7天总结 自我答题

1. Java概述与环境搭建

1.1 JDK ,JRE ,JVM分别是什么，有什么关系

> JVM 是Java的虚拟机 类似于模拟器，通过JVM可以让Java程序在不同的系统上运行
>
> JRE 是Java的运行环境 Java running environment
>
> JDK 是Java的开发工具包 其中也包括了JRE Java development kit
>
> 他们的关系是
>
> JVM < JRE < JDK

1.2 环境变量配置为jdk的哪个目录

> 环境变量配置到jdk中的 `bin` 目录下
>
> bin 为 `binary` 单词的简写 binary为二进制的意思 表示此文件夹下存放的全部为可执行

1.3 编译命令是什么，编译是将什么文件编译为什么文件

> 编译命令是 `Javac`
>
> 通过Javac 命令可以使我们的.Java文件编译成为 文件名.class文件

1.4 使用什么命令运行class文件

> 使用 `Java 文件名` 就可以在控制台运行 `.class` 文件

1.5 Java跨平台的核心是什么

> Java 跨平台的核心就是 Java 虚拟机. 也就是JVM 通过Java虚拟机可以使我们在不同的平台下或者不同的系统上模拟相同的环境 实现跨平台的操作

1.6 公开的类，名称有什么要求

> 是一个 **大写驼峰** 的命名规则 而且 **类名要与文件名字相同** 首字母大写 其后每个单词的首字母大写

1.7 一个源文件中可以书写多个类吗

> 一个源文件中可以书写多个类 但是只能有一个公共类

1.8 一个源文件中是否可以包含两个同名的类

> 一个源文件中 **不可以** 包含两个不同名的类

2. 基础数据类型与运算符

2.1 变量的命名规范

> 字下美人数驼峰
>
> 不能以数值开头
>
> 可以以 下划线，美元符号，人民币符号开头
>
> 其中可以包括数字
>
> 小写驼峰 见面之意 有意义的命名

2.2 类名的命名规范

> 大写驼峰 首字母大写 其后每个单独的单词首字母大写 其余小写 见面知义

2.3 自动类型转换的前提条件?

> 自动类型提升的前提是 提升的类型 要大于源数据类型 而且其中包括其中的数据类型

```
public static void mian(String[] args){
    short a = 10;
    int b = a;
}
```

2.4 强制类型转换的写法?

> 强制类型转换的写法

```
public static void main(String[] args){
    short a = 10;
    byte b = (byte) a;
}
```

2.5 八种基本数据类型有哪些?

> 其中包括
>
> **整数型**
>
> - byte 1字节 8个位
> - short 2字节
> - int 4字节
> - long 8字节
>
> **浮点型**
>
> - float 4字节
> - double 8字节
>
> **布尔型**
>
> - boolean 1字节
>
> **字符型**
>
> - char 2字节

2.6 什么是常量

> 常量是指在程序中无法改变的数据

2.7 byte的取值范围是什么? 为什么取这个范围

> byte 取值范围是 127~ -128 因为1 byte 是一个字节 一个字节占8个位 其中每个位只能是0和1 那么二进制的首位是符号位 所以 二进制中 一个 byte 最大我7个1 一个0 所以 换算为 127 和 -128

2.8 String属于基本数据类型还是引用数据类型?

> String 是属于引用数据类型

2.9 什么是标识符?

> 凡是需要自定义名称的内容都属于标识符

2.10 分别写明你知道的位运算符的运算方式

> '>> 2' 右移两位
>
> << 2 左移两位
>
> '>>>' 无符号填充 移动后的位置都使用 0来作为填充

& 两个有一个为1 取值1

| 两个都为1 取值1

~ 取反

^ 改0为1 改1为0

2.11 ++表示什么意思？在前在后的区别

前++为 先自增 后运算

后++为 先运算 后自增

2.12 &&和&的区别?

&& 是短路与 有短路效果 如果前面条件不符合 不会在判断后续条件 具有短路效果

& 无论前面是否条件通过 都会对后面的条件进行判断

2.13 ||和|的区别?

|| 是短路或 有短路效果 如果前面条件符合 不会在判断后续条件 具有短路效果

| 无论前面是否条件通过 都会对后面的条件进行判断

2.14 如何接收用户输入的信息，分别写明接收不同类型数据的方法

使用 Scanner 类 调用 Java.util.Scanner 工具包

接收数据类型的不同方法

整数型

- nextByte()
- nextShort()
- nextInt()
- nextLong()

浮点型

- nextFloat()
- nextDouble()

布尔型

- nextBoolean()

字符串

- next()

得到一整行数据 包括 符号 空格 数字 字符 等等

- nextLine()

3. 选择/循环结构

3.1 if(表达式) 表达式的最终值为什么类型?

最终的返回值类型为 **布尔型 Boolean**

3.2 多重if用来处理什么样的情况? 多重if中的else必须写吗?

多重if的判断通常使用在 返回的类型 在3个以上的情况 而多重if中是可以不写else的

3.3 多重if处理区间的值，条件编写有什么顺序要求

多重if在处理区间的情况下 判断条件需要重大到小 或者 重小到大的顺序排列 不可以乱序

3.4 switch支持的数据类型以及break在switch中的作用

switch 中 break 主要作用是防止case 击穿 终止判断的作用

3.5 变量的命名规范

小写驼峰 见名知义

3.6 学过哪些循环，分别有什么特点，循环四个必不可少的部分是什么

for 循环

```
for(int i = 0; i < lenght; i++){
    循环体
}
```

while 循环

```
int i = 0;
while(i < 10){
    循环体
    i++;
}
```

> do while 循环

```
int i = 0
do{
    循环体
    i++;
}while(i < 10);
```

> 其中包含的4个结构
>
> - 计数器
> - 循环条件
> - 循环体
> - 计数器变化

3.7 三种循环的执行和应用场景区别

> for 循环 通常使用在 循环次数确定的情况
>
> do while 和 while 循环 通常使用在 循环次数不确定的情况下

4. 方法

4.1 如何定义方法，定义在哪里？如何调用方法?

> 定义方法:

```
public static void getName(){
    方法体
}
```

> 定义方法时 需要与main方法同级

```
public class Test{
    public static void mian(String[] args){

    }

    public static void getName(){

    }

}
```

> 调用方法时 使用方法名 调用方法

4.2 形参和实参的区别？形参规定了哪些内容

> 形参是方法定义时确定的 实参是用户提供的实际参数
>
> 形参规定了 传入实参的 **个数 类型 顺序**

4.3 break和continue的区别

> break 在循环中 使用的意思是 结束循环 在 switch 中 防止 case击穿 结束判断
>
> continue 在循环中 表示 跳过本次循环 后续代码不执行 继续后续代码

```
for(int i = 0; i < 10; i++){
    if(i==5){
        continue;
    }
    在continue后续代码无效 而且while中计算器改变在continue后 会导致是循环
}
```

4.4 双重循环外层循环变量与内层循环变量的关系

> 外层循环变量变化一次 内层循环变量变化一轮

4.5 switch支持的数据类型

> switch 中支持 byte , short , int , char , String(JDK 1.7 加入), 枚举

4.6 描述局部变量：定义位置 默认值 作用范围 是否可重名等几个方面

> 书写在方法体类
>
> 没有默认值 需要赋值后 进行使用
>
> 作用范围在 最近的一个大括号{ }内

> 在一个作用范围内不可以重名

4.7 包名的命名规范，类名的命名规范

> 包的命名为 域名倒序 如 `baidu.com` 命名为 `com.baidu.包名`

5. 数组

5.1 用一句话描述数组（三个特点）

> 内存中一块连续的储存空间，存储同一类型的数据，长度是固定的

5.2 数组的长度是哪个属性

> length 关键字

```
int[] arr = new int[]{1,2,3}
arr.lenght
```

5.3 数组的下标从几开始

> 数组的下标从 0开始

5.4 数组在内存中的位置，数组名字和数组中的值

> 数值名字 存储在 栈 Stack
>
> 数值中的值 存储在 堆 Heap

5.5 创建数组的四种方式

```
//第一种
int[] arr;
arr = new int[3]{};

//第二种
int[] arr = new int[3]{};

//第三种
int[] arr = new int[]{1,2,3};

//第四种
int[] arr = {1,2,3};
```

5.6 求数组中最大值的思路是什么

> 定义一个值 为数组中的第一个元素 然后使用这个值对比数组中的每一个元素 如果条件成立 就替换双方的值 最后得出需要的数值
>
> 求最大和求最小都是同理

```
int[] arr = {1,2,3};
int max = arr[0];
for(int i = 1; i < arr.lenght; i++){
    if(max < arr[i]){
        max = arr[i];
    }
}
```

5.7 分别写明各个类型数组的默认值

> 其中不同的数据的默认值为
>
> byte， short， int， long 为 0
>
> float， double 为 0.0
>
> char 为 \u 0000
>
> boolean 为 false
>
> 引用数据类型为 null 如 String

5.8 数组长度为3，添加下标为3的元素可以吗？会发生什么

> 会发生角标越界错误 程序终止 控制台显示异常 `IndexOutOfBoundsException`

5.9 return关键字的几种使用情况 分别说明

> return 在有返回值的方法中 会需要返回相对应返回值的 数据类型的元素
>
> return 在没有返回值的方法中 表示终止方法 后续代码不在继续执
>
> return 在分支结构中必须保证每一条分支都有正确的返回值

5.10 方法重载的概念

> 方法的重载 同一个类和父子类的方法 名称相同 形参列表的（ **个数 ， 类型 ， 顺序** ）不同 都可以算是方法的重载 跟返回值 权限修饰 没有关系

6. 冒泡排序 和 选择排序 二分查找

```java
package day07.Exer;

import java.util.Arrays;

//冒泡排序
public class Test {
    public static void main(String[] args) {
        int[] arr = {3,12,436,558,43,65,31,87,42,67,213,4579,13,-1,4,-542,134};

        for (int i = 0; i < arr.length-1; i++) { // 冒泡次数
            for (int j = 0; j < arr.length-1-i ; j++) {
                // 最长的一次的次数为 数组长度减一 其后每次自减一
                if (arr[j] > arr[j+1]){
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1]=temp;
                }
            }
        }
        //打印数组
        System.out.println(Arrays.toString(arr));

    }
}
```

```java
package day07.Exer;

import java.util.Arrays;

//选址排序
public class Test1 {
    public static void main(String[] args) {
      int[] arr = {23,14,65,87,213,98,452,13,56,8,0,32,65,98,-13,-65,-13,-43,56,7,-8};

        for (int i = 0; i < arr.length - 1; i++) {
            // 使用第一个数组遍历 每次对比两个 所有最后一个不用考虑到

            //定义索引
            int index = i;
            for (int j = i+1; j < arr.length ; j++) {
                //每次考虑的数值为 前数值的后一个 所有 j = i+1;
                if (arr[index] > arr[j]){
                    index = j;
                }
            }

            if (index != i){
                int temp = arr[index];
                arr[index] = arr[i];
                arr[i] = temp;
            }
        }
```

```
        System.out.println(Arrays.toString(arr));
    }
}
```

```
package day07.Exer;

//二分查找
public class Test3 {
   public static int getIndex(int[] array , int target){
       int index = -1;
       int head = 0;
       int end = array.length-1;
       int midden = (head + end) / 2;

       while (head <= end){
           if (array[midden] > target){
               end = midden -1;
           } else if (array[midden] < target) {
               head = midden+1;
           }else {
               index = midden;
               break;
           }
           midden = (head + end ) /2;
       }

       return index;
   }

    public static void main(String[] args) {
        int[] array = {1,2,3,4,5,6,7,8,9,10};

        int index = getIndex(array, 5);

        System.out.println("index = " + index);
    }

}
```