

JAVASE 复习题

目录

JAVASE 复习题

目录

一、1-3

- 1, classpath
- 2, ^异或
- 3, switch
- 4, break; contiune
- 5, 数组篇
 - 1, 写一个方法, 用三种方法, 交换两个数的值
 - 2, 自定义20个整数, 使用冒泡法排序
 - 3, 自定义20个整数, 使用选择排序法排序
 - 4, 自定义20个整数, 使用二分法查找目标值
 - 5, 请完成自定义MyArrayList类的数据结构 (增删改查)
 - 6, {3,1,6,5,8,2} -->反转数组 {2,8,5,6,1,3}
 - 7, 完成一个简易万年历, 输入任意一个年份月份日期, 打印一个当月日历 (星期一到星期日)
 - 8, Arrays类的使用, 使用Arrays类操作数组copy, 排序, 二分查找, 填充Arrays.fill (a, 2, 4, 100), 获取数组长度
 - 9, 使用增强for循环遍历自定义数组, 再试一试能否在循环中修改数组的值?

二、1-4

- 6, 继承题
- 7, 请你构建一个builder的设计模式Person p=Person.builder().id().name().build ();
- 8, super与this的相关面试题
- 9*, 继承, 抽象, 多态——葵花宝典, 100道, se复习概念, 网上再找找相关面试题
- 10, 多态的成员变量与函数使用与父子类分别关系, 静态方法又是如何使用?
- 11, 嘴巴说清楚object的equals, toString, hashCode, getClass这四个方法的内容
- 12, Iterator的内部实现, 将一串数组, 升序排列, 使用迭代的方式it.next()

一、1-3

1, classpath

如果没有定义classpath, java启动jvm后, 会在当前文件夹下寻找class文件,
如果指定了classpath, 会在指定的文件下面寻找class文件

- : 1、如果classpath结尾有';', 在具体路劲下没有找到class文件, 会在当前目录下再找一次
- : 2、如果classpath结尾没有';', 在具体路劲下没有找到class文件, 不会再当前目录查找

2, ^异或

两边结果一样, 就为false。0
两边结果不一样, 就为true。和

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

```
//      int a=8;//0000 1000
//      int b=4;//0000 0100
//      int c=a^b;//1100 --12
//      System.out.println(c);//12

//查找只出现一次的值
int[] nums= {2,2,3,3,4};
for(int i = 1;i< nums.length;i++){
    nums[0] = nums[0]^nums[i];
}
System.out.println(nums[0]);
}
private static void method2() {
    int a = 15;//1111
    int b = 2;//10
    System.out.println(Integer.toBinaryString(a));//获取二进制码
    System.out.println(Integer.toBinaryString(b));
    int c =a ^ b;//1101----> 13
    System.out.println("a^b=" + c);
    System.out.println(Integer.toBinaryString(c));
}
}
```

3, switch

```
public static void method1(){//b c d
    int num = 4;
    switch (num) {
        case 9:
            System.out.println("a");
        default:
            System.out.println("b");
        case 3:
            System.out.println("c");
        case 6:
            System.out.println("d");
            break;
    }
}
}
```

4, break; contiune

```
private static void method1() {
    for(int i = 0;i<3;i++) {
        for(int j = 0;j<3;j++) {
            if(j==1) {
                break;// j=0 i=0 j=0 i=1 j=0 i=2
                //continue;// j=0 j=2 i=0 j=0 j=2 i=1 j=0 j=2 i=2
            }
            System.out.print(" j="+j);
        }
        System.out.print(" i="+i);
    }
}
}
```

5, 数组篇

1, 写一个方法, 用三种方法, 交换两个数的值


```
/**
 * 方式一: 赋值交换
 */
private static void method1() {
    int num1 = 88;
    int num2 = 99;
    int num3 = num1;
    System.out.println(num1 + "--原配--" + num2);
    num1 = num2;
    num2 = num3;
    System.out.println(num1 + "--交换--" + num2);
}

/**
 * 方式二: 通过加减法
 */
private static void method2() {
    int a = 100;
    int b = 101;
    System.out.println("原: " + a + ", " + b);
    a = a + b; //201
    b = a - b; //201-101> b=100
    a = a - b; //201-100> a=101
    System.out.println("交换: " + a + ", " + b);
}

/**
 * 方式三: 通过^异或
 */
private static void method3() {
    int a = 100;
    int b = 101;
    System.out.println("原: " + a + ", " + b);
    a = a^b;
    b = b^a;
    a = a^b;
    System.out.println("交换: " + a + ", " + b);
}

/**
 * 方式四: 通过数组
 */
private static void method4() {
    int num1 = 999;
    int num2 = 888;
    int[] arr = {num1, num2};
    num2 = arr[0];
    num1 = arr[1];
    for(int arrs:arr) {
        System.out.print(arrs+"原 ");
    }
    System.out.println("***"+num1+"--交换--"+num2);
}
```


2, 自定义20个整数, 使用冒泡法排序

 image-20200103173351070

```
//冒泡法排序(升序排列)
//比较相邻的元素。如果第一个比第二个大,就交换它们两个;
//对每一对相邻元素作同样的工作,从开始第一对到结尾的最后一对,这样在最后的元素应该会是最大的数;
//针对所有的元素重复以上的步骤,除了最后一个;
//重复步骤1~3,直到排序完成。
```

```
public static void method1() {
    // 自定义20个整数,使用冒泡法排序
    int arr[] = new int[20];
    for (int l = 0; l < arr.length; l++) {
        arr[l] = (int) (Math.random() * 100 + 1);
    }
    // 排序前,先把内容打印出来
    System.out.print("原数组: \n");
    for (int data : arr) {
        System.out.print(data + " ");
    }
    System.out.println("\n --- --- ---");
    // 冒泡法排序
    for (int j = 0; j < arr.length - 1; j++) {
        for (int i = 0; i < arr.length - j - 1; i++) {
            if(arr[i]>arr[i+1]) {
                int temp = arr[i];
                arr[i] = arr[i+1];
                arr[i+1] = temp;
            }
        }
        // 把内容打印出来
        for (int data : arr) {
            System.out.print(data + " ");
        }
        System.out.println(" ");
    }
}
```

3, 自定义20个整数, 使用选择排序法排序

 image-20200103173402405

```
//选择法排序的思路:
//把第一位和其他所有的进行比较,只要比第一位小的,就换到第一个位置来
//比较完后,第一位就是最小的
//然后再从第二位和剩余的其他所有进行比较,只要比第二位小,就换到第二个位置来
//比较完后,第二位就是第二小的
//以此类推
```

```
public static void method1() {
    // 自定义20个整数,使用选择排序
    int arr[] = new int[20];
    for (int l = 0; l < arr.length; l++) {
        arr[l] = (int) (Math.random() * 100 + 1);
    }
    // 排序前,先把内容打印出来
```

```

System.out.print("原数组:  \n");
for (int data : arr) {
    System.out.print(data + " ");
}
System.out.println("\n --- --- --- ---");

// 移动的位置是从0逐渐增加的
// 所以可以在外面套一层循环
for (int j = 0; j < arr.length - 1; j++) {
    for (int i = j + 1; i < arr.length; i++) {
        if (arr[i] < arr[j]) {
            int temp = arr[j];
            arr[j] = arr[i];
            arr[i] = temp;
        }
    }
}
// 把内容打印出来
for (int data : arr) {
    System.out.print(data + " ");
}
System.out.println(" ");
}
}

```

4, 自定义20个整数, 使用二分法查找目标值

```

public static void main(String[] args) {
    int[] arr = new int[20];
    for (int l = 0; l < arr.length; l++) {
        arr[l] = (int) (Math.random() * 100 + 1);
    }
    Arrays.sort(arr);
    //int arr[]= {1,2,13,15,25,30,32,33,39,45,49,54,95};
    System.out.println(halfSeach_2(arr, 25));
}

public static int halfSeach_2(int[] arr,int key){
    int min,max,mid;
    min = 0;//7
    max = arr.length-1;
    System.out.println(max+" 最大值");
    mid = (max+min)>>1; //(max+min)/2; (max+min)>>1; 结果一样
    System.out.println(mid+" 中间值");
    System.out.println(min+" "+max+" "+mid);
    while(arr[mid] !=key){//查找值不等于数组中间值的值
        if(key>arr[mid]){//如果大于中间值, 最小值就=中间值+1
            min = mid + 1;
        }
        else if(key<arr[mid]){//如果查找值小于中间值
            max = mid - 1;//最大值=中间值-1
        }
        if(max<min)//如果最大值小于最小值, 返回-1
            return -1;
        System.out.println(min+" "+max+" "+mid);
        mid = (max+min)>>1; //中间值=
    }
    return arr[mid];
}

```

5, 请完成自定义MyArrayList类的数据结构（增删改查）

```
class MyArray {
    // 准备数据空间
    private int[] data;
    // 定义元素个数
    private int size;
    // 定义初始化数组容量
    private int capacity = 16;
    // 默认构造函数：初始化容量
    MyArray(int capacity) {
        // 初始化设置容量
        if (capacity <= 0) {
            data = new int[this.capacity];
        } else {
            data = new int[capacity];
        }
        this.size = 0;
    }
    //
    MyArray() {
        this(10);
    }
    // -----基本方法-----
    public int getSize() {
        return size;
    }
    // 获取数组空间元素个数
    public int getCapacity() {
        return data.length;
    }
    // 返回空元素个数
    public boolean isEmpty() {
        return size == 0;
    }
    // -----添加元素-----
    // 末尾添加
    public void addLast(int e) {
        insert(size, e);
    }
    // 开头添加
    public void addFirst(int e) {
        insert(0, e);
    }
    // 插入方法
    private void insert(int index, int e) {
        // 如果我的容量已满、需要扩容
        if (size == data.length) {
            // throw new RuntimeException("容量已满");
            resize(2 * data.length);
        }
        if (index < 0 || index > size) {
            throw new IllegalArgumentException("插入失败,下标越界");
        }
        for (int i = size - 1; i >= index; i--) {
            data[i + 1] = data[i]; // 当前数据（后面的值）都向后移动
        }
    }
}
```

```

        data[index] = e;
        size++;
    }
    // -----删除-----
    // 通过下标
    public int remove(int index) {
        if (index < 0 || index > size) {
            throw new IllegalArgumentException("删除失败,下标越界");
        }
        // 拿到这个值
        int ret = data[index];
        //
        for (int i = index + 1; i < size; i++) {
            data[i - 1] = data[i]; // 向前移
        }
        size--;
        if (size == data.length / 2 - 1) {
            // 收缩容量
            delSize();
        }
        return ret;
    }
    // 通过数据值
    // -----改-----
    public void set(int index, int e) {
        if (index < 0 || index > size) {
            throw new IllegalArgumentException("修改失败,下标越界");
        }
        data[index] = e;
    }
    // -----查-----
    // 获取元素
    public int get(int index) {
        if (index < 0 || index > size - 1) {
            throw new IllegalArgumentException("获取失败,下标越界");
        }
        return data[index];
    }
    // 查找值
    public boolean contains(int e) {
        for (int i = 0; i < size; i++) {
            if (data[i] == e) {
                return true;
            }
        }
        return false;
    }
    // 扩容
    private void resize(int newCapacity) {
        // 新数组
        int[] newData = new int[newCapacity];
        // 把老数组数据复制到新数组
        for (int i = 0; i < data.length; i++) {
            newData[i] = data[i];
        }
        data = newData;
    }
    // 收缩容量

```

```

        private void delSize() {
            int[] newData = new int[data.length / 2];
            for (int i = 0; i < newData.length; i++) {
                newData[i] = data[i];
            }
            data = newData;
        }
        @Override
        public String toString() {
            return "MyArray [data=" + Arrays.toString(data) + "]";
        }
    }
    /**
     * 自定义数组集合
     *
     * @author jinfang yu
     */
    public class MyArrayList {
        public static void main(String[] args) {
            MyArray list = new MyArray();
            list.addFirst(5); // 开头
            list.addFirst(19);
            list.addFirst(89);
            list.addFirst(69);
            list.addFirst(459);
            list.addFirst(912);
            list.addFirst(900);
            list.addFirst(900);
            list.addFirst(900);
            list.addFirst(900);
            list.addFirst(90); // 开头
            list.addLast(3); // 末尾
            System.out.println(list.toString());
            System.out.println(list.get(1)); // 获取下标元素
            list.set(0, 99); // 改
            System.out.println(list.toString());
            System.out.println(list.contains(8));
            list.remove(0); // 删除
            System.out.println(list.toString());
        }
    }

```

6, {3,1,6,5,8,2} --> 反转数组 {2,8,5,6,1,3}

```

    public static void main(String[] args) {
        String[] arr = new String[20];
        for (int i = 0; i < arr.length; i++) {
            int ars = (int) (Math.random() * 100 + 1);
            arr[i] = String.valueOf(ars);
        }
        for (String arrs : arr) {
            System.out.print(arrs + " ");
        }
        System.out.println("\n-----反转后-----");
        for (String arrs : method1(arr)) {
            System.out.print(arrs + " ");
        }
    }
}

```



```

/**
 * 准备一个数组、遍历交换位置 把第一为
 */
private static String[] method1(String[] array) {
    String[] newArray = new String[array.length];
    for (int i = 0; i < newArray.length; i++) {
        newArray[i] = array[array.length - i - 1];
    }
    return newArray;
}

```

7, 完成一个简易万年历, 输入任意一个年份月份日期, 打印一个当月日历 (星期一到星期日)

```

//输入日期
//设置日期格式 SimpleDateFormat ()
//将String输入日期转化为 Date格式 df.parse(str);
//将时间转换为日历格式 Calendar cal = new GregorianCalendar(); cal.setTime(d);
//获取日 (几号 cal.get(Calendar.DAY_OF_MONTH);
//获取该日是星期几 cal.get(Calendar.DAY_OF_WEEK);
//获取当前月最大天数 cal.getActualMaximum(Calendar.DAY_OF_MONTH);
//打印标题
//

//通过date实现
public static void main(String[] args) throws ParseException {
    Scanner sc = new Scanner(System.in);
    System.out.println("请输入一个日期 yyyy-MM-dd");
    // 输入一个日期
    String str = sc.nextLine();
    DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
    // 设定格式
    Date d = df.parse(str);
    // 获取时间
    Calendar cal = new GregorianCalendar();
    cal.setTime(d);
    // 计算机获取时间
    int nowdate = cal.get(Calendar.DAY_OF_MONTH);
    // 获取输入的是月的第几天
    int dayofweek = cal.get(Calendar.DAY_OF_WEEK);
    // 获取输入的是一周的第几天
    int maxday = cal.getActualMaximum(Calendar.DAY_OF_MONTH);
    // 获取输入的设计一月的最后一天
    System.out.println("日\t一\t二\t三\t四\t五\t六");
    for (int i = 1; i < dayofweek; i++) {
        System.out.print("\t");
    }
    for (int i = 1; i <= maxday; i++) {
        System.out.print(i);
        if (i == nowdate) {
            System.out.print("*");
        }
        System.out.print("\t");

        if (cal.get(Calendar.DAY_OF_WEEK) == Calendar.SATURDAY) {
            // 到周六换行

```

```

        System.out.println();
    }
    cal.add(Calendar.DAY_OF_MONTH, 1);
    // 循环一次，换行的日期增加一天
}
sc.close();
}

```

//通过日历实现

```

class MyCalender {
    int year;//年
    int month;//月
    Calendar firstDayOfMonth;//日历
    int date;//该月总共有多少天
    int firstDay_s_week;// 0是星期日
    MyCalender(int year, int month) {
        this.year = year;
        this.month = month;
        firstDayOfMonth = Calendar.getInstance();
        firstDayOfMonth.set(year, month - 1, 1);
        date = firstDayOfMonth.getActualMaximum(firstDayOfMonth.DATE);
        firstDay_s_week = firstDayOfMonth.get(firstDayOfMonth.DAY_OF_WEEK) - 1;
    }
    public void coutCalendar() {
        System.out.println("星期日\t星期一\t星期二\t星期三\t星期四\t星期五\t星期六");
        for (int i = 0; i < firstDay_s_week; i++)// 找到该月第一天的位置
        {
            System.out.print("\t");
        }
        int tmp = firstDay_s_week;
        for (int i = 1; i <= date; i++) {
            System.out.print(i + "\t");
            tmp++;
            if (tmp == 7) {
                tmp = 0;
                System.out.println();
            }
        }
    }
}
public class Demo6 {
    public static void main(String[] args) {
        MyCalender mc = new MyCalender(2019, 1);// 在这里输入年和月
        mc.coutCalendar();
    }
}

```

image-20200103203319343

8, Arrays类的使用, 使用Arrays类操作数组copy, 排序, 二分查找, 填充Arrays.fill (a, 2, 4, 100) , 获取数组长度

```

public class Demo7 {

    public static void main(String[] args) {
        int[] arr = { 12, 22, 67, 23, 78 };
    }
}

```

```

        method4();
        // System.out.println(method3(arr, 67));
    }

    // 获取数组长度
    /**
     * 填充Arrays.fill
     */
    private static void method4() {
        int[] a = new int[] { 1, 2, 3, 4, 5, 6 };
        Arrays.fill(a, 0);
        System.out.println(Arrays.toString(a)); //[0, 0, 0, 0, 0, 0]

        // fromIndex - 要用指定值填充的第一个元素（含）的索引
        // toIndex - 要用指定值填充的最后一个元素（排他）的索引
        // val -要存储在数组的所有元素中的值
        Arrays.fill(a, 1, 5, 33);
        System.out.println(Arrays.toString(a)); //[0, 33, 33, 33, 33, 0]
        //
    }

    /**
     * 二分查找
     */
    private static int method3(int[] arr, int key) {
        int min, max, mid;
        min = 0;
        max = arr.length - 1;
        mid = (max + min) >> 1; // (max+min)/2; (max+min)>>1; 结果一样
        while (arr[mid] != key) { // 查找值不等于数组中间值的值
            if (key > arr[mid]) { // 如果大于中间值，最小值就=中间值+1
                min = mid + 1;
            } else if (key < arr[mid]) { // 如果查找值小于中间值
                max = mid - 1; // 最大值=中间值-1
            }
            if (max < min) { // 如果最大值小于最小值，返回-1
                return -1;
            }
            mid = (max + min) >> 1; // 中间值=
        }
        return mid;
    }

    /**
     * 排序
     */
    private static void method2() {
        int[] arr1 = { 12, 5, 67, 23, 78 };
        int[] arr2 = new int[arr1.length];
        arr2 = Arrays.copyOf(arr1, arr1.length);
        Arrays.sort(arr2);
        for (int arr : arr2) {
            System.out.print(arr + " ");
        }
    }

    /**
     * 复制数组

```

```

    */
    private static void method1() {
        int[] arr1 = { 12, 5, 67, 23, 78 };
        int[] arr2 = new int[arr1.length];
        arr2 = Arrays.copyOf(arr1, arr1.length);
        for (int arr : arr2) {
            System.out.print(arr + " ");
        }
        Arrays.sort(arr2);
    }
}

```

9，使用增强for循环遍历自定义数组，再试一试能否在循环中修改数组的值？

```

//使用增强for循环遍历自定义数组，再试一试能否在循环中修改数组的值？
private static void method2() {
    int[] arr = new int[20];
    for (int l = 0; l < arr.length; l++) {
        arr[l] = (int) (Math.random() * 100 + 1);
    }
    // 排序前，先把内容打印出来
    System.out.print("原数组: \n");
    for (int data : arr) {
        System.out.print(data + " ");
    }
}

```

增强for循环只做遍历，不做修改

二、1-4

6，继承题

```

public class Demo {
    public static void main(String[] args) {
        new Son().show();
        Fu fu = new Son();
        fu.show();
    }
    class Fu{
        String name="f";
        public void show() {
            System.out.println(name);
        }
    }
    class Son extends Fu{
        String name="s";
        public void show() {
            //String name="s1";
            System.out.println(name);
        }
    }
}

```

7, 请你构建一个builder的设计模式Person

p=Person.builder().id().name().build ();

```
/**
 * 建造者模式Builder
 * @author jinfang yu
 * @version 1.8
 */
public class Demo4 {
    public static void main(String[] args) {
        Person son = new Person().Builders()
            .setId(1)
            .setName("shang")
            .setAge(12)
            .setSex(15)
            .build();
        System.out.println(son.toString());
    }
}

/**
 * 人类
 * @author jinfang yu
 * @version 1.8
 */
class Person {
    private Integer id;// 编号
    private String name;// 姓名
    private Integer age;// 年龄
    private Integer sex;// 性别

    public Person() {
    }
    public Builders Builders() {
        return new Builders();
    }
    public Person(Builders builder) {
        this.id = builder.id;
        this.name = builder.name;
        this.age = builder.age;
        this.sex = builder.sex;
    }
    ....getter\setter方法
    @Override
    public String toString() {
        return "Person [id=" + id + ", name=" + name + ", age=" + age + ", sex="
+ sex + "]\n";
    }

    /**
     * 建造者类
     *
     * @author jinfang yu
     * @version 1.8
     */
    static class Builders {
```

```

        // id、name、age、sex
        private Integer id;// 编号
        private String name;// 姓名
        private Integer age;// 年龄
        private Integer sex;// 性别

        public Builders setId(Integer id) {
            this.id = id;
            return this;
        }
        public Builders setName(String name) {
            this.name = name;
            return this;
        }
        public Builders setAge(Integer age) {
            this.age = age;
            return this;
        }
        public Builders setSex(Integer sex) {
            this.sex = sex;
            return this;
        }
        public Person build() {
            return new Person(this);
        }
    }
}

```

8, super与this的相关面试题

image-20200106085811503

this是自身的一个对象，代表对象本身，可以理解为：指向对象本身的一个指针。

super可以理解为是指向自己超（父）类对象的一个指针，而这个超类指的是离自己最近的一个父类。

当子类构造函数需要显示调用父类构造函数时，**super()** 必须为构造函数中的第一条语句。

9*, 继承，抽象，多态——葵花宝典，100道，se复习概念，网上再找找相关面试题

10, 多态的成员变量与函数使用与父子类分别关系，静态方法又是如何使用？

11, 嘴巴说清楚object的equals, toString, hashCode, getClass这四个方法的内容

“==”比较两个变量本身的值，即两个对象在内存中的首地址。

“equals()”比较字符串(对象)中所包含的内容是否相同。

String toString() 返回该对象的字符串表示。通常，ToString方法会返回一个“以文本方式表示”此对象的字符串。结果应是一个简明但易于读懂的信息表达式。

toString() 只适用于对象的调用，普通的数据类型不可以调用，这也就是使用包装类的原因

hashCode() 方法用于返回字符串的哈希码。

getClass() 它可以获得一个实例的类型类。

12, Iterator的内部实现，将一串数组，升序排列，使用迭代的方式 it.next()

```
ArrayList list = new ArrayList();  
    list.add("a");  
    list.add("b");  
    list.add("c");  
    Iterator it = list.iterator();  
    while(it.hasNext()){  
        String str = (String) it.next();  
        System.out.println(str);  
    }
```

