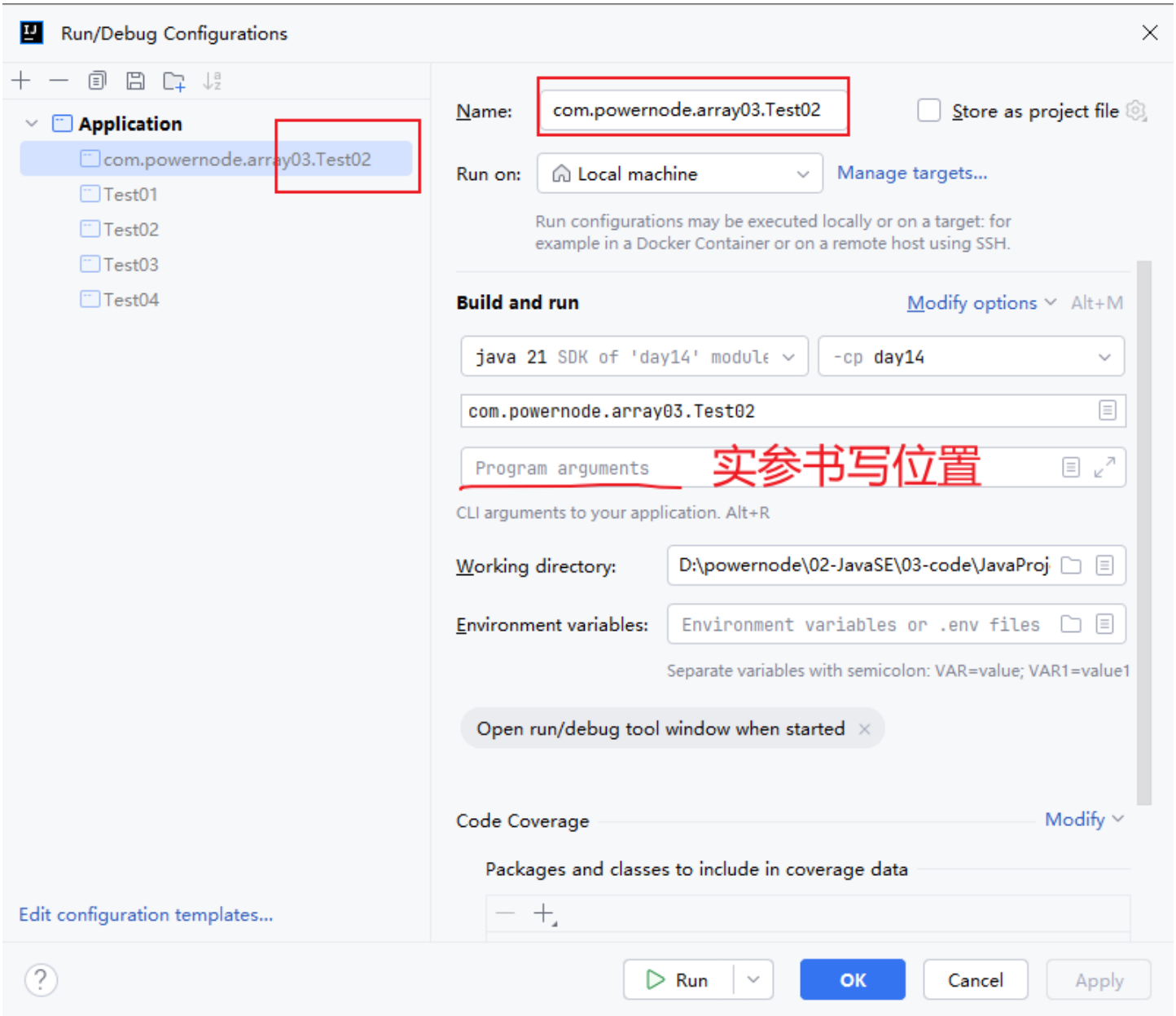
```
package com.powernode.array03;

public class Test01 {

    public static void main(String[] args) {
        add(2,3);
        add(2,3,4);
    }
   /* //定义一个方法，计算 2 个整数相加
    public static void add(int x,int y){
        System.out.println(x + y);
    }
    //定义一个方法，计算 3 个整数相加
    public static void add(int x,int y,int z){
        System.out.println(x + y + z);
    }*/
    //定义一个方法，计算 n 个整数相加
    public static void add(int ... ints){//可变参数是数组类型，在形参可以这么写
       int sum = 0;
        for (int i = 0; i < ints.length; i++) {
            sum += ints[i];
        }
        System.out.println(sum);
    }
}
```

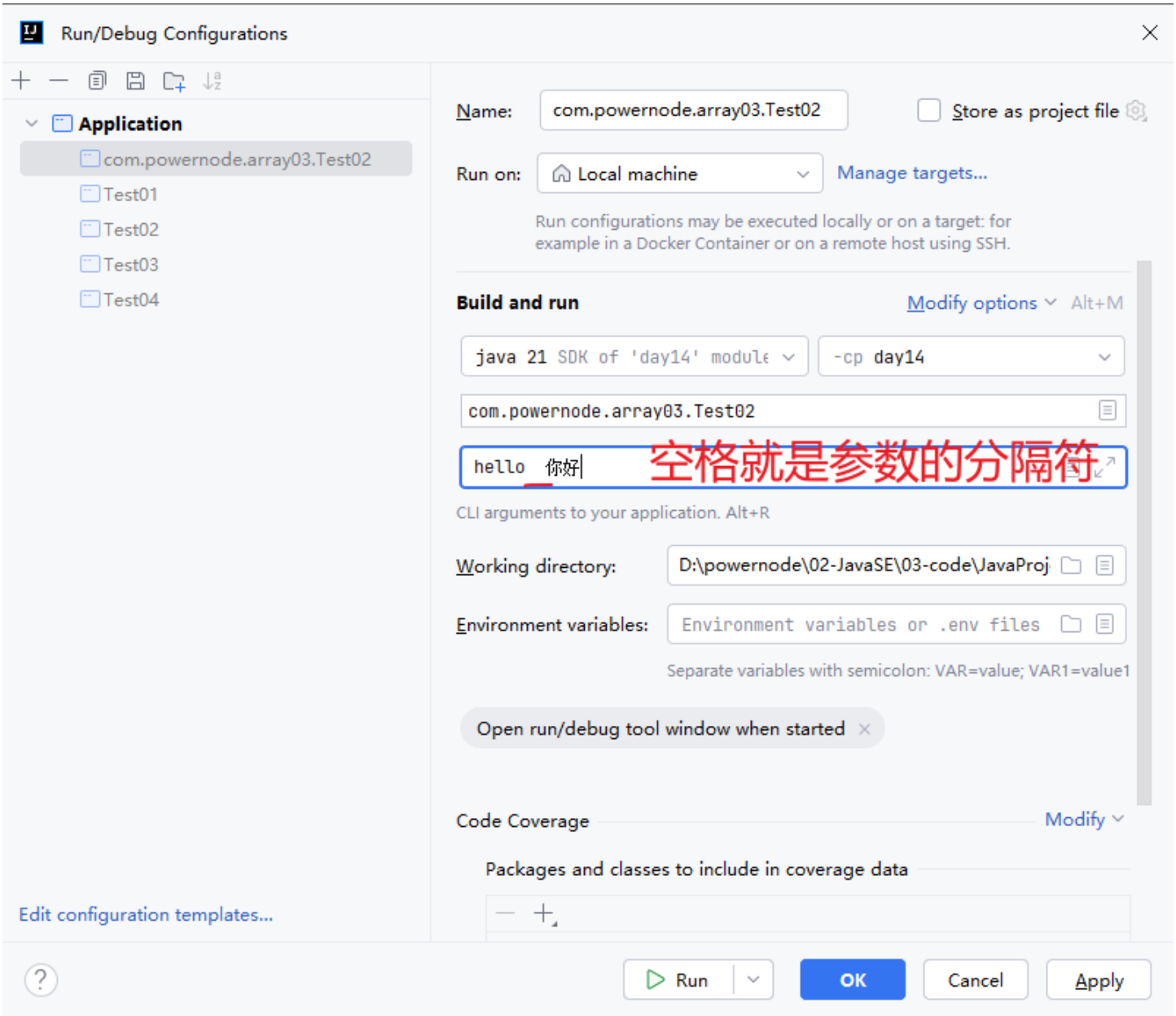
12. main方法的参数（了解）

代码块

```
1   package com.powernode.array03;
2
3   public class Test02 {
4
5       public static void main(String[] args) {
6           System.out.println("args[0] = " + args[0]);
7           System.out.println("args[1] = " + args[1]);
8       }
9
10  }
```




Run/Debug Configurations
Application
com.powernode.array03.Test02
Test01
Test02
Test03
Test04
Edit configuration templates...
Name:
com.powernode.array03.Test02
Store as project file
Run on:
Local machine
Manage targets...
Run configurations may be executed locally or on a target: for example in a Docker Container or on a remote host using SSH.
Build and run
Modify options
Alt+M
java 21 SDK of 'day14' module
-cp day14
com.powernode.array03.Test02
Program arguments
实参书写位置
CLI arguments to your application. Alt+R
Working directory:
D:\powernode\02-JavaSE\03-code\JavaProj
Environment variables:
Environment variables or .env files
Separate variables with semicolon: VAR=value; VAR1=value1
Open run/debug tool window when started
Code Coverage
Modify
Packages and classes to include in coverage data
Run
OK
Cancel
Apply



13. 学生管理系统_v1.0

13.1 添加和查询

代码块

```
1  package com.powernode.array04;
2  class Student{
3      private int sno;
4      private String name;
5      private int age;
6
7      public Student(int sno, String name, int age) {
8          this.sno = sno;
9          this.name = name;
```

```
10          this.age = age;
11      }
12
13      public int getSno() {
14          return sno;
15      }
16
17      public void setSno(int sno) {
18          this.sno = sno;
19      }
20
21      public String getName() {
22          return name;
23      }
24
25      public void setName(String name) {
26          this.name = name;
27      }
28
29      public int getAge() {
30          return age;
31      }
32
33      public void setAge(int age) {
34          this.age = age;
35      }
36
37      @Override
38      public String toString() {
39          return "Student{" +
40                  "sno=" + sno +
41                  ", name='" + name + '\'' +
42                  ", age=" + age +
43                  '}';
44      }
45  }
46  /**
47   * 1.学生管理系统（基于数组），管理学生
48   *      1.添加学生
49   *      2.查询学生
50   *      3.修改学生
51   *      4.删除学生
52   * 2.设计几个类:
53   *      1.Student
54   *      2.StudentManager
55   *      3.Test测试类
56   */
```

```
class StudentManager{
    //1.定义容器，用来存储学生对象
    private Student[] stus = new Student[2];;
    /*public StudentManager(){
        stus = new Student[2];
    }*/
    //2.添加学员
    public void insert(Student student){
        stus[0] = student;
    }
    //3.查询学员
    public void queryAll(){
        for (int i = 0; i < stus.length; i++) {
            Student stu = stus[i];
            System.out.println(stu);
        }
    }
}
public class Test {
    public static void main(String[] args) {
        StudentManager studentManager = new StudentManager();
        studentManager.insert(new Student(1001,"zs",23));
        studentManager.queryAll();
    }
}
```

13.2 插入多条记录

- 存储多条数据
- 查询时存储几个，输出几个，null不输出

代码块

```
package com.powernode.array05;
class Student{
    private int sno;
    private String name;
    private int age;

    public Student(int sno, String name, int age) {
        this.sno = sno;
        this.name = name;
        this.age = age;
    }

    public int getSno() {
```

```
14          return sno;
15      }
16
17      public void setSno(int sno) {
18          this.sno = sno;
19      }
20
21      public String getName() {
22          return name;
23      }
24
25      public void setName(String name) {
26          this.name = name;
27      }
28
29      public int getAge() {
30          return age;
31      }
32
33      public void setAge(int age) {
34          this.age = age;
35      }
36
37      @Override
38      public String toString() {
39          return "Student{" +
40                  "sno=" + sno +
41                  ", name='" + name + '\'' +
42                  ", age=" + age +
43                  '}';
44      }
45  }
46  /**
47   * 1.学生管理系统（基于数组），管理学生
48   *      1.添加学生
49   *      2.查询学生
50   *      3.修改学生
51   *      4.删除学生
52   * 2.设计几个类:
53   *      1.Student
54   *      2.StudentManager
55   *      3.Test测试类
56   */
57  class StudentManager{
58      //1.定义容器，用来存储学生对象
59      private Student[] stus = new Student[2];;
60
```

```
61      //2.添加学员
62      //定义一个变量：count
63      private int count;
64      public void insert(Student student){
65          stus[count ++] = student;
66      }
67      //3.查询学员
68      public void queryAll(){
69          for (int i = 0; i < count; i++) {//数组中存储几个元素，就输出几个
70              Student stu = stus[i];
71              System.out.println(stu);
72          }
73      }
74  }
75  public class Test {
76      public static void main(String[] args) {
77          StudentManager studentManager = new StudentManager();
78          studentManager.insert(new Student(1001,"zs",23));
79          studentManager.insert(new Student(1002,"ls",20));
80          studentManager.queryAll();
81      }
82  }
```

13.3 数组扩容

代码块

```
1   package com.powernode.array06;
2
3   import java.util.Arrays;
4
5   class Student{
6       private int sno;
7       private String name;
8       private int age;
9
10      public Student(int sno, String name, int age) {
11          this.sno = sno;
12          this.name = name;
13          this.age = age;
14      }
15
16      public int getSno() {
17          return sno;
18      }
19
```

```
20      public void setSno(int sno) {
21          this.sno = sno;
22      }
23
24      public String getName() {
25          return name;
26      }
27
28      public void setName(String name) {
29          this.name = name;
30      }
31
32      public int getAge() {
33          return age;
34      }
35
36      public void setAge(int age) {
37          this.age = age;
38      }
39
40      @Override
41      public String toString() {
42          return "Student{" +
43                  "sno=" + sno +
44                  ", name='" + name + '\'' +
45                  ", age=" + age +
46                  '}';
47      }
48  }
49  /**
50   * 1.学生管理系统（基于数组），管理学生
51   *      1.添加学生
52   *      2.查询学生
53   *      3.修改学生
54   *      4.删除学生
55   * 2.设计几个类:
56   *      1.Student
57   *      2.StudentManager
58   *      3.Test测试类
59   */
60  class StudentManager{
61      //1.定义容器，用来存储学生对象
62      private Student[] stus = new Student[2];;
63
64      //2.添加学员
65      //定义一个变量：count
66      private int count;
```

```
67      public void insert(Student student){
68          /**
69           * 1.如果容量不够了，就需要扩容
70           * 2.在插入之前进行判断容器容量
71           *    1.目前有多少个元素：count
72           *    2.容器的容量：stus.length
73           *    3.count >= stus.length : 进行扩容
74           */
75          if (count >= stus.length) {
76              /**
77               * T[] original:需要复制的数组
78               * int newLeng :  新数组的长度，扩容后的长度
79               */
80              stus = Arrays.copyOf(stus, stus.length << 1);
81          }
82          stus[count ++] = student;
83      }
84      //3.查询学员
85      public void queryAll(){
86          for (int i = 0; i < count; i++) {//数组中存储几个元素，就输出几个
87              Student stu = stus[i];
88              System.out.println(stu);
89          }
90      }
91  }
92  public class Test {
93      public static void main(String[] args) {
94          StudentManager studentManager = new StudentManager();
95          studentManager.insert(new Student(1001,"zs",23));
96          studentManager.insert(new Student(1002,"ls",20));
97          studentManager.insert(new Student(1003,"ww",26));
98          studentManager.queryAll();
99      }
100 }
```

13.4 学生是否存在

代码块

```
1   package com.powernode.array07;
2
3   import java.util.Arrays;
4
5   class Student{
6       private int sno;
7       private String name;
```

```
    private int age;

    public Student(int sno, String name, int age) {
        this.sno = sno;
        this.name = name;
        this.age = age;
    }

    public int getSno() {
        return sno;
    }

    public void setSno(int sno) {
        this.sno = sno;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student{" +
                "sno=" + sno +
                ", name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
}
/**
 * 1.学生管理系统（基于数组），管理学生
 *     1.添加学生
 *     2.查询学生
 *     3.修改学生
 *     4.删除学生
```

```
 * 2.设计几个类:
 *     1.Student
 *     2.StudentManager
 *     3.Test测试类
 */
class StudentManager{
    //1.定义容器，用来存储学生对象
    private Student[] stus = new Student[2];;

    //2.添加学员
    //定义一个变量: count
    private int count;
    public void insert(Student student){

        //在正式插入之前，判断一下学生是否（sno）存在
        boolean flag = isExists(student.getSno());
        if (flag) {
            System.out.println("需要添加的学生学号: 【" + student.getSno() + "】已经存在");
            return;//结束方法
        }
        /**
         * 1.如果容量不够了，就需要扩容
         * 2.在插入之前进行判断容器容量
         *    1.目前有多少个元素: count
         *    2.容器的容量: stus.length
         *    3.count >= stus.length : 进行扩容
         */
        if (count >= stus.length) {
            /**
             * T[] original:需要复制的数组
             * int newLeng :  新数组的长度，扩容后的长度
             */
            stus = Arrays.copyOf(stus, stus.length << 1);
        }
        stus[count ++] = student;
    }

    private boolean isExists(int sno) {
        /**
         * 1.找到数组容器
         * 2.把数组容器中的学生学号一个个取出来 和 传入的sno进行匹配
         * 3.如果等值匹配成功返回true，说明容器中存在
         * 4.否则不存在
         */
        for (int i = 0; i < count; i++) {
            Student stu = stus[i];
```

```
            if (stu.getSno() == sno) return true;
        }
        return false;
    }

    //3.查询学员
    public void queryAll(){
        for (int i = 0; i < count; i++) {//数组中存储几个元素，就输出几个
            Student stu = stus[i];
            System.out.println(stu);
        }
    }
}
public class Test {
    public static void main(String[] args) {
        StudentManager studentManager = new StudentManager();
        studentManager.insert(new Student(1001,"zs",23));
        studentManager.insert(new Student(1002,"ls",20));
        studentManager.insert(new Student(1001,"zs",23));
        studentManager.queryAll();
    }
}
```

13.5 修改学员

代码块

```
package com.powernode.array08;

import java.util.Arrays;

class Student{
    private int sno;
    private String name;
    private int age;

    public Student(int sno, String name, int age) {
        this.sno = sno;
        this.name = name;
        this.age = age;
    }

    public int getSno() {
        return sno;
    }

```

```
20      public void setSno(int sno) {
21          this.sno = sno;
22      }
23
24      public String getName() {
25          return name;
26      }
27
28      public void setName(String name) {
29          this.name = name;
30      }
31
32      public int getAge() {
33          return age;
34      }
35
36      public void setAge(int age) {
37          this.age = age;
38      }
39
40      @Override
41      public String toString() {
42          return "Student{" +
43                  "sno=" + sno +
44                  ", name='" + name + '\'' +
45                  ", age=" + age +
46                  '}';
47      }
48  }
49  /**
50   * 1.学生管理系统（基于数组），管理学生
51   *      1.添加学生
52   *      2.查询学生
53   *      3.修改学生
54   *      4.删除学生
55   * 2.设计几个类：
56   *      1.Student
57   *      2.StudentManager
58   *      3.Test测试类
59   */
60  class StudentManager{
61      //1.定义容器，用来存储学生对象
62      private Student[] stus = new Student[2];;
63
64      private int count;
65      //2.添加学员
66      //定义一个变量：count
```

```
67      public void insert(Student student){
68
69          //在正式插入之前，判断一下学生是否（sno）存在
70          boolean flag = isExists(student.getSno());
71          if (flag) {
72              System.out.println("需要添加的学生学号：【" + student.getSno() + "】已
    经存在");
73              return;//结束方法
74          }
75          /**
76           * 1.如果容量不够了，就需要扩容
77           * 2.在插入之前进行判断容器容量
78           *    1.目前有多少个元素：count
79           *    2.容器的容量：stus.length
80           *    3.count >= stus.length : 进行扩容
81           */
82          if (count >= stus.length) {
83              /**
84               * T[] original:需要复制的数组
85               * int newLeng :  新数组的长度，扩容后的长度
86               */
87              stus = Arrays.copyOf(stus, stus.length << 1);
88          }
89          stus[count ++] = student;
90      }
91      //4.判断学员是否存在
92      private boolean isExists(int sno) {
93          /**
94           * 1.找到数组容器
95           * 2.把数组容器中的学生学号一个个取出来 和 传入的sno进行匹配
96           * 3.如果等值匹配成功返回true，说明容器中存在
97           * 4.否则不存在
98           */
99          for (int i = 0; i < count; i++) {
100             Student stu = stus[i];
101             if (stu.getSno() == sno) return true;
102         }
103         return false;
104     }
105
106     //3.查询学员
107     public void queryAll(){
108         for (int i = 0; i < count; i++) {//数组中存储几个元素，就输出几个
109             Student stu = stus[i];
110             System.out.println(stu);
111         }
112     }
```

```
113       //5.修改学员
114       /**
115        * @param sno : 需要修改学生的编号
116        * @param student: 需要修改的数据，外界传入
117        */
118       public void updateStudentBySNO(int sno,Student student){
119           //1.判断学生是否存在
120           boolean flag = isExists(sno);
121           if (flag) {//存在
122              /* int[] ints = {1, 2, 3};
123               ints[2] = 6;*/
124               //修改数组中的学员（需要获得下标）
125               int updateIndex = getIndexBySNO(sno);
126               //修改学员
127               stus[updateIndex] = student;
128           }
129       }
130       //通过学生sno获得学员下标
131       private int getIndexBySNO(int sno) {
132           for (int i = 0; i < count; i++) {
133               Student stu = stus[i];
134               if (stu.getSno() == sno) {
135                   return i;
136               }
137           }
138           return -1;
139       }
140
141   }
142   public class Test {
143       public static void main(String[] args) {
144           StudentManager studentManager = new StudentManager();
145           studentManager.insert(new Student(1001,"zs",23));
146           studentManager.insert(new Student(1002,"ls",20));
147           studentManager.insert(new Student(1003,"ww",36));
148           studentManager.queryAll();
149           System.out.println("===========修改学员================");
150           studentManager.updateStudentBySNO(1002,new Student(1006,"zl",21));
151           studentManager.queryAll();
152       }
153   }
```

13.6 代码优化

代码块

```java
package com.powernode.array09;

import java.util.Arrays;

class Student{
    private int sno;
    private String name;
    private int age;

    public Student(int sno, String name, int age) {
        this.sno = sno;
        this.name = name;
        this.age = age;
    }

    public int getSno() {
        return sno;
    }

    public void setSno(int sno) {
        this.sno = sno;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student{" +
                "sno=" + sno +
                ", name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
```

```
48  }
49  /**
50   * 1.学生管理系统（基于数组），管理学生
51   *     1.添加学生
52   *     2.查询学生
53   *     3.修改学生
54   *     4.删除学生
55   * 2.设计几个类:
56   *     1.Student
57   *     2.StudentManager
58   *     3.Test测试类
59   */
60  class StudentManager{
61      //1.定义容器，用来存储学生对象
62      private Student[] stus = new Student[2];;
63
64      private int count;
65      //2.添加学员
66      //定义一个变量：count
67      public void insert(Student student){
68
69          //在正式插入之前，判断一下学生是否（sno）存在
70          int index = isExists(student.getSno());
71          if (index != -1) {
72              System.out.println("需要添加的学生学号：【" + student.getSno() + "】已
    经存在");
73              return;//结束方法
74          }
75          /**
76           * 1.如果容量不够了，就需要扩容
77           * 2.在插入之前进行判断容器容量
78           *    1.目前有多少个元素：count
79           *    2.容器的容量：stus.length
80           *    3.count >= stus.length : 进行扩容
81           */
82          if (count >= stus.length) {
83              /**
84               * T[] original:需要复制的数组
85               * int newLeng :  新数组的长度，扩容后的长度
86               */
87              stus = Arrays.copyOf(stus, stus.length << 1);
88          }
89          stus[count ++] = student;
90      }
91      //4.判断学员是否存在
92      private int isExists(int sno) {
93          /**
```

```
94              * 1.找到数组容器
95              * 2.把数组容器中的学生学号一个个取出来 和 传入的sno进行匹配
96              * 3.如果等值匹配成功返回true，说明容器中存在
97              * 4.否则不存在
98              */
99             for (int i = 0; i < count; i++) {
100                Student stu = stus[i];
101                if (stu.getSno() == sno) return i;
102            }
103            return -1;
104        }
105
106        //3.查询学员
107        public void queryAll(){
108            for (int i = 0; i < count; i++) {//数组中存储几个元素，就输出几个
109                Student stu = stus[i];
110                System.out.println(stu);
111            }
112        }
113        //5.修改学员
114        /**
115         * @param sno : 需要修改学生的编号
116         * @param student: 需要修改的数据，外界传入
117         */
118        public void updateStudentBySNO(int sno,Student student){
119            //1.判断学生是否存在
120          /*  boolean flag = isExists(sno);
121            if (flag) {//存在
122               *//* int[] ints = {1, 2, 3};
123                ints[2] = 6;*//*
124                //修改数组中的学员（需要获得下标）
125                int updateIndex = getIndexBySNO(sno);
126                //修改学员
127                stus[updateIndex] = student;
128            }*/
129            int index = isExists(sno);
130            if (index != -1) {//存在
131                //修改学员
132                stus[index] = student;
133            }
134        }
135
136
137    }
138    public class Test {
139        public static void main(String[] args) {
140            StudentManager studentManager = new StudentManager();
```

```
        studentManager.insert(new Student(1001,"zs",23));
        studentManager.insert(new Student(1002,"ls",20));
        studentManager.insert(new Student(1003,"ww",36));
        studentManager.queryAll();
        System.out.println("===========修改学员===============");
        studentManager.updateStudentBySNO(1002,new Student(1006,"zl",21));
        studentManager.queryAll();
    }
}
```

13.7 删除学员

代码块

```
package com.powernode.array10;

import java.util.Arrays;

class Student{
    private int sno;
    private String name;
    private int age;

    public Student(int sno, String name, int age) {
        this.sno = sno;
        this.name = name;
        this.age = age;
    }

    public int getSno() {
        return sno;
    }

    public void setSno(int sno) {
        this.sno = sno;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
```

```
33          return age;
34      }
35
36      public void setAge(int age) {
37          this.age = age;
38      }
39
40      @Override
41      public String toString() {
42          return "Student{" +
43                  "sno=" + sno +
44                  ", name='" + name + '\'' +
45                  ", age=" + age +
46                  '}';
47      }
48  }
49  /**
50   * 1.学生管理系统（基于数组），管理学生
51   *      1.添加学生
52   *      2.查询学生
53   *      3.修改学生
54   *      4.删除学生
55   * 2.设计几个类：
56   *      1.Student
57   *      2.StudentManager
58   *      3.Test测试类
59   */
60  class StudentManager{
61      //1.定义容器，用来存储学生对象
62      private Student[] stus = new Student[2];;
63
64      private int count;
65      //2.添加学员
66      //定义一个变量：count
67      public void insert(Student student){
68
69          //在正式插入之前，判断一下学生是否（sno）存在
70          int index = isExists(student.getSno());
71          if (index != -1) {
72              System.out.println("需要添加的学生学号：【" + student.getSno() + "】已
    经存在");
73              return;//结束方法
74          }
75          /**
76           * 1.如果容量不够了，就需要扩容
77           * 2.在插入之前进行判断容器容量
78           *     1.目前有多少个元素：count
```

```
79              *    2.容器的容量: stus.length
80              *    3.count >= stus.length : 进行扩容
81              */
82             if (count >= stus.length) {
83                 /**
84                  * T[] original:需要复制的数组
85                  * int newLeng :  新数组的长度，扩容后的长度
86                  */
87                 stus = Arrays.copyOf(stus, stus.length << 1);
88             }
89             stus[count ++] = student;
90         }
91         //4.判断学员是否存在
92         private int isExists(int sno) {
93             /**
94              * 1.找到数组容器
95              * 2.把数组容器中的学生学号一个个取出来 和 传入的sno进行匹配
96              * 3.如果等值匹配成功返回true，说明容器中存在
97              * 4.否则不存在
98              */
99             for (int i = 0; i < count; i++) {
100                Student stu = stus[i];
101                if (stu.getSno() == sno) return i;
102            }
103            return -1;
104        }
105
106        //3.查询学员
107        public void queryAll(){
108            for (int i = 0; i < count; i++) {//数组中存储几个元素，就输出几个
109                Student stu = stus[i];
110                System.out.println(stu);
111            }
112        }
113        //5.修改学员
114        /**
115         * @param sno : 需要修改学生的编号
116         * @param student: 需要修改的数据，外界传入
117         */
118        public void updateStudentBySNO(int sno,Student student){
119            //1.判断学生是否存在
120          /*  boolean flag = isExists(sno);
121            if (flag) {//存在
122               *//* int[] ints = {1, 2, 3};
123                ints[2] = 6;*//*
124                //修改数组中的学员（需要获得下标）
125                int updateIndex = getIndexBySNO(sno);
```

```
126                 //修改学员
127                 stus[updateIndex] = student;
128             }*/
129             int index = isExists(sno);
130             if (index != -1) {//存在
131                 //修改学员
132                 stus[index] = student;
133             }
134         }
135         //6.删除学生
136         public void deleteStudentBySNO(int sno){
137             //判断学生是否存在
138             int index = isExists(sno);
139             if (index == -1) {
140                 System.out.println("需要删除的学生学号：【" + sno + "】不存在");
141                 return;//结束方法
142             }
143             //把后面的往前挪动一位(数组：stus,索引：index)
144             System.arraycopy(stus,index + 1,stus,index,stus.length - index - 1);
145             //{1,2,3,4,5,6}, {1,2,3,5,6,6}
146             //stus = Arrays.copyOf(stus, stus.length - 1);//{1,2,3,5,6}
147             count --;
148         }
149 
150     }
151     public class Test {
152         public static void main(String[] args) {
153             StudentManager studentManager = new StudentManager();
154             studentManager.insert(new Student(1001,"zs",23));
155             studentManager.insert(new Student(1002,"ls",20));
156             studentManager.insert(new Student(1003,"ww",36));
157             studentManager.queryAll();
158             System.out.println("===========修改学员================");
159             studentManager.updateStudentBySNO(1002,new Student(1006,"zl",21));
160             studentManager.queryAll();
161             System.out.println("===========删除学员================");
162             studentManager.deleteStudentBySNO(1006);
163             studentManager.queryAll();
164 
165         }
166     }
```

14. 学生管理系统_v2.0

14.1 界面准备

代码块

```
package com.powernode.array11;

import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        /**
         * 1.do-while应用场景（简易版的增删改查的逻辑）
         * 2.界面如下:
         *    请选择如下操作:
         *       1.增加    2.删除    3.修改    4.查询    5.退出
         * 3.选择对应的编号执行相应的操作
         */
        int num = 0;
        do {
            System.out.println("请选择如下操作: ");
            System.out.println("1.增加   2.删除   3.修改   4.查询   5.退出");
            System.out.println("请选择（1-5）: ");
            num = new Scanner(System.in).nextInt();
            switch (num) {
                case 1 -> System.out.println("----------执行增加操作----------
");
                case 2 -> System.out.println("----------执行删除操作----------
");
                case 3 -> System.out.println("----------执行修改操作----------
");
                case 4 -> System.out.println("----------执行查询操作----------
");
                case 5 -> System.out.println("----------执行退出操作----------
");
                default -> System.out.println("----------无效操作----------");
            }
        }while(num != 5);
    }
}
```

14.2 融合代码（插入）

代码块

```
package com.powernode.array12;

import java.util.Arrays;
import java.util.Scanner;
```

```java

class Student{
    private int sno;
    private String name;
    private int age;

    public Student(int sno, String name, int age) {
        this.sno = sno;
        this.name = name;
        this.age = age;
    }

    public int getSno() {
        return sno;
    }

    public void setSno(int sno) {
        this.sno = sno;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student{" +
                "sno=" + sno +
                ", name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
}
/**
 * 1.学生管理系统（基于数组），管理学生
```

```
52      *      1.添加学生
53      *      2.查询学生
54      *      3.修改学生
55      *      4.删除学生
56      * 2.设计几个类:
57      *      1.Student
58      *      2.StudentManager
59      *      3.Test测试类
60      */
61  class StudentManager{
62      //1.定义容器，用来存储学生对象
63      private Student[] stus = new Student[2];;
64
65      private int count;
66      //2.添加学员
67      //定义一个变量: count
68      public void insert(Student student){
69
70          //在正式插入之前，判断一下学生是否（sno）存在
71          int index = isExists(student.getSno());
72          if (index != -1) {
73              System.out.println("需要添加的学生学号: 【" + student.getSno() + "】已
    经存在");
74              return;//结束方法
75          }
76          /**
77           * 1.如果容量不够了，就需要扩容
78           * 2.在插入之前进行判断容器容量
79           *    1.目前有多少个元素: count
80           *    2.容器的容量: stus.length
81           *    3.count >= stus.length : 进行扩容
82           */
83          if (count >= stus.length) {
84              /**
85               * T[] original:需要复制的数组
86               * int newLeng :  新数组的长度，扩容后的长度
87               */
88              stus = Arrays.copyOf(stus, stus.length << 1);
89          }
90          stus[count ++] = student;
91      }
92      //4.判断学员是否存在
93      private int isExists(int sno) {
94          /**
95           * 1.找到数组容器
96           * 2.把数组容器中的学生学号一个个取出来 和 传入的sno进行匹配
97           * 3.如果等值匹配成功返回true，说明容器中存在
```

```
98              * 4.否则不存在
99              */
100             for (int i = 0; i < count; i++) {
101                 Student stu = stus[i];
102                 if (stu.getSno() == sno) return i;
103             }
104             return -1;
105         }
106
107         //3.查询学员
108         public void queryAll(){
109             for (int i = 0; i < count; i++) {//数组中存储几个元素，就输出几个
110                 Student stu = stus[i];
111                 System.out.println(stu);
112             }
113         }
114         //5.修改学员
115         /**
116          * @param sno : 需要修改学生的编号
117          * @param student: 需要修改的数据，外界传入
118          */
119         public void updateStudentBySNO(int sno, Student student){
120             //1.判断学生是否存在
121           /*  boolean flag = isExists(sno);
122             if (flag) {//存在
123                *//* int[] ints = {1, 2, 3};
124                 ints[2] = 6;*//*
125                 //修改数组中的学员（需要获得下标）
126                 int updateIndex = getIndexBySNO(sno);
127                 //修改学员
128                 stus[updateIndex] = student;
129             }*/
130             int index = isExists(sno);
131             if (index != -1) {//存在
132                 //修改学员
133                 stus[index] = student;
134             }
135         }
136         //6.删除学生
137         public void deleteStudentBySNO(int sno){
138             //判断学生是否存在
139             int index = isExists(sno);
140             if (index == -1) {
141                 System.out.println("需要删除的学生学号：【" + sno + "】不存在");
142                 return;//结束方法
143             }
144             //把后面的往前挪动一位(数组：stus,索引：index)
```

```
145             System.arraycopy(stus,index + 1,stus,index,stus.length - index - 1);
146             //{1,2,3,4,5,6}, {1,2,3,5,6,6}
147             //stus = Arrays.copyOf(stus, stus.length - 1);//{1,2,3,5,6}
148             count --;
149         }
150
151     }
152     public class Test {
153         public static void main(String[] args) {
154             /**
155              * 1.do-while应用场景（简易版的增删改查的逻辑）
156              * 2.界面如下:
157              *     请选择如下操作:
158              *         1.增加     2.删除     3.修改     4.查询     5.退出
159              * 3.选择对应的编号执行相应的操作
160              */
161             int num = 0;
162             StudentManager studentManager = new StudentManager();
163             do {
164                 System.out.println("请选择如下操作: ");
165                 System.out.println("1.增加    2.删除    3.修改    4.查询    5.退出");
166                 System.out.print("请选择（1-5）: ");
167                 num = new Scanner(System.in).nextInt();
168                 switch (num) {
169                     case 1 -> {
170                         System.out.println("----------执行增加操作----------");
171                         System.out.println("请输入学号: ");
172                         int sno = new Scanner(System.in).nextInt();
173                         System.out.println("请输入姓名: ");
174                         String name = new Scanner(System.in).next();
175                         System.out.println("请输入年龄: ");
176                         int age = new Scanner(System.in).nextInt();
177                         studentManager.insert(new Student(sno,name,age));
178                         studentManager.queryAll();
179                     }
180                     case 2 -> System.out.println("----------执行删除操作----------
    ");
181                     case 3 -> System.out.println("----------执行修改操作----------
    ");
182                     case 4 -> System.out.println("----------执行查询操作----------
    ");
183                     case 5 -> System.out.println("----------执行退出操作----------
    ");
184                     default -> System.out.println("----------无效操作----------");
185                 }
186             }while(num != 5);
187         }
```

```
188  }
```

14.3 融合其他功能

代码块

```
1   package com.powernode.array13;
2
3   import java.util.Arrays;
4   import java.util.Scanner;
5
6   class Student{
7       private int sno;
8       private String name;
9       private int age;
10
11      public Student(int sno, String name, int age) {
12          this.sno = sno;
13          this.name = name;
14          this.age = age;
15      }
16
17      public int getSno() {
18          return sno;
19      }
20
21      public void setSno(int sno) {
22          this.sno = sno;
23      }
24
25      public String getName() {
26          return name;
27      }
28
29      public void setName(String name) {
30          this.name = name;
31      }
32
33      public int getAge() {
34          return age;
35      }
36
37      public void setAge(int age) {
38          this.age = age;
39      }
40
```

```
41      @Override
42      public String toString() {
43          return "Student{" +
44                  "sno=" + sno +
45                  ", name='" + name + '\'' +
46                  ", age=" + age +
47                  '}';
48      }
49  }
50  /**
51   * 1.学生管理系统（基于数组），管理学生
52   *      1.添加学生
53   *      2.查询学生
54   *      3.修改学生
55   *      4.删除学生
56   * 2.设计几个类：
57   *      1.Student
58   *      2.StudentManager
59   *      3.Test测试类
60   */
61  class StudentManager{
62      //1.定义容器，用来存储学生对象
63      private Student[] stus = new Student[2];;
64
65      private int count;
66      //2.添加学员
67      //定义一个变量：count
68      public void insert(Student student){
69
70          //在正式插入之前，判断一下学生是否（sno）存在
71          int index = isExists(student.getSno());
72          if (index != -1) {
73              System.out.println("需要添加的学生学号：【" + student.getSno() + "】已
    经存在");
74              return;//结束方法
75          }
76          /**
77           * 1.如果容量不够了，就需要扩容
78           * 2.在插入之前进行判断容器容量
79           *     1.目前有多少个元素：count
80           *     2.容器的容量：stus.length
81           *     3.count >= stus.length : 进行扩容
82           */
83          if (count >= stus.length) {
84              /**
85               * T[] original:需要复制的数组
86               * int newLeng :  新数组的长度，扩容后的长度
```

```
87                 */
88                 stus = Arrays.copyOf(stus, stus.length << 1);
89             }
90             stus[count ++] = student;
91         }
92         //4.判断学员是否存在
93         private int isExists(int sno) {
94             /**
95              * 1.找到数组容器
96              * 2.把数组容器中的学生学号一个个取出来 和 传入的sno进行匹配
97              * 3.如果等值匹配成功返回true，说明容器中存在
98              * 4.否则不存在
99              */
100            for (int i = 0; i < count; i++) {
101                Student stu = stus[i];
102                if (stu.getSno() == sno) return i;
103            }
104            return -1;
105        }
106
107        //3.查询学员
108        public void queryAll(){
109            for (int i = 0; i < count; i++) {//数组中存储几个元素，就输出几个
110                Student stu = stus[i];
111                System.out.println(stu);
112            }
113        }
114        //5.修改学员
115        /**
116         * @param sno : 需要修改学生的编号
117         * @param student: 需要修改的数据，外界传入
118         */
119        public void updateStudentBySNO(int sno, Student student){
120            //1.判断学生是否存在
121          /*  boolean flag = isExists(sno);
122            if (flag) {//存在
123               *//* int[] ints = {1, 2, 3};
124                ints[2] = 6;*//*
125                //修改数组中的学员（需要获得下标）
126                int updateIndex = getIndexBySNO(sno);
127                //修改学员
128                stus[updateIndex] = student;
129            }*/
130            int index = isExists(sno);
131            if (index != -1) {//存在
132                //修改学员
133                stus[index] = student;
```

```
134             }
135         }
136         //6.删除学生
137         public void deleteStudentBySNO(int sno){
138             //判断学生是否存在
139             int index = isExists(sno);
140             if (index == -1) {
141                 System.out.println("需要删除的学生学号：【" + sno + "】不存在");
142                 return;//结束方法
143             }
144             //把后面的往前挪动一位(数组：stus,索引：index)
145             System.arraycopy(stus,index + 1,stus,index,stus.length - index - 1);
146             //{1,2,3,4,5,6}, {1,2,3,5,6,6}
147             //stus = Arrays.copyOf(stus, stus.length - 1);//{1,2,3,5,6}
148             count --;
149         }
150 
151     }
152     public class Test {
153         public static void main(String[] args) {
154             /**
155              * 1.do-while应用场景（简易版的增删改查的逻辑）
156              * 2.界面如下：
157              *     请选择如下操作：
158              *         1.增加     2.删除     3.修改     4.查询     5.退出
159              * 3.选择对应的编号执行相应的操作
160              */
161             int num = 0;
162             StudentManager studentManager = new StudentManager();
163             do {
164                 System.out.println("请选择如下操作：");
165                 System.out.println("1.增加    2.删除    3.修改    4.查询    5.退出");
166                 System.out.print("请选择（1-5）：");
167                 num = new Scanner(System.in).nextInt();
168                 switch (num) {
169                     case 1 -> {
170                         System.out.println("----------执行增加操作------------");
171                         System.out.println("请输入学号：");
172                         int sno = new Scanner(System.in).nextInt();
173                         System.out.println("请输入姓名：");
174                         String name = new Scanner(System.in).next();
175                         System.out.println("请输入年龄：");
176                         int age = new Scanner(System.in).nextInt();
177                         studentManager.insert(new Student(sno,name,age));
178                         studentManager.queryAll();
179                     }
180                     case 2 -> {
```

```
                    System.out.println("----------执行删除操作------------");
                    System.out.println("请输入学号: ");
                    int sno = new Scanner(System.in).nextInt();
                    studentManager.deleteStudentBySNO(sno);
                    studentManager.queryAll();
                }
                case 3 -> {
                    System.out.println("----------执行修改操作------------");
                    System.out.println("请输入学号: ");
                    int updateSno = new Scanner(System.in).nextInt();

                    System.out.println("请输入学号: ");
                    int sno = new Scanner(System.in).nextInt();
                    System.out.println("请输入姓名: ");
                    String name = new Scanner(System.in).next();
                    System.out.println("请输入年龄: ");
                    int age = new Scanner(System.in).nextInt();
                    studentManager.updateStudentBySNO(updateSno,new
Student(sno,name,age));
                    studentManager.queryAll();
                }
                case 4 -> {
                    System.out.println("----------执行查询操作------------");
                    studentManager.queryAll();
                }
                case 5 -> System.out.println("----------执行退出操作------------
");
                default -> System.out.println("----------无效操作------------");
            }
        }while(num != 5);
    }
}
```

14.4 优化代码

代码块

```
package com.powernode.array14;

import java.util.Arrays;
import java.util.Scanner;

class Student{
    private int sno;
    private String name;
    private int age;
```

```
10
11      public Student(int sno, String name, int age) {
12          this.sno = sno;
13          this.name = name;
14          this.age = age;
15      }
16
17      public int getSno() {
18          return sno;
19      }
20
21      public void setSno(int sno) {
22          this.sno = sno;
23      }
24
25      public String getName() {
26          return name;
27      }
28
29      public void setName(String name) {
30          this.name = name;
31      }
32
33      public int getAge() {
34          return age;
35      }
36
37      public void setAge(int age) {
38          this.age = age;
39      }
40
41      @Override
42      public String toString() {
43          return "Student{" +
44                  "sno=" + sno +
45                  ", name='" + name + '\'' +
46                  ", age=" + age +
47                  '}';
48      }
49  }
50  /**
51   * 1.学生管理系统（基于数组），管理学生
52   *      1.添加学生
53   *      2.查询学生
54   *      3.修改学生
55   *      4.删除学生
56   * 2.设计几个类:
```

```
 *      1.Student
 *      2.StudentManager
 *      3.Test测试类
 */
class StudentManager{
    //1.定义容器，用来存储学生对象
    private Student[] stus = new Student[2];;

    private int count;
    //2.添加学员
    //定义一个变量：count
    public void insert(Student student){

        //在正式插入之前，判断一下学生是否（sno）存在
        int index = isExists(student.getSno());
        if (index != -1) {
            System.out.println("需要添加的学生学号：【" + student.getSno() + "】已经存在");
            return;//结束方法
        }
        /**
         * 1.如果容量不够了，就需要扩容
         * 2.在插入之前进行判断容器容量
         *    1.目前有多少个元素：count
         *    2.容器的容量：stus.length
         *    3.count >= stus.length : 进行扩容
         */
        if (count >= stus.length) {
            /**
             * T[] original:需要复制的数组
             * int newLeng :  新数组的长度，扩容后的长度
             */
            stus = Arrays.copyOf(stus, stus.length << 1);
        }
        stus[count ++] = student;
    }
    //4.判断学员是否存在
    private int isExists(int sno) {
        /**
         * 1.找到数组容器
         * 2.把数组容器中的学生学号一个个取出来 和 传入的sno进行匹配
         * 3.如果等值匹配成功返回true，说明容器中存在
         * 4.否则不存在
         */
        for (int i = 0; i < count; i++) {
            Student stu = stus[i];
            if (stu.getSno() == sno) return i;
```

```
103             }
104             return -1;
105         }
106 
107         //3.查询学员
108         public void queryAll(){
109             for (int i = 0; i < count; i++) {//数组中存储几个元素，就输出几个
110                 Student stu = stus[i];
111                 System.out.println(stu);
112             }
113         }
114         //5.修改学员
115         /**
116          * @param sno : 需要修改学生的编号
117          * @param student: 需要修改的数据，外界传入
118          */
119         public void updateStudentBySNO(int sno, Student student){
120             //1.判断学生是否存在
121           /*  boolean flag = isExists(sno);
122             if (flag) {//存在
123                */*//* int[] ints = {1, 2, 3};
124                 ints[2] = 6;*//*
125                 //修改数组中的学员（需要获得下标）
126                 int updateIndex = getIndexBySNO(sno);
127                 //修改学员
128                 stus[updateIndex] = student;
129             }*/
130             int index = isExists(sno);
131             if (index != -1) {//存在
132                 //修改学员
133                 stus[index] = student;
134             }
135         }
136         //6.删除学生
137         public void deleteStudentBySNO(int sno){
138             //判断学生是否存在
139             int index = isExists(sno);
140             if (index == -1) {
141                 System.out.println("需要删除的学生学号：【" + sno + "】不存在");
142                 return;//结束方法
143             }
144             //把后面的往前挪动一位(数组：stus,索引：index)
145             System.arraycopy(stus,index + 1,stus,index,stus.length - index - 1);
146             //{1,2,3,4,5,6}, {1,2,3,5,6,6}
147             //stus = Arrays.copyOf(stus, stus.length - 1);//{1,2,3,5,6}
148             count --;
149         }
```

```
150
151 }
152 public class Test {
153     public static void main(String[] args) {
154         /**
155          * 1.do-while应用场景（简易版的增删改查的逻辑）
156          * 2.界面如下:
157          *     请选择如下操作:
158          *         1.增加     2.删除     3.修改     4.查询     5.退出
159          * 3.选择对应的编号执行相应的操作
160          */
161         int num = 0;
162         StudentManager studentManager = new StudentManager();
163         do {
164             System.out.println("请选择如下操作: ");
165             System.out.println("1.增加   2.删除   3.修改   4.查询   5.退出");
166             System.out.print("请选择（1-5）: ");
167             num = new Scanner(System.in).nextInt();
168             switch (num) {
169                 case 1 -> {
170                     System.out.println("----------执行增加操作------------");
171                     studentManager.insert(getStudent());
172                     studentManager.queryAll();
173                 }
174                 case 2 -> {
175                     System.out.println("----------执行删除操作------------");
176                     studentManager.deleteStudentBySNO(getSNO());
177                     studentManager.queryAll();
178                 }
179                 case 3 -> {
180                     System.out.println("----------执行修改操作------------");
181                     studentManager.updateStudentBySNO(getSNO(),getStudent());
182                     studentManager.queryAll();
183                 }
184                 case 4 -> {
185                     System.out.println("----------执行查询操作------------");
186                     studentManager.queryAll();
187                 }
188                 case 5 -> System.out.println("----------执行退出操作------------
    ");
189                 default -> System.out.println("----------无效操作------------");
190             }
191         }while(num != 5);
192     }
193
194     public static Student getStudent(){
195         System.out.println("请输入学号: ");
```

```
196             int sno = new Scanner(System.in).nextInt();
197             System.out.println("请输入姓名：");
198             String name = new Scanner(System.in).next();
199             System.out.println("请输入年龄：");
200             int age = new Scanner(System.in).nextInt();
201
202             return new Student(sno, name, age);
203         }
204
205         public static int getSNO(){
206             System.out.println("请输入学号：");
207            return new Scanner(System.in).nextInt();
208         }
209
210     }
```

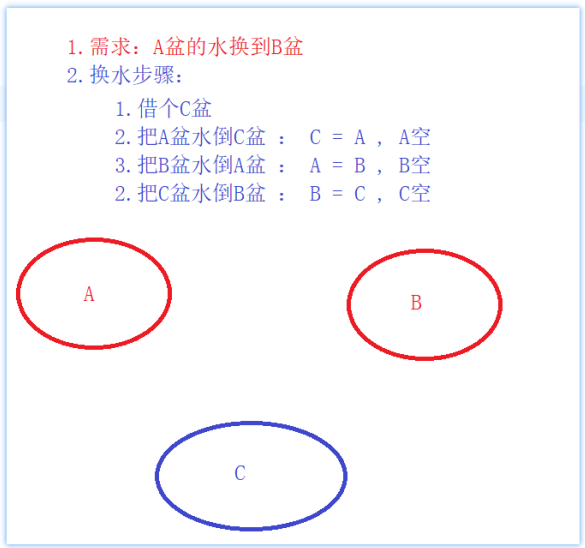
15. 排序

15.1 冒泡排序交换位置

代码块

```
1   package com.powernode.array15;
2
3   import java.util.Arrays;
4
5   public class Test01 {
6       public static void main(String[] args) {
7           //冒泡：大的在后面
8           int[] ints = {6, 1};
9           /**
10           * 如果第一个数 > 第二个数
11           *    交换位置
12           */
13          if (ints[0] > ints[1]) {//A盆：ints[0]，B盆：ints[1]
14              int temp = 0;//C盆
15              temp = ints[0];//把A盆导入C盆，A空
16              ints[0] = ints[1];//把B盆导入A盆，B空
17              ints[1] = temp;//把C盆导入B盆，C空
18          }
19          System.out.println(Arrays.toString(ints));
20      }
21  }
```

```java
public class Test01 {
    public static void main(String[] args) {
        //冒泡：大的在后面
        int[] ints = {6, 1};
        /**
         * 如果第一个数 > 第二个数
         *     交换位置
         */
        if (ints[0] > ints[1]) {//A盆：ints[0]，B盆：ints[1]
            int temp = 0;//C盆
            temp = ints[0];//把A盆导入C盆，A空
            ints[0] = ints[1];//把B盆导入A盆，B空
            ints[1] = temp;//把C盆导入B盆，C空
        }
        System.out.println(Arrays.toString(ints));
    }
}
```



15.2 冒泡实现的原理

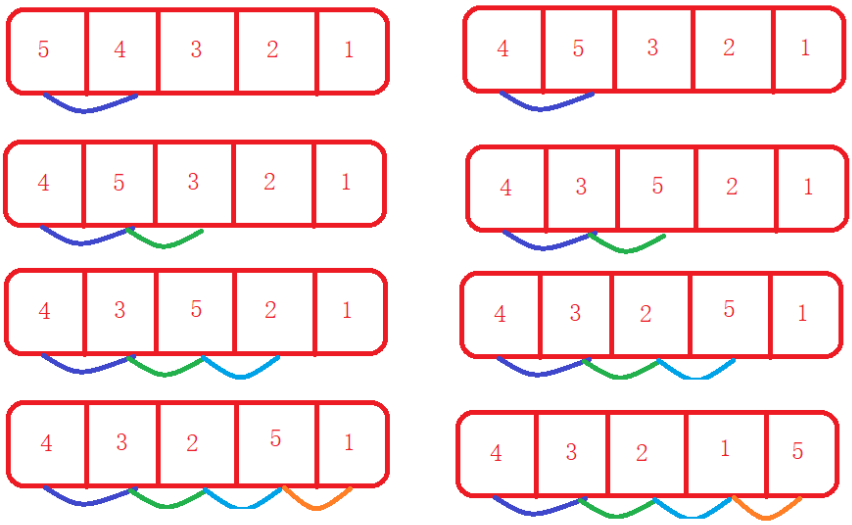
代码块

```java
package com.powernode.array15;

import java.util.Arrays;

public class Test02 {
    public static void main(String[] args) {
        //冒泡：大的在后面
        int[] ints = {5,4,3,2,1};
        for (int i = 0; i < 4; i++) {
            if (ints[i] > ints[i + 1]) {
                int temp = 0;
                temp = ints[i];
                ints[i] = ints[i + 1];
                ints[i + 1] = temp;
            }
        }
        //[4, 3, 2, 1, 5],下一轮5不参与
        for (int i = 0; i < 3; i++) {
            if (ints[i] > ints[i + 1]) {
                int temp = 0;
                temp = ints[i];
                ints[i] = ints[i + 1];
                ints[i + 1] = temp;
            }
        }
        //[3, 2, 1, 4, 5] ,下一轮4不参与
```

```
27            for (int i = 0; i < 2; i++) {
28                if (ints[i] > ints[i + 1]) {
29                    int temp = 0;
30                    temp = ints[i];
31                    ints[i] = ints[i + 1];
32                    ints[i + 1] = temp;
33                }
34            }
35            //[2, 1, 3, 4, 5],下一轮3不参与
36            for (int i = 0; i < 1; i++) {
37                if (ints[i] > ints[i + 1]) {
38                    int temp = 0;
39                    temp = ints[i];
40                    ints[i] = ints[i + 1];
41                    ints[i + 1] = temp;
42                }
43            }
44            //[1, 2, 3, 4, 5]
45            System.out.println(Arrays.toString(ints));
46        }
47    }
```

1. 排序的原理
   1. 相邻两个元素进行比较
   2. 如果前一个 > 后一个，就交换位置
   3. 比如一轮后，最后一个就是最大的

2. 比较次数
   1. 如果有5个元素，比较4次
   2. 如果有4个元素，比较3次



15.3 冒泡排序代码优化（升序）

代码块

```
1   package com.powernode.array15;
2
3   import java.util.Arrays;
4
5   public class Test03 {
6       public static void main(String[] args) {
```

```java
        //冒泡：大的在后面
        int[] ints = {5,4,3,2,1};
        for (int j = ints.length - 1; j > 0 ; j--) {
            for (int i = 0; i < j; i++) {
                if (ints[i] > ints[i + 1]) {
                    int temp = 0;
                    temp = ints[i];
                    ints[i] = ints[i + 1];
                    ints[i + 1] = temp;
                }
            }
        }
        /*for (int i = 0; i < 4; i++) {
            if (ints[i] > ints[i + 1]) {
                int temp = 0;
                temp = ints[i];
                ints[i] = ints[i + 1];
                ints[i + 1] = temp;
            }
        }
        //[4, 3, 2, 1, 5],下一轮5不参与
        for (int i = 0; i < 3; i++) {
            if (ints[i] > ints[i + 1]) {
                int temp = 0;
                temp = ints[i];
                ints[i] = ints[i + 1];
                ints[i + 1] = temp;
            }
        }
        //[3, 2, 1, 4, 5] ,下一轮4不参与
        for (int i = 0; i < 2; i++) {
            if (ints[i] > ints[i + 1]) {
                int temp = 0;
                temp = ints[i];
                ints[i] = ints[i + 1];
                ints[i + 1] = temp;
            }
        }
        //[2, 1, 3, 4, 5],下一轮3不参与
        for (int i = 0; i < 1; i++) {
            if (ints[i] > ints[i + 1]) {
                int temp = 0;
                temp = ints[i];
                ints[i] = ints[i + 1];
                ints[i + 1] = temp;
            }
        }*/
```

```
54          //[1, 2, 3, 4, 5]
55          System.out.println(Arrays.toString(ints));
56      }
57  }
```

15.4 冒泡排序代码优化（降序）

代码块

```
1   package com.powernode.array15;
2
3   import java.util.Arrays;
4
5   public class Test04 {
6       public static void main(String[] args) {
7           int[] ints = {1,2,3,4,5};
8           /**
9            * 1.相邻两个交换位置
10           * 2.如前一个 < 后一个 就交换位置
11           *    第一次：{1,2,3,4,5} ：  {2,1,3,4,5}
12           *    第二次：{2,1,3,4,5} ：  {2,3,1,4,5}
13           *    第三次：{2,3,1,4,5} ：  {2,3,4,1,5}
14           *    第四次：{2,3,4,1,5} ：  {2,3,4,5,1}
15           */
16          for (int j = ints.length - 1; j > 0 ; j--) {
17              for (int i = 0; i < j; i++) {
18                  if (ints[i] < ints[i + 1]) {
19                      int temp = 0;
20                      temp = ints[i];
21                      ints[i] = ints[i + 1];
22                      ints[i + 1] = temp;
23                  }
24              }
25          }
26
27          System.out.println(Arrays.toString(ints));
28      }
29  }
```

作业

1. 把学生管理系统敲一遍（学生管理系统_v1.0）
2. 冒泡排序敲一遍

16. 选择排序

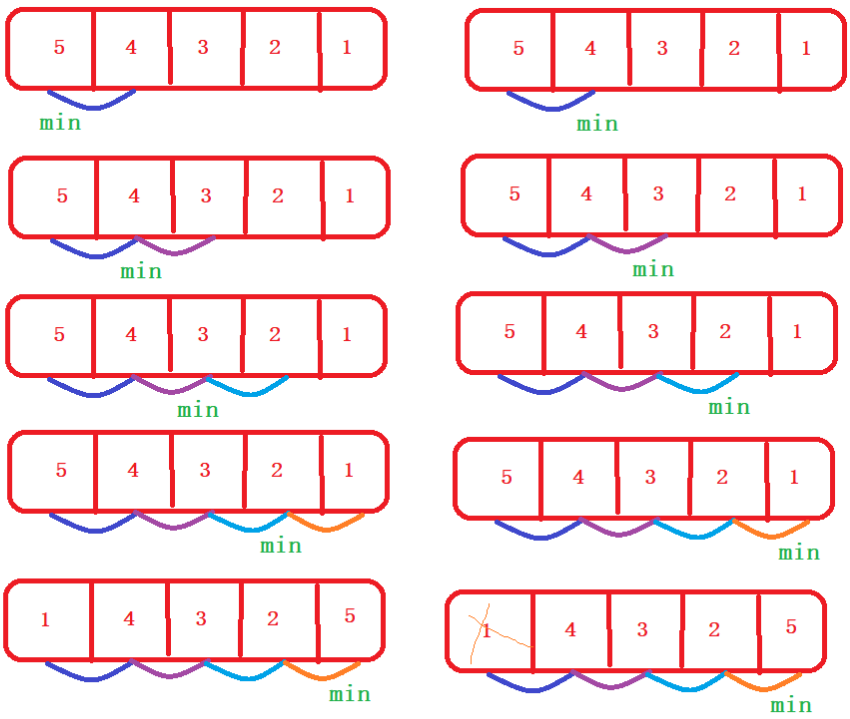
16.1 选择排序的实现原理

选择排序

1. 从待排序数组中找到最小的元素下标
    1. 假设第一个就是最小的（标记为min）
    2. 拿到最小的与之相邻的进行比较
    3. 如果最小的 > 相邻的
    4. 相邻的就是最小的（标记为min）

2. 把最小的与第一个交换位置

有5个元素，比较4次，交换一次



冒泡排序

1. 排序的原理
    1. 相邻两个元素进行比较
    2. 如果前一个 > 后一个 ，就交换位置
    3. 比较一轮后，最后一个就是最大的
2. 比较次数
    1. 如果有5个元素，比较4次
    2. 如果有4个元素，比较3次

16.2 选择排序的代码实现

- 为什么不用min+1，而使用i+1

代码块

```
1  package com.powernode.array01;
2
3  public class Test {
4      public static void main(String[] args) {
5          int[] ints = {3, 1, 2, 0};
6          //1.假设第一个就是最小的
7          int min = 0;
8          for (int i = 0; i < 3; i++) {
9              //2.最小的如果大于相邻的
10             if (ints[min] > ints[min + 1]) {
11                 //3.相邻的就是最小的
12                 min = i + 1;
13             }
14         }
15         System.out.println(min);
16     }
17 }
```

代码块

```
package com.powernode.array01;

import java.util.Arrays;

public class Test {
    public static void main(String[] args) {
        int[] ints = {5, 4, 0, 2, 1};
        //1.假设第一个就是最小的
        int min = 0;
        for (int i = 0; i < 4; i++) {
            //2.最小的如果大于相邻的
            if (ints[min] > ints[i + 1]) {
                //3.相邻的就是最小的
                min = i + 1;
            }
        }
        //假设条件不成立，才交换位置
        if (min != 0) {
            //4.把最小的与第一个交换位置
            int temp = ints[0];
            ints[0] = ints[min];
            ints[min] = temp;
        }
        //[1, 4, 3, 2, 5]
        min = 1;//假设第二个就是最小的
        for (int i = 1; i < 4; i++) {
            if (ints[min] > ints[i+1]) {
                min = i +1 ;
            }
        }
        //假设条件不成立，才交换位置
        if (min != 1) {
            //4.把最小的与第2个交换位置
            int temp = ints[1];
            ints[1] = ints[min];
            ints[min] = temp;
        }


        min = 2;
        for (int i = 2; i < 4; i++) {
            if (ints[min] > ints[i+1]) {
                min = i +1 ;
            }
        }
```

```
        if (min != 2) {
            int temp = ints[2];
            ints[2] = ints[min];
            ints[min] = temp;
        }

        min = 3;
        for (int i = 3; i < 4; i++) {
            if (ints[min] > ints[i+1]) {
                min = i +1 ;
            }
        }
        if (min != 3) {
            int temp = ints[3];
            ints[3] = ints[min];
            ints[min] = temp;
        }
        System.out.println(Arrays.toString(ints));


    }
}
```

16.3 选择排序的代码优化（升序）

代码块

```
package com.powernode.array01;

import java.util.Arrays;

public class Test02 {
    public static void main(String[] args) {
        int[] ints = {5, 4, 0, 2, 1};
        for (int j = 0; j < 4; j++) {
            //1.假设第一个就是最小的
            int min = j;
            for (int i = j; i < 4; i++) {
                //2.最小的如果大于相邻的
                if (ints[min] > ints[i + 1]) {
                    //3.相邻的就是最小的
                    min = i + 1;
                }
            }
            //假设条件不成立，才交换位置
            if (min != j) {
```

```
                //4.把最小的与第一个交换位置
                int temp = ints[j];
                ints[j] = ints[min];
                ints[min] = temp;
            }
        }
        System.out.println(Arrays.toString(ints));


    }
}
```

16.4 选择排序的代码优化（降序）

代码块

```
package com.powernode.array01;

import java.util.Arrays;

public class Test03 {
    public static void main(String[] args) {
        int[] ints = {1,2,3,4,5};
        for (int j = 0; j < 4; j++) {
            //1.假设第一个就是最大的（max）
            int max = j;
            for (int i = j; i < 4; i++) {
                //2.最大的如果小于相邻的
                /**
                 * 原理：{1,2,3,4,5}
                 *   1.从待排序数组中找到最大元素下标
                 *      1.假设第一个就是最大的（max）
                 *      2.拿到最大的与之相邻的进行比较
                 *      3.最大的 < 相邻的
                 *      4.那么相邻的就是最大的
                 *   2.把最大的与第一个交换位置
                 */
                if (ints[max] < ints[i + 1]) {
                    //3.相邻的就是最大的
                    max = i + 1;
                }
            }
            //假设条件不成立，才交换位置
            if (max != j) {
                //4.把最大的与第一个交换位置
                int temp = ints[j];
```

```
                    ints[j] = ints[max];
                    ints[max] = temp;
                }
            }
            System.out.println(Arrays.toString(ints));


        }
    }
```

17. 二分查找

17.1 二分查找的原理

总结：

1.需要查找的数设为x，中间值设为y

2.如果 x == y :找到了

3.如果 x > y :要查找的数在右半部分

1.修改start=mid+1

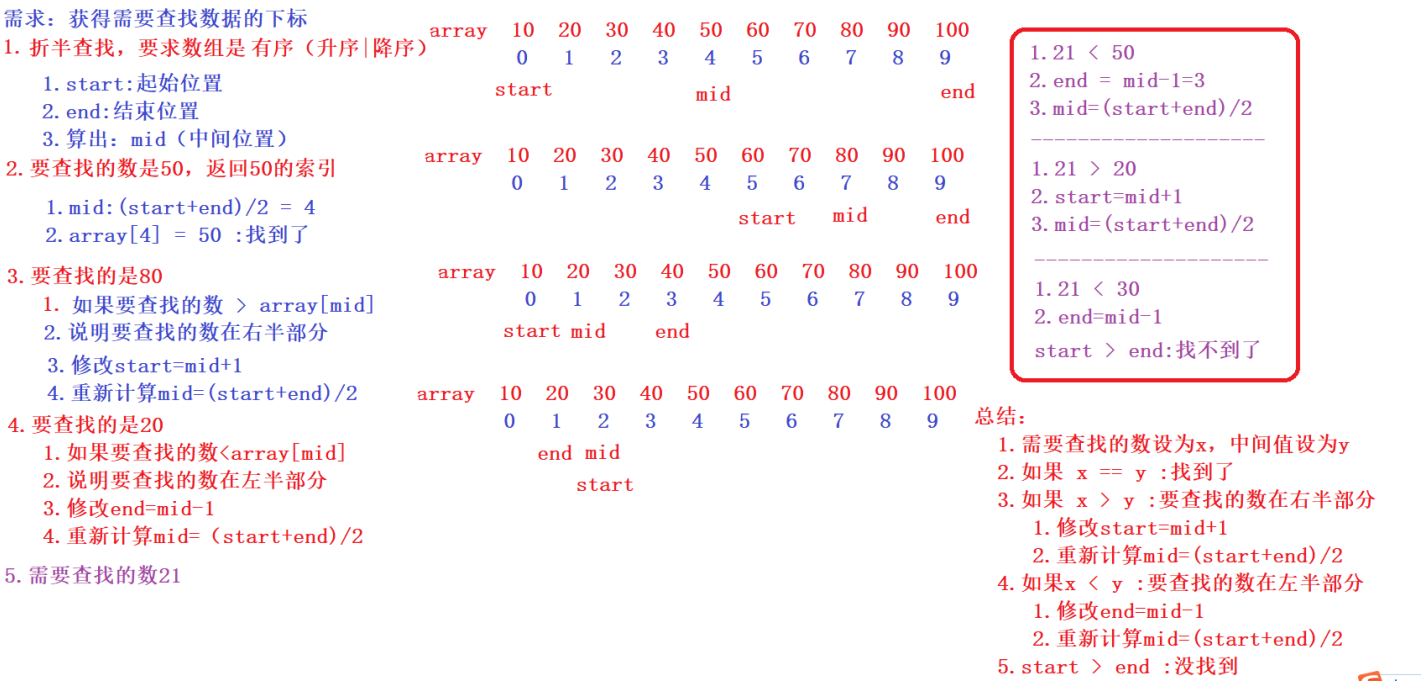
2.重新计算mid=(start+end)/2

4.如果x < y :要查找的数在左半部分

1.修改end=mid-1

2.重新计算mid=(start+end)/2

5.start > end :没找到



17.2 二分查找的循环写法

代码块

```
package com.powernode.array02;


public class Test {
    public static void main(String[] args) {
        /**
         * 总结:
         *   1.需要查找的数设为x，中间值设为y
         *   2.如果 x == y :找到了
         *   3.如果 x > y :要查找的数在右半部分
         *      1.修改start=mid+1
         *      2.重新计算mid=(start+end)/2
         *   4.如果x < y :要查找的数在左半部分
         *      1.修改end=mid-1
         *      2.重新计算mid=(start+end)/2
         *   5.start > end :没找到
         */
        int[] ints = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
        //1.定义起始位置，和结束位置
        int start = 0;
        int end = ints.length - 1;
        //2.需要查找的数
        int num = 50;
        int index = binarySearch(ints, start, end, num);
        System.out.println("index = " + index);
    }

    /**
     * @param ints : 需要查找的数组
     * @param start: 数组的起始位置
     * @param end: 数组的结束位置
     * @param num: 需要查找的数
     * @return: 返回下标
     */
    public static int binarySearch(int[] ints, int start, int end, int num) {
        int mid = (start + end) / 2;
        while(start <= end){//start > end :没找打，所以start <= end 一直找
            if (num == ints[mid]) {//如果 x == y :找到了
                return mid;//返回索引
            }else if (num > ints[mid]){ // 如果x > y ,右半部分
                //1.修改start=mid+1
                start = mid + 1;
                //2.重新计算mid=(start+end)/2
```

```
                mid = (start + end) / 2;
            }else{
                //1.修改end=mid-1
                end = mid - 1;
                //2.重新计算mid=(start+end)/2
                mid = (start + end) / 2;
            }
        }
        return -1;
    }
}
```

17.3 二分查找的递归写法

代码块

```
package com.powernode.array02;


public class Test02 {
    public static void main(String[] args) {
        /**
         * 总结:
         *   1.需要查找的数设为x，中间值设为y
         *   2.如果 x == y :找到了
         *   3.如果 x > y :要查找的数在右半部分
         *      1.修改start=mid+1
         *      2.重新计算mid=(start+end)/2
         *   4.如果x < y :要查找的数在左半部分
         *      1.修改end=mid-1
         *      2.重新计算mid=(start+end)/2
         *   5.start > end :没找到
         */
        int[] ints = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
        //1.定义起始位置，和结束位置
        int start = 0;
        int end = ints.length - 1;
        //2.需要查找的数
        int num = 50;
        int index = binarySearch(ints, start, end, num);
        System.out.println("index = " + index);
    }

    /**
     * @param ints : 需要查找的数组
     * @param start: 数组的起始位置
```

```
31          * @param end: 数组的结束位置
32          * @param num: 需要查找的数
33          * @return: 返回下标
34          */
35         public static int binarySearch(int[] ints, int start, int end, int num) {
36             if(start > end) return -1;
37             int mid = (start + end) / 2;
38             //while(start <= end){//start > end :没找打，所以start <= end 一直找
39                 if (num == ints[mid]) {//如果 x == y :找到了
40                     return mid;//返回索引
41                 }else if (num > ints[mid]){ // 如果x > y ,右半部分
42                     //1.修改start=mid+1
43                     start = mid + 1;
44                     //2.重新计算mid=(start+end)/2
45                     //mid = (start + end) / 2;
46                     return binarySearch(ints, start, end, num);
47                 }else{
48                     //1.修改end=mid-1
49                     end = mid - 1;
50                     //2.重新计算mid=(start+end)/2
51                     //mid = (start + end) / 2;
52                     return binarySearch(ints, start, end, num);
53                 }
54             // }
55             //return -1;
56         }
57     }
```

18. Arrays工具类的常用方法

代码块

```
1   package com.powernode.array03;
2
3   import java.util.Arrays;
4
5   public class Test01 {
6       public static void main(String[] args) {
7           int[] ints = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
8           int index = Arrays.binarySearch(ints, 0, 9, 50);
9           /**
10           * public static String toString(int[] a)   将数组转换为字符串拼接
11           * public static int[] copyOf(int[] original, int newLength)    将数组扩容（original : 第一个参数需要扩容的数组，length: 第二个参数，复制的长度）
12           * public static int[] copyOfRange(int[] original, int from, int to)   将数组扩容到某个范围（original: 第一个参数为要拷贝的数组对象，from: 第二个参数为拷贝的开
```

```
始位置（包含）to：第三个参数为拷贝的结束位置（不包含））
13          * public static void sort(int[] a)      数组升序排序
14          * public static int binarySearch(int[] a, int key) 数组的查找（ a 要搜索的数组， key要查找的值）
15          * public static void fill(int[] a, int val)    想数组填充数据（a : 被填充的数组 , val：填充的数据）
16          */
17         System.out.println(Arrays.toString(ints));
18         ints =new int[] {1,2,3,0,5};//静态初始化，为数组重新赋值
19         Arrays.sort(ints);//数组升序排序 : [0, 1, 2, 3, 5]
20         Arrays.fill(ints,8);//[8, 8, 8, 8, 8]
21
22         ints = new int[]{5, 4, 0, 2, 1};
23         //int[] newInts = Arrays.copyOf(ints, ints.length << 1);
24         int[] newInts = Arrays.copyOf(ints, ints.length >> 1);//[5, 4]
25
26         int[] ints1 = Arrays.copyOfRange(ints, 1, 3);//[1,3):4,0
27         System.out.println(Arrays.toString(ints1));
28
29     }
30 }
```

19. 引用类型的排序

19.1 自定义对象排序（单一字段排序）

代码块

```
1  package com.powernode.array04;
2
3  import java.util.Arrays;
4  import java.util.concurrent.CopyOnWriteArraySet;
5
6  class Teacher implements Comparable{
7      private String name;
8      private int age;
9
10     public Teacher(String name, int age) {
11         this.name = name;
12         this.age = age;
13     }
14
15     public String getName() {
16         return name;
17     }
```

```
18
19      public void setName(String name) {
20          this.name = name;
21      }
22
23      public int getAge() {
24          return age;
25      }
26
27      public void setAge(int age) {
28          this.age = age;
29      }
30
31      @Override
32      public String toString() {
33          return "Teacher{" +
34                  "name='" + name + '\'' +
35                  ", age=" + age +
36                  '}';
37      }
38
39      /**
40       * 1.升序的规则
41       *    1.第一个数  > 第二个数：正整数
42       *    2.第一个数  < 第二个数：负整数
43       *    3.第一个数  = 第二个数：零
44       * 2.按照年龄排序
45       * 3.涉及2个对象
46       *    1.this
47       *    2.o
48       */
49      @Override
50      public int compareTo(Object o) {
51          Teacher teacher = (Teacher) o;
52          if (this.age > teacher.age) {
53              return 1;
54          } else if (this.age < teacher.age) {
55              return -1;
56          }
57          return 0;
58      }
59  }
60
61  public class Test {
62      public static void main(String[] args) {
63          Teacher t1 = new Teacher("zs",23);
64          Teacher t2 = new Teacher("ls",20);
```

```
65          Teacher t3 = new Teacher("ww",26);
66
67          Teacher[] teachers = {t1, t2,t3};
68          /**
69           * java.lang.ClassCastException : 类型转换异常
70           *   1.Teacher cannot be cast to class Comparable : 自定义对象不能转换为
    Comparable
71           *   2.程序在运行的过程中，Teacher 自动转换为  Comparable
72           *   3.什么情况下Teacher可以转换Comparable 【Teacher 实现 Comparable】
73           *   4.自定义对象排序
74           *      1.实现 Comparable
75           *      2.重写compareTo
76           *
77           */
78          Arrays.sort(teachers);
79          System.out.println(Arrays.toString(teachers));
80
81      }
82  }
```

19.2 总结编写步骤

1. 自定义类实现Comparable
2. 重写*compareTo*
3. 编写比较规则

代码块

```
1   1.升序的规则
2       1.第一个数  > 第二个数：正整数
3       2.第一个数  < 第二个数：负整数
4       3.第一个数  = 第二个数：零
5   2.降序的规则
6       1.第一个数  > 第二个数：负整数
7       2.第一个数  < 第二个数：正整数
8       3.第一个数  = 第二个数：零
```

19.3 自定义对象排序（多字段排序）

代码块

```
1   package com.powernode.array05;
2
3   import java.util.Arrays;
```

```
4
5   class Teacher implements Comparable{
6       private String name;
7       private int age;
8
9       public Teacher(String name, int age) {
10          this.name = name;
11          this.age = age;
12      }
13
14      public String getName() {
15          return name;
16      }
17
18      public void setName(String name) {
19          this.name = name;
20      }
21
22      public int getAge() {
23          return age;
24      }
25
26      public void setAge(int age) {
27          this.age = age;
28      }
29
30      @Override
31      public String toString() {
32          return "Teacher{" +
33                  "name='" + name + '\'' +
34                  ", age=" + age +
35                  '}';
36      }
37      //首先按照年龄进行排序，年龄排不开了，按照姓名
38      @Override
39      public int compareTo(Object o) {
40          Teacher teacher = (Teacher) o;
41          if (age > teacher.age) {
42              return 1;
43          } else if (age < teacher.age) {
44              return -1;
45          }
46          return name.compareTo(teacher.name);
47      }
48  }
49  public class Test {
50      public static void main(String[] args) {
```

```
        Teacher t1 = new Teacher("aa", 23);
        Teacher t2 = new Teacher("cc", 20);
        Teacher t3 = new Teacher("bb", 20);
        Teacher t4 = new Teacher("dd", 26);

        Teacher[] teachers = {t1, t2, t3,t4};
        Arrays.sort(teachers);
        System.out.println(Arrays.toString(teachers));
    }
}
```

19.4 改变String的比较规则

代码块

```
package com.powernode.array06;

import java.util.Arrays;
import java.util.Comparator;

public class Test {
    public static void main(String[] args) {
        String[] strings = {"a", "e", "c", "b", "d", "f"};
        Arrays.sort(strings);//[a, b, c, d, e, f]
        System.out.println(Arrays.toString(strings));
        /**
         * 现在需要降序
         *    1.String类实现了Comparable，String类具有了比较性
         *    2.String类的默认比较规则是【升序】
         *    3.我们需要降序，改变String类的比较规则，就需要用到Comparator
         *    4.sort的两个参数：
         *       1.T[] a ：  可以把T看出Object,需要排序的数组
         *       2.Comparator<? super T> c ：指定新的排序规则
         *
         */
        Arrays.sort(strings, new Comparator<String>() {
            @Override
            public int compare(String o1, String o2) {
                //return o1.compareTo(o2);
                return o2.compareTo(o1);//降序
            }
        });
        System.out.println(Arrays.toString(strings));

    }
}
```

19.5 自定义类使用Comparable和Comparator

代码块

```
package com.powernode.array07;

import java.util.Arrays;
import java.util.Comparator;

class Teacher implements Comparable{
    private String name;
    private int age;

    public Teacher(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public int getAge() {
        return age;
    }

    @Override
    public String toString() {
        return "Teacher{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }

    @Override
    public int compareTo(Object o) {
        Teacher teacher = (Teacher) o;
       /* if (age >  teacher.age) {
            return 1;
        } else if (age < teacher.age) {
            return -1;
        }
        return 0;*/
        return age - teacher.age;
    }
}
public class Test {
    public static void main(String[] args) {
       Teacher t1 = new Teacher("zs",23);
```

```
        Teacher t2 = new Teacher("ls",20);
        Teacher t3 = new Teacher("ww",26);

        Teacher[] teachers = {t1, t2,t3};
         Arrays.sort(teachers);
         System.out.println(Arrays.toString(teachers));
         System.out.println("------------------------------");
         Arrays.sort(teachers, new Comparator<Teacher>() {
             @Override
             public int compare(Teacher o1, Teacher o2) {
                 return o2.getAge() - o1.getAge();
             }
         });
         System.out.println(Arrays.toString(teachers));
    }
}
```

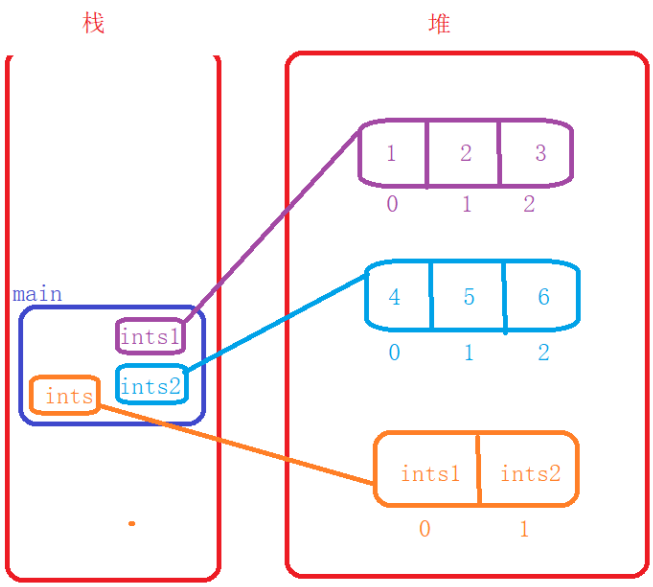
- Comparable：让类具有比较性
- Comparator：改变比较规则

20. 二维数组

- 数组中又存储了一个数组

20.1 二维数组的内存结构

```
package com.powernode.array08;

public class Test01 {
    public static void main(String[] args) {
        int[] ints1 = {1, 2, 3};
        int[] ints2 = {4, 5, 6};
        //二维数组中存储一维数组
        int[][] ints = {ints1, ints2};

        System.out.println(ints[0][0]);//1
        System.out.println(ints[0][1]);//2
        System.out.println(ints[0][2]);//3

        System.out.println(ints[1][0]);//4
        System.out.println(ints[1][1]);//5
        System.out.println(ints[1][2]);//6



    }
}
```



代码块

```java
package com.powernode.array08;

public class Test01 {
    public static void main(String[] args) {
        int[] ints1 = {1, 2, 3};
        int[] ints2 = {4, 5, 6};
        //二维数组中存储一维数组
        int[][] ints = {ints1, ints2};
        System.out.println(ints[0][0]);//1
        System.out.println(ints[0][1]);//2
        System.out.println(ints[0][2]);//3

        System.out.println(ints[1][0]);//4
        System.out.println(ints[1][1]);//5
        System.out.println(ints[1][2]);//6


    }
}
```

20.2 使用for循环取值

代码块

```java
package com.powernode.array08;

public class Test02 {
    public static void main(String[] args) {
        int[] ints1 = {1, 2, 3};
        int[] ints2 = {4, 5, 6};
        //二维数组中存储一维数组
        int[][] ints = {ints1, ints2};
        System.out.println(ints[0][0]);//1
        System.out.println(ints[0][1]);//2
        System.out.println(ints[0][2]);//3

        System.out.println(ints[1][0]);//4
        System.out.println(ints[1][1]);//5
        System.out.println(ints[1][2]);//6
        System.out.println("----------------------");
        //双层for，外层循环执行一次，内存循环执行n次
        //外层循环控制行，内存循环控制列
        for (int i = 0; i < ints.length; i++) {//ints.length 二维数组的长度
```

```
            for (int j = 0; j < ints[i].length; j++) {//ints[i].length ,一维数组的长度
                System.out.println(ints[i][j]);
            }

        }


    }
}
```

20.3 使用增强for

代码块

```
package com.powernode.array08;

public class Test03 {
    public static void main(String[] args) {
        int[] ints1 = {1, 2, 3};
        int[] ints2 = {4, 5, 6};
        //二维数组中存储一维数组
        int[][] ints = {ints1, ints2};
        System.out.println(ints[0][0]);//1
        System.out.println(ints[0][1]);//2
        System.out.println(ints[0][2]);//3

        System.out.println(ints[1][0]);//4
        System.out.println(ints[1][1]);//5
        System.out.println(ints[1][2]);//6
        System.out.println("----------------------");
        for (int[] anInt : ints) {
            for (int i : anInt) {
                System.out.println(i);
            }
        }


    }
}
```

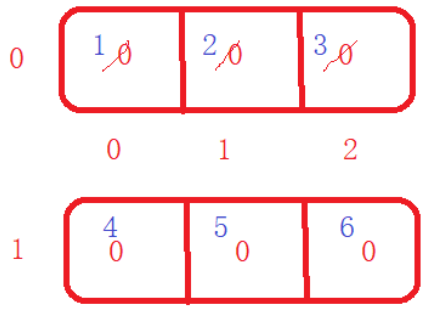
20.4 二维数组简单写法和重新赋值

代码块

```java
package com.powernode.array08;

public class Test04 {
    public static void main(String[] args) {
        /*int[] ints1 = {1, 2, 3};
        int[] ints2 = {4, 5, 6};*/
        //二维数组中存储一维数组
        int[][] ints = {{1, 2, 3}, {4, 5, 6}};

        ints = new int[][]{{7,8,9}, {1,2}};

        for (int[] anInt : ints) {
            for (int i : anInt) {
                System.out.println(i);
            }
        }

        //int [][] ints1 = new int[2][];

    }
}
```

20.5 二维数组动态初始化

```java
package com.powernode.array08;

public class Test05 {
    public static void main(String[] args) {
        int[][] ints = new int[2][3];

        ints[0][0] = 1;
        ints[0][1] = 2;
        ints[0][2] = 3;

        ints[1][0] = 4;
        ints[1][1] = 5;
        ints[1][2] = 6;

    }
}
```



2代表有几个数组
3代表每个数组有几个元素

可以理解为

2代表的是行
3代表的是列

代码块

```java
package com.powernode.array08;

public class Test05 {
```

```
    public static void main(String[] args) {
        int[][] ints = new int[2][3];
        ints[0][0] = 1;
        ints[0][1] = 2;
        ints[0][2] = 3;

        ints[1][0] = 4;
        ints[1][1] = 5;
        ints[1][2] = 6;

        /*for (int i = 0; i < ints.length; i++) {
            int[] anInt = ints[i];
            for (int j = 0; j < anInt.length; j++) {
                int i1 = anInt[j];
                System.out.println("i1 = " + i1);
            }
        }*/
        for (int i = 0; i < ints.length; i++) {
            for (int j = 0; j < ints[i].length; j++) {
                System.out.println(ints[i][j]);
            }
        }

    }
}
```

20.6 二维数组存储对象

代码块

```
package com.powernode.array08;

public class Test06 {
    public static void main(String[] args) {
        String[][] strings = {
                {"a", "b", "c"},
                {"d", "e", "f"}
        };
        /**
         * 输出的结果:
         *   row1:a   row1:b   row1:c
         *   row2:d   row2:e   row2:f
         */
```

```
14          for (int i = 0; i < strings.length; i++) {//外层循环控制行
15              String[] string = strings[i];
16              for (int j = 0; j < string.length; j++) {//内层循环控制列
17                  String s = string[j];
18                  System.out.print("row" + (i + 1) + ":" + s + "\t");
19              }
20              System.out.println();
21          }
22
23
24      }
25  }
```

作业

1. 调用JDK的方法：
   public static String toString(int[] a)　　将数组转换为字符串拼接 输出
   public static int[] copyOf(int[] original, int newLength)　　将数组扩容（original : 第一个参数需要扩容的数组，length：第二个参数，复制的长度）
   public static int[] copyOfRange(int[] original, int from, int to)　　将数组扩容到某个范围（original：第一个参数为要拷贝的数组对象，from：第二个参数为拷贝的开始位置（包含）to：第三个参数为拷贝的结束位置（不包含））
   public static void sort(int[] a)　　数组升序排序
   public static int binarySearch(int[] a, int key)　　数组的查找（ a 要搜索的数组， key要查找的值）
   public static void fill(int[] a, int val)　　想数组填充数据（a : 被填充的数组 ，val：填充的数据）
2. 自定义对象排序（按照年龄），使用Comparable和Comparator
   - Student（name，age)
3. 二维数组：
   a. 使用简单形式创建3行5列的二维String数组，值分别为 {1a,1b,1c,1d,1f}、{2g,2h,2i,2j,2k}、{3m,3n,3o,3p,3q} ；
   b. 使用增强型for循环遍历并打印数组。
   c. 使用for循环遍历数组，为第1行的所有列加上“row1:”前缀，为第2行的所有列加上“row2:”前缀，为第3行的所有列加上“row3:”前缀, 效果：
   d.　row1:1a　row1:1b　row1:1c　row1:1d　row1:1f
   e.　row2:2g　row2:2h　row2:2i　row2:2j　row2:2k
   f.　row3:3m　row3:3n　row3:3o　row3:3p　row3:3q

20.7