**中软国际考题-笔试**

1. 单选题

1、下列说法错误的是（A）

1. 当方法参数为final时，式可以改变该参数值的;
2. final类中的方法默认是final的();
3. 用final修饰的成员变量表示常量，值一旦给定就无法改变;
4. inal方法不能被子类的方法覆盖，但可以被继承;

2、下列描述错误的是（D）

A.一旦声明的final变量被赋值后，就不能改变该变量的值。

B.被final定义的方法不能被重写

C.一个定义为private的方法隐式的被指定为final类型

D.定义为final的类不能被继承，能再派生子类

3、下面有关java代码安全性的叙述哪些是对的？（ D ）

1. 字节码校验器加载查询执行需要的所有的类。
2. 运行时解释器执行代码。
3. 在运行时，字节码被加载，验证然后在解释器里面去运行。
4. 类加载器通过分离本机文件系统的类和从网络导入的类增加安全性。

4、下列语句关于final的叙述正确的是（D）

A. final可以修饰变量，被final修饰的变量被赋初始值之后，能对它重新赋值。

B. final可以修饰方法，被final修饰的方法能被重写。

C. final可以修饰类，被final修饰的类能够被继承。

D. 用来修饰方法参数，表示在变量的生存期中它的值不能被改变

5、Person类和Test类的代码如下所示，则代码中的错误语句是（ C ）。

public class Person {

public String name;

public Person(String name) {

this.name = name;

}

}

public class Test {

public static void main(String[] args) {

final Person person = new Person("欧欧");

person.name = "美美";

person = new Person("亚亚");

}

}

A、 final Person person = new Person("欧欧")；

B、person.name = "美美"；

C、 person = new Person("亚亚")；

D、 没有错误

6、java语言中，在类的定义时使用final关键字修饰，说明这个类（A）

A. 不能被继承

B. 在子类方法中不能被调用

C. 能被别的程序自由调用

D. 不能被子类的方法覆盖

7、下面的是关于类及其修饰符的一些描述，不正确的是 （C）

A．abstract类只能用来派生子类，不能用来创建abstract类的对象

B．abstract不能与final同时修饰一个类

C．final类不但可以用来派生子类，也可以用来创建final类的对象

D．abstract方法必须在abstract类中声明，但abstract类定义中可以没有abstract方法

8、下列描述中正确的是（B）

A. Final关键字可以用来修饰变量，方法和对象

B. Final修饰一个类时，表明这个类不能被继承

C. Final表示修饰符常量的

D. Final修饰的方法代表可以重写

9、下列关于常量描述错误的是（B）

A. 被final修饰的类没有子类

B. Final修饰的类，其属性变量也无法改变

C. Final修饰的方法可以被继承

D. Final修饰的方法无法被覆盖。

10、java中常量描述正确的是（D）

A.final关键词可以用来修饰抽象类

B.final关键词可以用来修饰接口

C. Java常量只存在于属性变量中

D.Final修饰的变量赋值后无法再次改变

11、下列哪个关键字修饰的类不能被继承（A）

A.final B.private C.abstract D.static

12、下列语句中正确的是（B）

byte b1 = 0,b2 = 0,b3 = 0,b6 = 0,b8 = 0;

final byte b4=4,b5=6,b7=9;

public void test()

{

b3=(b1+b2); /\*语句1\*/

b6=b4+b5; /\*语句2\*/

b8=(b1+b4); /\*语句3\*/

b7=(b2+b5); /\*语句4\*/

}

A.语句1 B.语句2 C.语句3 D.语句4

13. 下列代码的编译运行结果是：CD

public class Test

{

public static void main(String[] args)

{

String str1 = "helloworld";

String str2 = "hello";

final String str3 = "hello";

String str4 = str2 + "world";

String str5 = str3 + "world";

System.out.println(str1 == str4);

System.out.println(str1 == str5);

}

}

14. 下列代码的编译运行结果是：A

Integer i1 = 97;

Integer i2 = 97;

System.out.println(i1 == i2);

System.out.println(i1.equals(i2));

Integer i3 = 197;

Integer i4 = 197;

System.out.println(i3 == i4);

System.out.println(i3.equals(i4));

A．true true false true

B．false true false true

C．false false true false

D．true fasle true false

A.编译失败 B.运行异常 C.false D.true

15. 下列关于Error类和Exception类描述正确的是：A

A．Error类和Exception类都是Throwable类的子类

B．Error类是一个final类，而Exception类是个非final类

C．Exception类是个final类，而Error是个非final类

D．Error类和Exception类都实现了Throwable接口

16.以下说法错误的是（ B ）。（单选）

A.final修饰的类不能作为父类

B.final修饰的类不能作为子类

C.final修饰的方法在继承关系中不能被重写

final修饰的属性有且只有一次赋值的机会

1. 多选题
2. 下列说正确的是？（B，C）
3. final类中的方法可以不是final的;
4. final类不能被继承，没有子类
5. final方法不能被子类的方法覆盖，但可以被继承;
6. 接口可以被final修饰;

2、下列选项正确的是？（C、D）

1. final类不能被继承，没有子类，final类中的方法默认是final的。
2. final方法不能被子类的方法覆盖，但可以被继承。
3. final成员变量表示常量，只能被赋值一次，赋值后可以改变。
4. final能用于修饰构造方法。
5. 用final修饰的成员变量表示常量，值一旦给定就无法改变

当函数参数为final类型时，你可以读取使用该参数，但是无法改变该参数的值。

3、被final修饰的变量必须显示的初始化，初始化可以以哪些方式：（B、C、D）

A. 使用new方法重新定义

B. 定义时初始化;

C. 在构造器中设置值

D. 在非静态块中为final实例变量设置值;

4、final关键字说法正确的是？（A、C）

A． 如果修饰变量，则一旦赋值，就等同一个常量

B．如果修饰类，则该类只能被一个子类继承

C. 如果修饰方法，则该方法不能再子类中被覆盖

D．如果修饰方法，则该方法所在类不能被继承

1. 以下说法正确的是（A、B，D）
2. 使用final关键字，如果编译器能够在编译阶段确定某变量的值，那么编译器就会把该变量当做编译期常量来使用。如果需要在运行时确定，那么编译器就不会优化相关代码。
3. 接口中的变量都是public static final 的。;
4. final与finally的用法大致相同
5. 将类、方法、变量声明为final能够提高性能，这样JVM就有机会进行估计，然后优化;

6、下面的是关于类及其修饰符的一些描述，不正确的是(选择3项) （ABD）

A．abstract类只能用来派生子类，不能用来创建abstract类的对象

B．abstract不能与final同时修饰一个类

C．final类不但可以用来派生子类，也可以用来创建final类的对象

D．abstract方法必须在abstract类中声明，但abstract类定义中可以没有abstract方法

7、以下关于final关键字说法错误的是？（A、C）

1. final是java中的修饰符，可以修饰类、接口、抽象类、方法和属性
2. final修饰的类肯定不能被继承
3. final修饰的方法不能被重载
4. final修饰的变量不允许被再次赋值

8、final关键字可以在以下哪个位置前面使用 （ABCD）

A.类前面 B.数据成员前面 C.方法成员前面 D.局部变量前面

9、下面哪些项编译会出错？（A、B）

1. float[]=new float(3) B. float f2[]=new float[]; C. float[]f1=new float[3]
2. float f3[]=new float[3]; E. float f5[]={1.0f,2.0f,2.0f}
3. float f4[]=new float[]{1.0f,2.0f,3.0f}

10.在Java中，关于final关键字的说法正确的是（ AB ）。（多选）

A．如果修饰变量，则一旦赋了值，就等同一个常量

B．如果修饰类，则该类只能被一个子类继承

C．如果修饰方法，则该方法不能在子类中被覆盖

D．如果修饰方法，则该方法所在的类不能被继承

1. 填空题
2. 一个类被修饰为final那此类\_\_\_\_\_不能被继承 \_\_\_\_\_ 一个方法被修饰成final那此方法,\_\_\_\_ 不能被重写 \_\_\_\_\_
3. 给final变量初值后，值\_\_\_\_\_\_\_\_\_\_再改变 （不能）
4. final是一个关键字，可以用于\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_。(修饰类，成员变量，成员方法)
5. Final作用范围 、 、 (变量、方法、类)
6. 被final修饰的变量必须显示的初始化，初始化可以以三种方式：\_\_\_\_\_\_\_\_\_\_。(1）定义时初始化，2）在构造器中设置值，3）在非静态块中为final实例变量设置值)
7. 定义常量格式：\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ COUNT2=10.2f; 。(final float )
8. 定义类时，前面使用final关键字，则该类不能 被继承
9. 在数据成员前添加final关键字，则该方法不能 被重写
10. final修饰的类，类中的所有成员方法都被隐式地指定为\_\_final\_\_方法。
11. **修饰变量，被final修饰的变量必须要初始化，赋初值后不能再重新\_\_赋值。\_\_**
12. 判断题
13. 类中所有的private方法都隐式的指定为是final的（T）
14. 对private方法添加final修饰词会使该方法更加安全而不必担心被子类串改（F）（可以对private方法添加final修饰词，但是没有任何意义（java核心思想第四版上说的））
15. final修饰词不能用来修饰属性变量（F）（可以修饰）
16. final变量定义的时候，可以先声明，而不给初值。（ T ）
17. 无论什么情况，编译器都确保空白final在使用之前必须被初始化。（T）
18. 当函数参数为final类型时，你可以读取使用该参数，但是无法改变该参数的值。（T）
19. 被final修饰的方法不能被重写。（ T ）
20. Final修饰成员变量时，必须要赋初始值，而且是初始化多次。（ F ）
21. 当用final去修饰一个类的时候，表示这个类不能被继承。（ T ）
22. final在Java中是一个保留的关键字，可以声明成员变量、方法、类以及本地变量。（ T ）
23. final关键字可以用来修饰类、方法，但是不能修饰变量。（ F ）
24. final成员变量表示常量，只能被赋值一次，赋值后值不再改变。（ T ）
25. 如果方法用final修饰，则意味着该方法能被子类重写。（ F）
26. 最终方法只能存在于最终类中。( F )
27. 使用final修饰的变量将变成常量，其中不能再被改变；使用final修饰的方法将无法被子类重载；使用final修饰的类将无法再被其他类继承。（ T ）
28. final修饰的类，类中的所有成员方法都被隐式地指定为final方法。（T）
29. final修饰变量指的是：这个变量被初始化后便不可改变，这里不可改变的意思对基本类型来说是其值不可变，而对于对象变量来说其引用不可变，即不能再指向其他的对象（T）
30. final不能用于修饰类。（F）
31. Final修饰后无法添加static修饰词。（ N ）
32. final用在类的方法成员前面，则该方法不能被重载（错误）
33. final可以和static一起使用（正确）
34. fianl修饰的方法不可以被覆盖但是可以被重载。（ T ）
35. final修饰的类可以被继承。（F）

五、简答题

# 1、使用final方法的原因？

答案：把方法锁定，以防任何继承类修改它的含义。这是出于设计的考虑：想要确保在继承中使方法行为保持不变，并且不会被覆盖。（还有一个效率问题，在现在的java版本中已不需要使用final进行效率优化）

2、final用于修饰变量、方法和类分别有什么作用（5分）

答案：在Java中，final关键字有最终的，不可修改的含义。final关键字有三种用途，可以分别应用于变量、成员方法和类。如果将某个变量修饰为final，那么该变量就成为常量，常量在声明时必须初始化，声明之后不能对其进行二次赋值。

如果将某个成员方法修饰为final，则意味着该方法不能被子类覆盖，这就和抽象方法必须由子类实现的规定互相矛盾，因此，final和abstract不能同时修饰一个方法。

如果将某个类修饰为final，则说明该类无法被继承

3、final类的特点（5分）

答案：它修饰的类不能被继承。

它修饰的成员变量是一个常量。

它修饰的成员方法是不能被子类重写的。

4、final和private的区别（5分）

答案：final修饰的类可以访问；

private不可以修饰外部类，但可以修饰内部类（其实把外部类私有化是没有意义的）。

final修饰的方法不可以被子类重写；

private修饰的方法表面上看是可以被子类重写的，其实不可以，子类是看不到父类的私有方法的。

final修饰的变量只能在显示初始化或者构造函数初始化的时候赋值一次，以后不允许更改；

private修饰的变量，也不允许直接被子类或一个包中的其它类访问或修改，但是他可以通过set和get方法对其改值和取值。

5、Final修饰成员变量时，需注意些什么（5分）

答案：a. 必须初始化值。

b. 被fianl修饰的成员变量赋值，有两种方式：1、直接赋值 2、全部在构造方法中赋初值。

c. 如果修饰的成员变量是基本类型，则表示这个变量的值不能改变。

d. 如果修饰的成员变量是一个引用类型，则是说这个引用的地址的值不能修改，但是这个引用所指向的对象里面的内容还是可以改变的。

6、final, finally, finalize的区别（5分）

答案：final 用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。

finally是异常处理语句结构的一部分，表示总是执行。

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。

7、被final修饰的变量必须被初始化。初始化的方式有哪几种（5分）

答案：1、在定义的时候初始化

2、final变量可以在初始化块中初始化，不可以在静态初始化块中初始化。

3、静态final变量可以在静态初始化块中初始化，不可以在初始化块中初始化。

4、final变量还可以在类的构造器中初始化，但是静态final变量不可以。

8、是否可以继承String类?

final表示最终的，用在类修饰符上表示此类不能被其他类扩展。string是final的，不可以继承。

9、 Java中，请说明final关键字的使用

1.用来修饰数据，包括成员变量和局部变量，该变量只能被赋值一次且它的值无法被改变。对于成员变量来讲，我们必须在声明时或者构造方法中对它赋值；

2.用来修饰方法参数，表示在变量的生存期中它的值不能被改变；

3.修饰方法，表示该方法无法被重写；

4.修饰类，表示该类无法被继承。

10、简述final的作用（5分）

答案final根据修饰位置的不同作用也不相同，针对三种情况：

　　1）修饰变量，被final修饰的变量必须要初始化，赋初值后不能再重新赋值。

　　　注意：局部变量不在我们讨论的范畴，因为局部变量本身就有作用范围，不使用private、public等词修饰。

　　2）修饰方法，被final修饰的方法代表不能重写。

　　3）修饰类，被final修饰的类，不能够被继承。

　　注意：final修饰的类，类中的所有成员方法都被隐式地指定为final方法。

　　2.1 final修饰变量

　　　被final修饰的变量必须显示的初始化，初始化可以以三种方式：1）定义时初始化，2）在构造器中设置值，3）在非静态块中为final实例变量设置值。

　　　final修饰变量指的是：这个变量被初始化后便不可改变，这里不可改变的意思对基本类型来说是其值不可变，而对于对象变量来说其引用不可变，即不能再指向其他的对象。

如果final修饰的变量是对象类型，那么不可更改指的是该变量不可以再指向别的对象，但是对象的值时可以更改的

11、使用final修饰属性后，属性有什么特征？（5分）

答案：fianl修饰属性后，属性就成为一个常量。常量只要被赋值，就不能被改变。

12、final类与final方法有什么作用？

答案：final类是不能够被继承的类，称为终极类，如String类就是final类，不能有子类。final修饰方法后，是终极方法，不能被子类覆盖，但是可以被子类继承使用。

* 13、请简述final可以修饰哪几种java组成元素，及其各自特点（5分）

答案： A、修饰类(class)。

     1、该类不能被继承。

     2、类中的方法不会被覆盖，因此默认都是final的。

     3、用途：设计类时，如果该类不需要有子类，不必要被扩展，类的实现细节不允许被改变，那么就设计成final类

B、修饰方法(method)

     1、该方法可以被继承，但是不能被覆盖。

     2、用途：一个类不允许子类覆盖该方法，则用final来修饰

     3、好处：可以防止继承它的子类修改该方法的意义和实现；更为高效，编译器在遇到调用fianal方法转入内嵌机制，提高了执行效率。

     4、注意：父类中的private成员方法不能被子类覆盖，因此，private方法默认是final型的(可以查看编译后的class文件)

C、修饰变量(variable)

     1、用final修饰后变为常量。包括静态变量、实例变量和局部变量这三种。

     2、特点：可以先声明，不给初值，这种叫做final空白。但是使用前必须被初始化。一旦被赋值，将不能再被改变。

D、修饰参数(arguments)

     1、用final修饰参数时，可以读取该参数，但是不能对其作出修改

13、final修饰的类、方法、变量有什么特点：

答案：

final类不能被继承，没有子类，final类中的方法默认是final的。

final方法不能被子类的方法覆盖，但可以被继承。

final成员变量表示常量，只能被赋值一次，赋值后值不再改变。

final不能用于修饰构造方法。

14. 列举常见的final类

答案：

包装类：Boolean，Character，Short，Integer，Long，Float，Double，Byte，Void

字符串类：String，StringBuilder，StringBuffer

系统类：Class，System，RuntimePermission，Compiler

数学类：Math，StrictMath

其他：Character.UnicodeBlock，ProcessBuilder，StackTraceElement

15. String、StringBufer、StringBuilder的区别

答案：

String是内容可变的，而StringBuffer和StringBuilder都是内容可变的

StringBuffer是同步的，数据安全，效率低；而StringBuilder是不同步的，数据不安全，效率高。

六、程序实现题

1、单利模式类（使用final，让引用的指向不可变，不需要判断实例是否为空）

|  |
| --- |
| **public** **class** Book {  **private** **static** **final** Book ***book*** = **new** Book();  **private** Book() {}  **public** **static** Book getInstance() {  **return** ***book***;  }  } |

2、使用final来定义类，方法和变量（10分）

|  |
| --- |
| Public final class Test{  } private final String S="final实例变量S";  Public final void test(){  } |

3、以下代码输出结果为？

（10分）

|  |
| --- |
| public class Test  {  public static void main(String[] args)  {  String str1 = "helloworld";  String str2 = "hello";  final String str3 = "hello";  String str4 = str2 + "world";  String str5 = str3 + "world";  System.out.println(str1 == str4);  System.out.println(str1 == str5);  }  } |

4、用代码说明final修饰的类不能被继承

（10分）

|  |
| --- |
| final class Animal //final修饰  {  public Animal()  {  System.out.println("动物类的无参构造");  }  }  class Cat extends Animal //报错，原因final修饰的额类不能被继承  {  }  public class Test  {  public static void main(String[] args)  {  System.out.println("hello");  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*分隔符\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  class Animal //去掉final  {  public Animal()  {  System.out.println("动物类的无参构造");  }  }  class Cat extends Animal //正确  {  }  public class Test  {  public static void main(String[] args)  {  System.out.println("hello");  }  } |

5、实现不同符合PCI规范的适配器

需求说明：PCI是一种规范，所有实现了该规范的适配器，必如显卡、声卡、网卡都可以安装到PCI插槽上并工作。模拟实现该功能。

实现思路及关键代码

1) 定义PCI接口，具有传送数据send()方法

2) 定义显卡VideaCard类，实现该接口

3) 定义声卡AudioCard类，实现PCI接口

4) 定义网卡NetCard类，实现PCI接口

5) 定义测试类，让显卡、声卡、网卡发送数据

（10分）

|  |
| --- |
| class a  {  /\*\*  \*  \* PCI接口  \*  \* @author Administrator  \*  \*  \*  \*/  public interface PCI  {  /\*\*  \*  \* 传输数据  \*  \*/  public void send();  }  public class VideoCard implements PCI  {  @Override  public void send()  {  System.out.println("使用PCI接口传输视频数据");  }  }  public class SoundCard implements PCI  {  @Override  public void send()  {  System.out.println("使用PCI接口传输声音数据");  }  }  public class NetCard implements PCI  {  @Override  public void send()  {  System.out.println("使用PCI接口传输网络数据");  }  }  public class Test  {  public static void main(String[] args)  {  PCI p1 = new VideoCard();  p1.send();  PCI p2 = new SoundCard();  p2.send();  PCI p3 = new NetCard();  p3.send();  }  }  } |

6、定义常量： 定义一个常量类及常量属性 并用main方法实例化访问

（10分）

|  |
| --- |
| public final class Test10 {  public final int index=56;  public static void main(String[] args) {  // TODO Auto-generated method stub  Test10 test=new Test10();    System.out.println(test.index);  }  } |

7、定义一个圆类。

答案：

class Circle{

static final float PI;

float radius;

//对静态fainl成员PI初始化

static{

PI = 3.1415926f;

}

//获取面积

public float getArea(){

return PI \* radius \* radius;

}

//获取周长

public float getGirth(){

return 2 \* PI \*radius;

}

}

8、单例模式：

public class Singleton {  
    private static Singleton uniqueInstance;  
    private final String name;  
    private Singleton(String name){  
        this.name = name;  
    }  
    public static Singleton getInstance(String name){  
        if(uniqueInstance == null){  
            synchronized (Singleton.class){  
                if(uniqueInstance == null){  
                    uniqueInstance = new Singleton(name);  
                }  
            }  
        }  
        return uniqueInstance;  
    }  
}