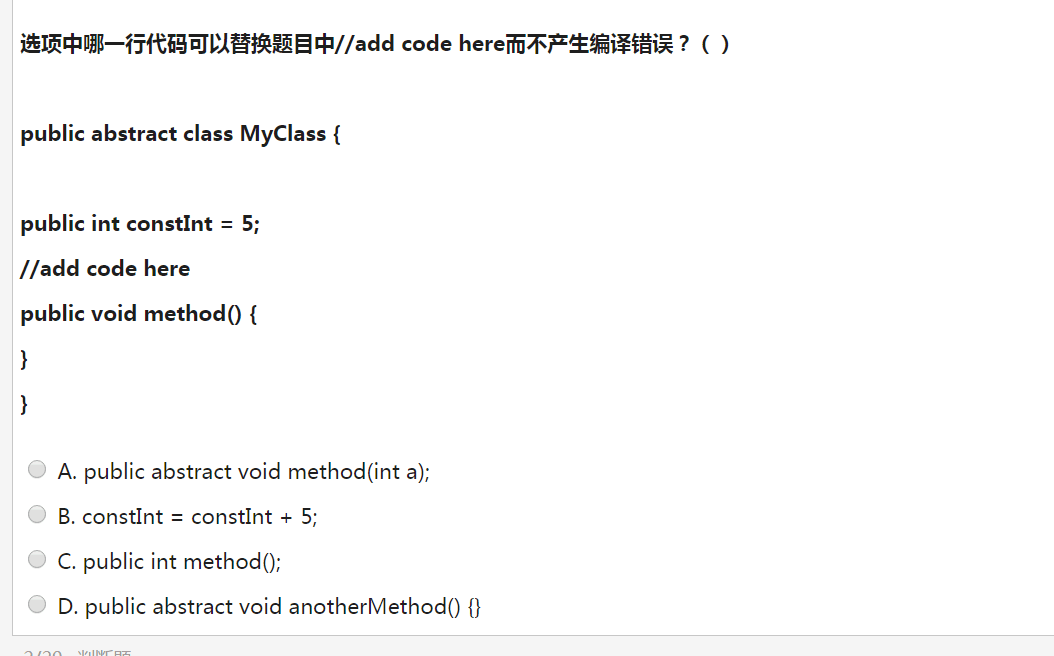
Java面试题

1. 

abstract class MyClass

{

public int constInt = 5;

//constInt = constInt + 5; //需要标识符，要么定义在方法内，要么用大括号括起来，作为构造代码块或者加上static作为静态代码块

//public int method();//要重载方法，只能通过参数的类型和顺序区别不同方法，不能通过返回值得类型进行辨别

//public abstract void method(int a);//加上这句话，编译不会出现错误

//public abstract void anotherMethod(){}//错误的，因为抽象方法不能有主体

public void method() {

}

}

1. 面试时如果需要上机写程序，一定要写注释，不仅注重结果，还会注重自己程序员的基本素养和规范。并且在写代码前需要先写些自己的思想，然后写代码，因为程序写的就是想法。所以在写代码之前，先写一块代码块注释，写明 需求、思路、步骤。

**代码只是思想的一种体现形式。**

如：

/\*

需求： %%%%%%%%%%%%%%

思路：%%%%%%%%%%%%%%

步骤：（1）%%%%%（2）%%%%%%%%%%%%%%%%

\*/

1. . 关于二分查找的一个面试题目： 给定一个有序的数组，如果往该数组中存储一个元素，并保证这个数组还是有序的，怎么获取这个元素应该存放的角标。

答 ： 利用JDK提供的Arrays.binarySearch（）方法即可实现。如果数组中存在查找的的数据，返回的就是此数值的角标，如果不存在此数值，返回的就是一个负数，其绝对值是此数值应当插入的第几个位置处（也就是索引加1）.在刚才的基础上把-1换成min就可以了。

解析：如果要存放的元素，在数组中存在，则直接找到此元素的索引即可，但是如果不存在此元素，那么最后肯定会是min 和 max值是紧挨着的，下一步就是min 加1，而max减1；且分析知需要存放的数据应该存放到到大于此数值的第一个元素位置，所以返回min。例如：arr[9]=100;arr[10]=200; key = 150; min = 9;max = 10;mid = 9;接下来，由于key 大于arr[mid] 所以，min = mid +1 =10,所以返回min = 10,正确。

1. 面试时通常会问你怎么理解面向对象？

答：符合人们的思考习惯，使复杂的事情简单化，让我们从曾经的执行者变成了指挥者。一般都要加以例子进行说明：面试官，您本身就是在利用面向对象的思想解决问题，（引起面试官的注意，会问为什么？）因为以面试官您的能力而言，如果来了一个大项目的话，您从项目的需求分析、开发设计、测试到完成，都可以完成，但是要耗费很多时间，您为了提高效率，就需要找一些具备专业编程的人完成这个项目，也就是相当于在找对象，我就是具备专业编程经验的对象，您只要指挥我这个对象去做事情就可以了，我会给您一个非常满意的结果，至于过程，你可以不用关心，所以您就是正在利用面向对象思想来提高工作效率。

举例子要贴切生活。

1. **class** HelloA {

**static** {

System.***out***.println("static A");}

HelloA() {

System.***out***.println("Hello A");}

{System.***out***.println("I'm A class."); }

}

**public** **class** HelloB **extends** HelloA {

**static** {

System.***out***.println("static B"); }

HelloB() {

System.***out***.println("Hello B"); }

{System.***out***.println("I'm class B.");}

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

**new** HelloB(); }

}

上面程序的运行结果是什么？

答：（真正操作结果）

static A

static B

I'm A class.

Hello A

I'm class B.

Hello B

1. 要求利用匿名内部类调用接口中的两个方法

**public** **class** Ni {

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

//要求利用匿名内部类调用接口中的两个方法

**Inter i = new Inter(){**

**public void show(int a, int b) {**

**System.*out*.println(a+" \*\*\* "+b);**

**}**

**public void func() {**

**System.*out*.println("ddddddd");**

**}**

**};**

**i.show(3,4);**

**i.func();**

}

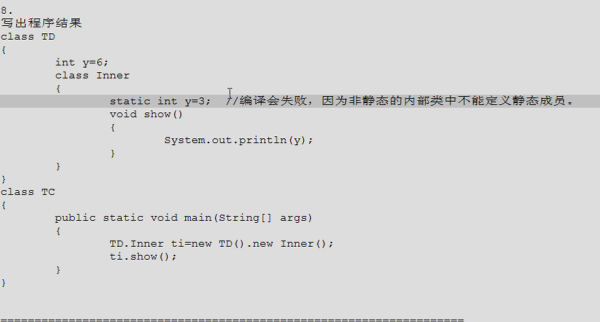
}

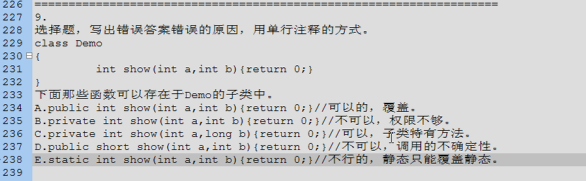
**interface** Inter{

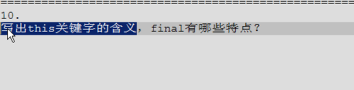
**void** show(**int** a,**int** b);

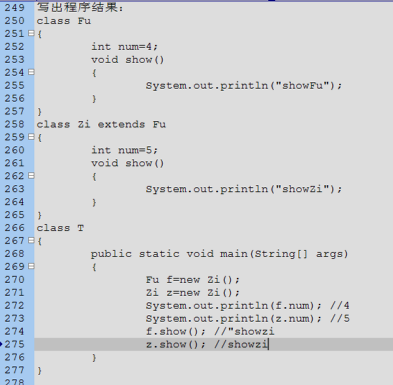
**void** func();

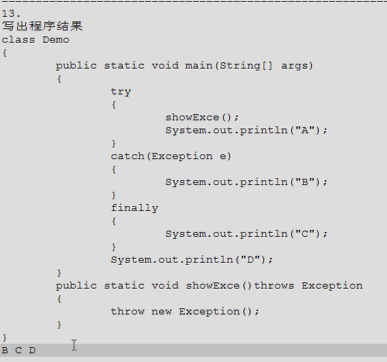
}

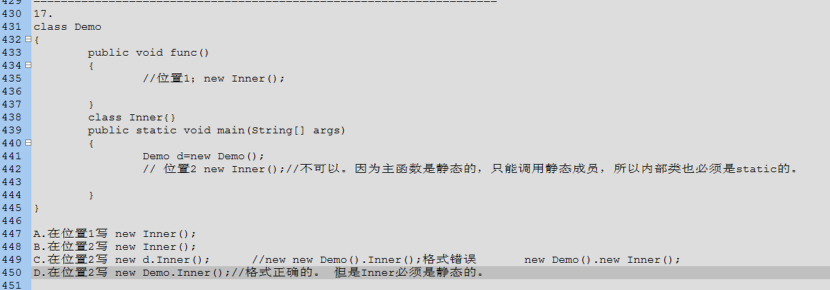


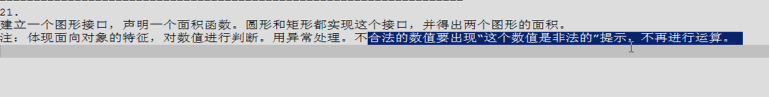
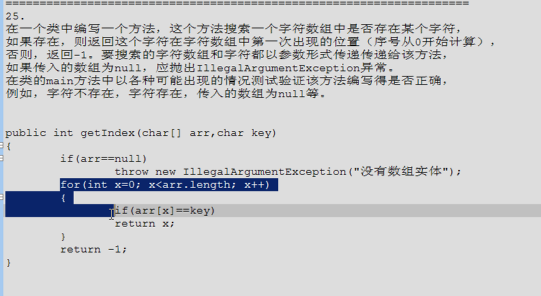










1. 编程
2. 

**考点：如果有三个变量，有其中两个变量相加，赋值给第三个变量，必须左边的变量的范围能够包含左边的变量的最大值的之和， 否则编译失败。**

**例如： byte a = 3；**

**byte a = 4 + 5；//正确，因为右边是常量，判断没有超出字节的范围，所以合法**

**byte b = 4；**

**byte c = 5；**

**a = b + c；//报错，因为右边是变量，编译时不判断变量的大小值，只会提示可能超出字节范围，所以出错。 解决办法，要么对右边强制类型转换为byte，或者直接利用一个int变量接收即可。**

**int a = 3；**

**int b = 4；**

**int c = 5；**

**a = b + c；// 合法，为什么？因为int是默认的整数类型，如果两个int变量相加超过int范围，会有自动转换机制，即把超出范围舍弃，保留低位，但是会出现正负不正确问题。**

**此种情况，int类型、long类型、float类型 、 double类型都可以通过，是合法的。但是对于byte类型、short类型都是不合法的 。**

1. Dd
2. Sss
3. Ss
4. S
5. **考点：package** sdk.November25.am;

**public** **class** SwapString {

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

Swap sp1 = **new** Swap();

System.***out***.println(sp1.strarr[0] + " " +sp1.strarr[1]);//结果：aaaa cccc

sp1.swap(sp1.s1,sp1.s2,sp1.strarr);

System.***out***.println(sp1.strarr[0] + " " +sp1.strarr[1]);//结果：ffff gggg

System.***out***.println(sp1.s1+sp1.s2);//结果为没有交换的字符串：djkjkabdcgde

**//在交换方法中实现了交换，但是调用后并没有改变对象的字符串引用的改变，显然字符串String如同int一样。这样交换不能达到目的**

**//而对于数组来说，虽然也是在方法中建立了局部引用，但是局部引用始终与对象的引用指向同一个数组，改变数组中的任何一个元素的值，都会引起对象中数组的改变**

**//如果欲想对于字符串的内容进行修改，是不可能的，因为字符串一旦建立不可修改，如s1 = “ddddd”;本质是使s1指向了这个新的字符串，原来的字符串并未改变，**

**//然而此引用的指向地址并不不能反传回对象中的引用，所以才不能达到目的。**

}

}

**class** Swap{

String s1 = "djkjk";

String s2 = "abdcgde";

String[] strarr = {"aaaa","cccc"};

**public** **void** swap(String s1, String s2, String[] strarr) {

String temp = "";

System.***out***.println(s1 == **this**.s1);//结果为true

temp = s1;//只是引用地址的传递（未创建新的字符串）

s1 = s2;//只是引用地址的传递

System.***out***.println(s1 == **this**.s2);//结果为true

s2 = temp;//只是引用地址的传递

System.***out***.println(s2 == **this**.s1);//结果为true

System.***out***.println(s1+ s2);//输出结果为交换了位置的字符串：abdcgdedjkjk

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

s1 = "dddddd";//欲改变s1的字符串内容，实质是s1自己的指向变了，而原字符串并未改变，达不到目的

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

strarr[0] = "ffff";

strarr[1] = "gggg"; //利用传递数组名，可以实现对数组元素的赋新值

//\*\*\*\*\*\*\*\*利用字符串交换数组\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

String temp1;

temp = strarr[0];

strarr[0] = strarr[1];

strarr[1] = temp; //可以实现数组元素的交换

} }

1. 练习题
2. 字符串数组排序：

① 字符串大小比较利用compareTo方法；

②排序利用选择或冒泡法；

③**自定义一个交换两个字符串的方法，注意参数一定是数组名和需要交换的索引号，不能是数组元素（字符串），因为字符串作为参数，只是一种值传递的方式，并没有改变原来的字符串。实参字符串把内容传给形参字符串，相当于在交换方法中创建了新的字符串，所以没有改变实参的字符串的顺序。**

**package** sdk.November25.am;

**public** **class** StringSort {

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

String[] str = {"jgdgds","djsj","sdjkhjkg","abc","aad","z","zzzz"};

*printStringArray*(str);

*sortStringArray*(str);

*printStringArray*(str);

}

**private** **static** **void** sortStringArray(String[] str) {

**for**(**int** i = 0;i < str.length - 1;i ++)

{

**for**(**int** j = 0;j < str.length - 1 - i;j++)

{

**if**(str[j].compareTo(str[j+1]) > 0)//字符串的比较大小利用compareTo方法

// swapString(str[j],str[j+1]);//仍然是以值传递的方式传进去的，所以必须是传递数组名才可以

*swapString*(str,j,j+1);

}

}

}

**private** **static** **void** swapString(String[] str,**int** i,**int** j) { **//一定是把数组名传进//去private static void swapString(String string, String string2) {**

**//}这中字符串作为参数的方法是错误的，传进去后，就变成了局部变量**

String temp = **null**;

temp = str[i];

str[i] = str[j];

str[j] = temp;

}

**private** **static** **void** printStringArray(String[] str) {

**for**(**int** i = 0; i < str.length; i++) {

**if**(i != str.length-1)

System.***out***.print(str[i] + " ,");

**else**

System.***out***.print(str[i]);

}

System.***out***.println();

}

}

1. 面试题：

public class Example{   
String str=new String("good");   
char[] ch={'a','b','c'};   
public static void main(String args[]){   
 Example ex=new Example();   
 ex.change(ex.str,ex.ch);   
 System.out.print(ex.str+" and ");   
 Sytem.out.print(ex.ch);   
}   
public void change(String str,char ch[]){   
str="test ok";   
ch[0]='g';   
}   
}  
请问这个程序为什么输出是什么？？？？？？

答案：good and gbc

解析：这里 str是一个字符串的引用，ch是个数组引用，都是引用类型；但是字符串比较特殊，字符串一旦创建，其字符串的内容就不会改变，改变的是字符串引用的值，即此字符串引用重新指向一个新的字符串，但是并不能将形参本身发生的变化返给对象的此实参，而数组始终没有改变形参本身的值，所以始终与对象中的实参保持一致。在change方法中，str 和 ch都是局部引用变量，str一开始指向类中的成员变量字符串good的地址，但是str = “test ok ”，这句话就是在创建一个新的字符串，然后把地址赋给了str，此时局部引用变量指向了test ok 的地址，但是出了此方法后，对象中的str 还是指向good；因为局部变量的名字可以是任意的，如str1；而对于数组来说，把类成员数组的地址传给了ch局部引用变量，此时局部的引用和类的成员引用变量指向同一数组，且都可以对此数组进行修改。

从上面可以看出，对于引用来说，传递的都是地址，但是字符串一旦创建不可改变（字符串不可改变，但是字符串引用可以改变），如果字符串不同，就是说明创建了新的字符串，所以字符串的引用（即字符串形参）就变了；但是对于数组而言，数组引用没有发生改变，始终指向同一数组。

变式训练1.只改变方法小括号里形参名字，此时方法体中的str就是类的引用了，不再是局部引用，所以结果就不一样了。

2.如果小括号和方法体中的str都变为str11，结果还是不变。

3.对于ch数组名，变与不变，变一个或者两个都一样，都会修改数组元素。

package sdk.November25.am;

public class Example11 {

String str = new String("good");

char[] ch = { 'a', 'b', 'c' };

public static void main(String args[]) {

Example11 ex = new Example11();

ex.change(ex.str, ex.ch);

System.out.print(ex.str + " and ");

System.out.print(ex.ch);

}

public void change(String str11, char ch11[]) {

//str11 = "test ok";//此时是局部引用变量，方法一结束，此变量就会消失了，并不会改变对象的成员变量str的指向

str = "test ok";//此时直接调用的就是对象的成员变量了

ch11[0] = 'g';//无论数组名字如何改变，因为操作数组的地址没有改变，所以操作的始终是同一个数组

}

}