**中软国际考题-笔试**

1. 单选题

1、下列说法正确的的是（D）

1. 所有的内部类都能像普通类一样实现多个接口。
2. 任何情况下都不可以在内部类中实现其外围接口
3. 所有的内部类对象与其外围类对象之间必然要有联系
4. 不能从静态内部类的对象中访问非静态的外围类对象。

2、局部内部类可以用哪些修饰符修饰?（B）

A.public

B.final

C.private

D.static

3、下列不属于内部类的是（B）

A 、常规内部类

B 、final内部类

C 、局部内部类

D 、匿名内部类。

4、编译运行以下程序后，关于输出结果的说明正确的是 （ C）：

public class Conditional{

public static void main(String args[ ]){

int x=4;

System.out.println(“value is “+ ((x>4) ? 99.9 :9));

}

}

A、 输出结果为：value is 99.99

B、 输出结果为：value is 9

C、 输出结果为：value is 9.0

D、 编译错误

5、编译MyClass.java之后，得到了三个字节码文件：MyClass.class , MyClasslittle$.class，MyClass$1.class。这表明（ C ）

A) MyClass类中的两个方法：little和1

B) MyClass.Java中有三个类：MyClass、little和1

C) MyClass类中有两个内部类：一个是命名的little，另一个是匿名的1

D) MyClass、little和1，这三者没什么关系

6、什么是内部类以下说法最准确的是（A）

1. 在Java中，可以将一个类定义在另一个类里面或者一个方法里面，这 样 的类称为内部类。
2. 在Java中，可以将一个类声明在另一个类里面，这 样 的类称为内部类。
3. 在Java中，可以将一个类定义在另一个类的方法里面，这 样 的类称为内部类。

在Java中，可以将一个类写入到在另一个类里面或者一个方法里面，这 样 的类称为内部类。

7、下列代码能否正常运行，如果能够正常运行，输出结果是什么 (D)

public class TestClass {

public static void main(String[] args) {

int num1=5;

int num2=5;

class InnerClass{

public int add(){

return num1+num2;

}

};

InnerClass in=new InnerClass();

System.out.println(in.add());

}

}

A. 输出10

B. 输出0

C. 运行时输出异常信息

D. 编译不能通过

8、对于内部类，以下说法错误的是 (C)

A．匿名内部类可以实现接口或继承其他类，但不能同时即实现接口又继承类

B．匿名内部类不能有任何明确的构造器

C．内部类可以定义在外部类类体中，也可以定义在外部类的方法体中，和外部类不同，内部类均能使用访问修饰符，并能使用static修饰

D．在Java中，对内部类的嵌套层次没有限制

9、对于以下代码，请问插入哪个语句可以访问到内部类InsideOne? (D)

public class Example {

public static void main(String[] args) {

EnclosingOne eo=new EnclosingOne();

//插入语句处

}

}

class EnclosingOne{

public class InsideOne{

}

}

A．InsideOne ei=new eo.InsideOne();

B．InsideOne ei=EnclosingOne.new InsideOne();

C．InsideOne ei=new EnclosingOne.InsideOne();

D．EnclosingOne.InsideOne ei=eo.new InsideOne();

10、下列表述正确的是（B）

A.一个java文件 有多个public公共类

B.一个java文件 有多个公共类的子类

C.一个java文件 有多个子类相互影响

D.一个java文件 多个子类由全局类同一管理

11、下列关于内部类描述错误的是（C）

1. 内部类的类名只能在定义它的类或程序段中或在表达式内部匿名使用
2. 内部类可以使用它所在类的静态成员变量和实例成员变量
3. 内部类不可以用abstract修饰符定义为抽象类
4. 内部类可作为其他类的成员，而且可访问它所在类的成员。

12、Java内部类描述正确的是（D）

A.内部类与外部类毫无联系

B.内部类无法调用外部类方法

C.内部类无法访问外部类属性

D.内部类可以访问外部类属性及方法

13、有关匿名内部类描述错误的有（）

A匿名内部类没有显式构造器

B.匿名内部类可以实现接口

C.匿名内部类可以继承非final类

D.匿名内部类可以同时实现接口和继承非final类

14、下面的程序插入的代码能正确运行的是(B)

class Boo{

Boo(String s){ }

Boo(){ }

}

class Bar extends Boo{

Bar(){ }

Bar(String s){super(s);}

void zoo(){

//此处插入代码

}

}

A.Boo f = new Boo(24){ }; B.Boo f = new Bar( ){ }; C.Boo f = new Boo( ).Bar{String s;}; D.Bar f = new Boo(String s){ };

15.下面程序运行的结果是（D）

1. public class HorseTest {

2. public static void main(String[ ] args) {

3. class Horse {

4. public String name;

5. public Horse(String s) {

6. name = s;

7. }

8. }

9. Object obj = new Horse(“Zippo”);

10. System.out.println(obj.name);

11. }

12. }

A.出现运行时异常在第10行 B.Zippo C.编译失败因为在第3行出现错误 D.编译失败因为在第10行出现错误

16.下列内部类的说法不正确的是（C）

A.内部类的类名只能在定义它的类或程序中或在表达式内部使用

B.内部类可以使用它所在类的静态成员变量和实例成员变量

C.内部类不可以用abstract修饰符定义为抽象类

D.内部类可作为其他类的成员，而且可访问它所在类的成员

1. 多选题
2. 下列说法正确的是？（A、B）
3. 静态内部类不需要对外部类对象的引用;
4. 不是所有的内部类对象与其外围类对象之间都会有联系，比如静态内部类
5. 匿名类不一定是内部类;
6. 嵌套类不一定是内部类;
7. 下面哪些项编译会出错？（A、B）

A.匿名内部类不能是抽象类，必须实现它的抽象父类或者接口里包含的所有抽象方法

B.匿名内部类没有类名，所以匿名内部类不能定义构造方法

C.匿名内部类可以实现多个接口，或者继承一个父类

D.匿名内部类可以直接访问外部类的所有局部变量

1. 内部类可以拥有哪些权限（A、B、C、D）
2. private访问权限
3. protected访问权限
4. public访问权限
5. 包访问权限

4、以下选项中关于匿名内部类的说法正确的是（ BC ）。（选择二项）

A.匿名内部类可以实现多个接口，或者继承一个父类

B.匿名内部类不能是抽象类，必须实现它的抽象父类或者接口里包含的所有抽象方法

C.匿名内部类没有类名，所以匿名内部类不能定义构造方法

D.匿名内部类可以直接访问外部类的所有局部变量

5.下列内部类的说法正确的是（ABD）。

A.内部类的类名只能在定义它的类或程序中或在表达式内部使用

B.内部类可以使用它所在类的静态成员变量和实例成员变量

C.内部类不可以用abstract修饰符定义为抽象类

D.内部类可作为其他类的成员，而且可访问它所在类的成员

6、内部类是在一个类内嵌套定义的类，其特点描述正确的是( BCD )：

A. 只能在定义它的类或程序段中或表达式内匿名使用，外部使用时必须给出类的全名

B. 可以使用它所在类的静态成员变量或实例成员变量，但不可以使用所在类中的局部变量

C. 可以使用其他类的成员，而且可访问它所在类的成员

D. 除static内部类之外，不能在类内声明static成员

7、以下说法中关于静态内部类说法正确的是：（A、B、C、D、E）

A.是 static 修饰的内部类,  
 B.静态内部类不能直接访问外部类的非静态成员，但可以通过 new 外部 类().成员 的方式访问   
 C.如果外部类的静态成员与内部类的成员名称相同，可通过“类名.静态 成员”访问外部类的静态成员；  
 D.如果外部类的静态成员与内部类的成员名称不相同，则可通过“成员 名”直接调用外部类的静态成员  
 E.创建静态内部类的对象时，不需要外部类的对象，可以直接创建 内部 类 对象名 = new 内部类();

8、有关匿名内部类描述正确的有（ABC）

A. 匿名内部类没有显式构造器

B. 匿名内部类可以实现接口

C. 匿名内部类可以继承非final类

1. 匿名内部类可以同时实现接口和继承非final类

9、以下哪些类型的变量可以被一个内部类访问？(ABC)

A． 所有static变量

B．所有final变量

C．所有的实例成员变量

D．只有final静态变量

10、下面哪些项描述不正确？（A、B）

A. 内部类与接口，抽象类各自规定

B. 外部实体类的情况下，内部类不允许规定为抽象类

C. 外部类与内部类class修饰词各自规定

D. 外部类的class修饰词与内部类的class修饰词相互不影响

11. 以下哪些类型的变量可以被一个内部类访问？（A、B、C）

A. 所有static变量 B. 所有final变量 C. 所有的实例成员变量

D. 只有final静态变量

12、局部内部类可以用哪些修饰符修饰？(C D)

A. public

B. private

C. abstract

D. final

1. 填空题
2. \_\_\_\_\_外部）\_\_\_\_\_类可以把内部类声明对象作为\_\_\_\_\_外部类\_\_\_\_\_的成员
3. 内部类分为四种,分别是：分别是、\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_、分别是。(成员内部类，局部内部类)
4. 静态内部类也是定义在另一个类里面的类，只不过在类的前面多了一个关键字\_\_\_\_\_\_\_\_\_\_ 。(static)
5. 内部类 指那些类定义代码被置于其他类定义中的类。
6. 广泛意义上的内部类一般来说包括这四种：\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_\_(成员内部类、局部内部类、匿名内部类和静态内部类)。
7. 内部类可以拥有\_\_\_\_\_\_\_\_\_访问权限、\_\_\_\_\_\_\_\_\_访问权限、\_\_\_\_\_\_\_\_\_访问权限及\_\_\_\_\_\_\_\_\_访问权限。(private、protected、public、包)
8. Java中的四种内部类分别是：\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_。

（成员内部类 、静态内部类 、 局部内部类、匿名内部类）

1. 外部类是不能直接使用\_\_\_\_\_\_\_\_\_\_类的成员和方法(内部)
2. 如果外部类和内部类具有相同的成员变量或方法，内部类默认访问自己的成员变量或方法，如果要访问外部类的成员变量，可以使用\_\_\_\_\_\_\_\_\_\_关键字（this）
3. 内部类分为\_\_\_\_\_\_\_、\_\_\_\_\_\_\_、\_\_\_\_\_\_\_、\_\_\_\_\_\_\_4种(静态内部类，成员内部类，局部内部类，匿名内部类)
4. 判断题
5. 可以在一个方法里面或者在任意的作用域内定义内部类（T）
6. 内部类中的方法任何情况下都不可以调用外嵌类的方法（F）（可以）
7. 内部类的外嵌类的成员变量在内部类中不可能有效（F）（仍然有效）
8. 内部类（也叫成员内部类）可以有4种访问权限。（ T ）
9. 匿名内部类适合创建那种只需要使用一次的类，它可以实现一个或多个接口，或者继承一个父类。（  F  ）
10. 内部类作为外部类成员，权限修饰符和其他成员一样，可声明为private、默认、protected或public。(  T  )
11. 静态内部类是不依赖于外部类的。（ T ）
12. 对于成员内部类，必须先产生外部类的实例化对象，才能产生内部类的实例化对象。（T）
13. 匿名内部类是唯一一种没有构造器的类。（T）
14. 内部类作为外部类成员，权限修饰符和其他成员一样，可声明为private、默认、protected或public。( T )
15. 匿名内部类适合创建那种只需要使用一次的类，它可以实现一个或多个接口，或者继承一个父类。（ F ）
16. 任何类都可以写成匿名内部类的形式（ F ）
17. 匿名内部类其实就是一个没有类名的内部类（ F ）
18. 除 static 内部类之外，不能在类内声明 static 成员 ( T )
19. 内部类是在一个类内嵌套定义的类, 可以使用其他类的成员，而且可访问它所在类的成员( F )
20. 内部类中的类体中可以声明类变量和类方法么？( F )
21. 内部类中的方法也可以调用外嵌类中的方法么？ ( T )
22. 局部内部类是定义在一个方法或者一个作用域里面的类，它和成员内部类的区别在于局部内部类的访问仅限于方法内或者该作用域内。（ T ）
23. 内部类能访问外部的属性和方法 。（ T ）
24. 内部类无法继承或链接一个java文件中的外部类。（ N ）
25. 一个内部类实际上是外部类的子类。（ F ）
26. 匿名内部类其实就是一个没有类名的内部类( T )
27. 方法内部类是指在成员方法中定义的类，它只能在当前方法中被使用。（T）
28. 如果想通过外部类去访问内部类，则需要通过外部类对象去创建内部类对象。（T）
29. 成员内部类可以直接访问其所在的外部类中的成员,不用创建对象。（T）
30. 静态内部类不可以直接访问外围类的非静态数据，而非静态内部类可以直接访问外围类的数据，包括私有数据。（ T）

五、简答题

# 1、什么是内部类？

答案：可以将一个类的定义放在另一个类的定义内部，即一个类中声明另一个类，我们称为内部类。

2、请**Java 1.8 之前为什么方法内部类和匿名内部类访问局部变量和形参时必须加 final？**（5分）

答案：在 Java 1.8 以下因为对于普通局部变量或者形参的作用域是方法内，当方法结束时局部变量或者形参就要随之消失，而其匿名内部类或者方法内部类的生命周期又没结束，匿名内部类或者方法内部类如果想继续使用方法的局部变量就需要一些手段，所以 Java 在编译匿名内部类或者方法内部类时就有一个规定来解决生命周期问题，即如果访问的外部类方法的局部变量值在编译期能确定则直接在匿名内部类或者方法内部类里面创建一个常量拷贝，如果访问的外部类方法的局部变量值无法在编译期确定则通过构造器传参的方式来对拷贝进行初始化赋值。由此说明在匿名内部类或者方法内部类中访问的外部类方法的局部变量或者形参是内部类自己的一份拷贝，和外部类方法的局部变量或者形参不是一份，所以如果在匿名内部类或者方法内部类对变量做修改操作就一定会导致数据不一致性（外部类方法的参数不会跟着被修改，引用类型仅是引用，值修改不存在问题），为了杜绝数据不一致性导致的问题 Java 就要求使用 final 来保证，所以必须是 final 的。在 Java 1.8 开始我们可以不加 final 修饰符了，系统会默认添加，Java 将这个功能称为 Effectively final

3、为什么在Java中需要内部类？（5分）

　　答：1.每个内部类都能独立的继承一个接口的实现，所以无论外部类是否已经继承了某个(接口的)实现，对于内部类都没有影响。内部类使得多继承的解决方案变得完整，

　　2.方便将存在一定逻辑关系的类组织在一起，又可以对外界隐藏。

　　3.方便编写事件驱动程序

　　4.方便编写线程代码

4.什么是匿名内部类？

内部类，存在于另一个类内部的类，而匿名内部类，顾名思义，就是没有名字的内部类。

**5.**内部类可以引用它的包含类的成员吗？有没有什么限制？

答案：

可以。如果不是静态内部类，那没有什么限制！

如果你把静态嵌套类当作内部类的一种特例，那在这种情况下不可以访问外部类的普通成员变量，而只能访问外部类中的静态成员，例如，下面的代码：

class Outer

{

static int x;

static class Inner{

void test(){

syso(x);

}

}

}

6.java的内部类有什么作用和特点

答案：

内部类分为: 成员内部类、静态嵌套类、方法内部类、匿名内部类。  
(1)、内部类仍然是一个独立的类，在编译之后内部类会被编译成独立的.class文件，但是前面冠以外部类的类名和$符号。  
(2)、内部类不能用普通的方式访问。内部类是外部类的一个成员，因此内部类可以自由地访问外部类的成员变量，无论是否是private的。  
(3)、内部类声明成静态的，就不能随便的访问外部类的成员变量，仍然是只能访问外部类的静态成员变量。

7.怎么实现匿名内部类？

匿名内部类的实现非常简单，如有一接口

 Interface A{

public void method();

}

这时你可以写一个如下的语句得到接口A的实现类的对象

A a=new A(){ public void method(){System.out.println("hehe");} };  
而其中的{ public void method(){System.out.println("hehe");} };就是定义了一个匿名内部类

8.为什么内部类调用的外部变量必须是final修饰的？

因为生命周期的原因。方法中的局部变量，方法结束后这个变量就要释放掉，final保证这个变量始终指向一个对象。首先，内部类和外部类其实是处于同一个级别，内部类不会因为定义在方法中就会随着方法的执行完毕而跟随者被销毁。问题就来了，如果外部类的方法中的变量不定义final，那么当外部类方法执行完毕的时候，这个局部变量肯定也就被GC了，然而内部类的某个方法还没有执行完，这个时候他所引用的外部变量已经找不到了。如果定义为final，java会将这个变量复制一份作为成员变量内置于内部类中，这样的话，由于final所修饰的值始终无法改变，所以这个变量所指向的内存区域就不会变。   
为了解决：局部变量的生命周期与局部内部类的对象的生命周期的不一致性问题

9、Java中静态内部类的加载时机

1、外部类初次加载，会初始化静态变量、静态代码块、静态方法，但不会加载内部类和静态内部类。

2、实例化外部类，调用外部类的静态方法、静态变量，则外部类必须先进行加载，但只加载一次。

3、直接调用静态内部类时，外部类不会加载。

10、内部类可以引用它的包含类的成员吗？有没有什么限制？

完全可以。如果不是静态内部类，那没有什么限制。

如果你把静态嵌套类当作内部类的一种特例，那在这种情况下不可以访问外部类的普通成员变量，而只能访问外部类中的静态成员，例如，下面的代码：

class Outer

{

static int x;

static class Inner

{

void test()

{

syso(x);

}

}

}

11、为什么在Java中需要内部类？总结一下主要有以下四点：

　　1.每个内部类都能独立的继承一个接口的实现，所以无论外部类是否已经继承了某个(接口的)实现，对于内部类都没有影响。内部类使得多继承的解决方案变得完整，

　　2.方便将存在一定逻辑关系的类组织在一起，又可以对外界隐藏。

　　3.方便编写事件驱动程序

　　4.方便编写线程代码

　　个人觉得第一点是最重要的原因之一，内部类的存在使得Java的多继承机制变得更加完善。

12.Anonymous Inner Class(匿名内部类)是否可以继承其它类？是否可以实现接口？

可以继承其他类或实现其他接口，在Swing编程和Android开发中常用此方式来实现事件监听和回调。

13.为什么内部类调用的外部变量必须是final修饰的？

因为生命周期的原因。方法中的局部变量，方法结束后这个变量就要释放掉，final保证这个变量始终指向一个对象。首先，内部类和外部类其实是处于同一个级别，内部类不会因为定义在方法中就会随着方法的执行完毕而跟随者被销毁。问题就来了，如果外部类的方法中的变量不定义final，那么当外部类方法执行完毕的时候，这个局部变量肯定也就被GC了，然而内部类的某个方法还没有执行完，这个时候他所引用的外部变量已经找不到了。如果定义为final，java会将这个变量复制一份作为成员变量内置于内部类中，这样的话，由于final所修饰的值始终无法改变，所以这个变量所指向的内存区域就不会变。   
为了解决：局部变量的生命周期与局部内部类的对象的生命周期的不一致性问题

14、请说明java内部类和匿名内部类的区别（5分）

答案：匿名内部类只能使用一次，不区分static和非static。如果用到外部类的变量的话，必须是类变量或者实例变量，就是必须是类定义的变量，或者final的局部变量。匿名内部类如果是继承某个类的话是可以重写那个类的方法的，这个和普通内部类一样。内部类可以是static的或者非static的，static内部类只能包含静态方法和静态类变量，只能访问外部类的静态元素，内部类可以实例化，多次使用。  
所有可以用匿名内部类的地方都可以用内部类替换，但是用匿名内部类编码非常简洁，也容易读懂，除非你需要多次使用那个内部类，不然总是建议使用匿名内部类

15、请简述内部类的产生定义及总结四种内部类（5分）

答案：在Java中，可以将一个类定义在另一个类里面或者一个方法里面，这样的类称为内部类。广泛意义上的内部类一般来说包括这四种：成员内部类、局部内部类、匿名内部类和静态内部类。

16、为什么内部类调用的外部变量必须是final修饰的？

因为生命周期的原因。方法中的局部变量，方法结束后这个变量就要释放掉，final保证这个变量始终指向一个对象。首先，内部类和外部类其实是处于同一个级别，内部类不会因为定义在方法中就会随着方法的执行完毕而跟随者被销毁。问题就来了，如果外部类的方法中的变量不定义final，那么当外部类方法执行完毕的时候，这个局部变量肯定也就被GC了，然而内部类的某个方法还没有执行完，这个时候他所引用的外部变量已经找不到了。如果定义为final，java会将这个变量复制一份作为成员变量内置于内部类中，这样的话，由于final所修饰的值始终无法改变，所以这个变量所指向的内存区域就不会变。   
为了解决：局部变量的生命周期与局部内部类的对象的生命周期的不一致性问题

17、Anonymous Inner Class(匿名内部类)是否可以继承其它类？是否可以实现接口？

可以继承其他类或实现其他接口，在Swing编程和Android开发中常用此方式来实现事件监听和回调。

六、程序实现题

1、编写程序

（1）创建一个类China，China中有内部了LiaoNing; China有属性x,y有方法f;

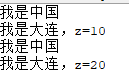
(2)LiaoNing中有属性z,有方法g和方法speak;

(3)China中x,y的初始值都为10，方法f有语句System.out.println("我是中国");，并且调用LiaoNing类的speak方法;

(3)LiaoNing类中的z初始值为10，speek方法中有语句System.out.println("我是大连，z="+z);

(4)LiaoNing类中的f方法中使China类的x,y相加赋值给b.且调用China类中的f方法

（5）在一个mian()方法中，在不添加和改动任何打印语句的情况下，最终打印结果如图：



|  |
| --- |
| **public** **class** China {  **int** x=10,y=10;  LiaoNing dalian=**new** LiaoNing();  **void** f(){  System.***out***.println("我是中国");  dalian.speak();  }  **class** LiaoNing{  **int** z=10;  **void** speak(){  System.***out***.println("我是大连，z="+z);  }  **void** g(){  z=x+y;  f();  }  }  }  **public** **class** Test {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  China china=**new** China();  china.f();  china.dalian.g();  }  } |

2、定义一个乐器(Instrument)接口，其中有抽象方法

   void play();

在InstrumentTest类中，定义一个方法

   void playInstrument(Instrument ins);

   并在该类的main方法中调用该方法。

要求：分别使用下列内部类完成此题。

成员内部类

局部内部类

匿名类

（10分）

|  |
| --- |
| //使用成员内部类实现  interface Instrument{  public abstract void play();  }  class InstrumentTest{  class Inner1 implements Instrument{  public void play(){  System.out.println("演奏钢琴");  }    }  public void playInstrument(Instrument ins){  ins.play();  }  }  public class demo8 {    public static void main(String[] args) {  InstrumentTest test = new InstrumentTest();//创建外部类对象  InstrumentTest.Inner1 inner= test.new Inner1();//创建内部类对象  test.playInstrument(inner);  }    } |
| //使用局部内部类实现  interface Instrument{  public abstract void play();  }  class InstrumentTest{  public void tell(){  class Piano implements Instrument{  public void play() {  System.out.println("演奏钢琴");  }  }  Instrument piano = new Piano();  piano.play();  }  public void playInstrument(Instrument ins){  ins.play();  }  }  public class demo8 {    public static void main(String[] args) {  InstrumentTest test = new InstrumentTest();//创建类对象  test.tell();  }    } |
| //使用匿名内部类实现  interface Instrument{  public abstract void play();  }  class InstrumentTest{  public void playInstrument(Instrument ins){  ins.play();  }  }  public class demo8 {    public static void main(String[] args) {  InstrumentTest test = new InstrumentTest();//创建类对象  test.playInstrument(new Instrument(){  public void play(){  System.out.println("弹奏钢琴");  }  });  }    } |

3、编程实现内部类的使用（10分）

|  |
| --- |
| public class Jubu {  private int size=5,y=7;  public Object makeInner(int localVar){   final int finalLocalVar=localVar;   //创建内部类，该类只在makeInner（）方法有效，就像局部变量一样。在方法体外部不能创建MyInner类的对象   class MyInner{    int y=4;    public String toString(){     return "OuterSize:"+size+         "\nfinalLocalVar"+" "+"this.y="+this.y;        }     }     return new MyInner();    } }  class Main{ /\*\*   \* @param args   \*/  public static void main(String[] args) {     Object obj=new Jubu().makeInner(47);//创建Jubu对象obj，并调用它的makeInner（）方法，该方法返回一个   //该方法返回一个MyInner类型的的对象obj，然后调用其同toString方法。   System.out.println(obj.toString());   // TODO Auto-generated method stub   } } |

4、按照要求，补齐代码

interface Inter { void show(); }

class Outer { //补齐代码 }

class OuterDemo {

public static void main(String[] args) {

Outer.method().show();

}

}

要求在控制台输出”HelloWorld”

\*/

|  |
| --- |
| 答案：  interface Inter {  void show();  //public abstract  }    class Outer {  //补齐代码  public static Inter method() {  //子类对象 -- 子类匿名对象  return new Inter() {  public void show() {  System.out.println("HelloWorld");  }  };  }  }    class OuterDemo {  public static void main(String[] args) {  Outer.method().show();    }  } |

5、编写一个主类为TestKey的程序实现如下功能：在窗口画出一个红色圆点，并可以通过键盘方向键（左右上下）控制红色圆点圆点进行移动。涉及到KeyEvent对应监听器接口的实现，paint()方法的重写，内部类等知识点。

import java.awt.\* ;

import java.awt.event.\*;

import java.util.\*;

public class TestKey {

public static void main(String[] args){

new KeyFrame().launchFrame() ;

}

}

class KeyFrame extends Frame{

int framLocaX= 200 ;

int framLocaY= 200 ;

//Point p = new Point(framLocaX+100, framLocaY+100) ;

Point p = new Point(100, 100) ;

public void launchFrame(){

setSize(400,400) ;

setLocation(framLocaX,framLocaY);

addKeyListener(new MyKeyMonitor()) ;

setVisible(true);

}

public void paint(Graphics g){

g.setColor(Color.BLUE);

g.fillOval(p.x, p.y, 10, 10);

}

class MyKeyMonitor extends KeyAdapter{

public void keyPressed(KeyEvent e){

int keyCode = e.getKeyCode() ;

/\*if(keyCode ==KeyEvent.VK\_UP){

System.out.println("UP") ;

}\*/

switch (keyCode){

case KeyEvent.VK\_UP:

p.y = p.y-10 ;

break;

case KeyEvent.VK\_DOWN:

p.y = p.y+10 ;

break;

case KeyEvent.VK\_RIGHT:

p.x = p.x+10 ;

break;

case KeyEvent.VK\_LEFT:

p.x = p.x-10 ;

break;

}

KeyFrame f= (KeyFrame)e.getSource() ;

f.repaint() ;

}

}

}

6、以下代码的输出结果是什么？；

（10分）

|  |
| --- |
| **public** **class** BwfOuterClass {    **private** **int** x = 1;  **private** **int** y = 2;  **private** **class** BwfInnerClass{  **private** **int** x = 3;  **public** **void** print(){  System.***out***.println("x+y="+(x+y) );  }  }  **public** **static** **void** main(String[] args) {  **new** BwfOuterClass().**new** BwfInnerClass().print();  }  } |

答案：x+y=5

7、创建Swan(天鹅)类既继承抽象类FlyingAnimal(飞行动物),又要继承SwimmingAnimal(游动物)。但Swan只能继承一个类FlyingAnimal，另一个类的继承与方法复写，只能在内部类中继承实现。所以在内部类中，通过getSwimming()方法，返回一个匿名内部类实例，并在Swan外部类中构造了swim()方法，来调用匿名内部类对象的swim()方法。

（10分）

|  |
| --- |
| abstract class FlyingAnimal {  abstract void fly();  }    abstract class SwimmingAnimal {  abstract void swim();  }    /\*\*  \* Swan(天鹅)类既继承抽象类FlyingAnimal(飞行动物),又要继承SwimmingAnimal(游动物)  \* @author zhoua  \*  \*/  public class Swan extends FlyingAnimal{    @Override  void fly() {  // TODO Auto-generated method stub  System.out.println("Swan.fly()");  }    void swim() {  this.getSwimming().swim();  }    SwimmingAnimal getSwimming() {  return new SwimmingAnimal(){    @Override  void swim() {  // TODO Auto-generated method stub  System.out.println("Swan.swim()");  }    };  }      public static void main(String[] args) {  Swan swan = new Swan();  swan.fly();  swan.swim();  }  } |

8、内部类编码：创建一个外部类及其内部类，并利用外部类访问成员内部类的属性方法

（10分）

|  |
| --- |
| class Circle {      private double radius = 0;        public Circle(double radius) {          this.radius = radius;          getDrawInstance().drawSahpe();   //必须先创建成员内部类的对象，再进行访问      }        private Draw getDrawInstance() {          return new Draw();      }        class Draw {     //内部类          public void drawSahpe() {              System.out.println(radius);  //外部类的private成员          }     }  } |

9、根据注释填写(1)，(2)，(3)处的代码

public class Test{

public static void main(String[] args){

// 初始化Bean1

(1)

bean1.I++;

// 初始化Bean2

(2)

bean2.J++;

//初始化Bean3

(3)

bean3.k++;

}

class Bean1

{

public int I = 0;

}

static class Bean2

{

public int J = 0;}

}

class Bean{

class Bean3{

public int k = 0;

}

// 1

Test t=new Test();

Test.Bean1 b1=t.new Bean1();

// 2

Test.Bean2 b2=new Test.Bean2();

// 3

Bean b=new Bean();

Bean.Bean3 b3=b.new Bean3();

10 下面这段代码的输出结果是什么？

public class Test {

public static void main(String[] args) {

Outter outter = new Outter();

outter.new Inner().print();

}

}

class Outter

{

private int a = 1;

class Inner {

private int a = 2;

public void print() {

int a = 3;

System.out.println("局部变量：" + a);

System.out.println("内部类变量：" + this.a);

System.out.println("外部类变量：" + Outter.this.a);

}

}

}

3

2

1

11、按照要求，补齐代码

interface Inter{

void show();

}

class Outer{

//补齐代码

public static Inter method(){

return new Inter(){

public void show(){

System.out.println("Hello World");

};

};

}

}

class OuterDemo{

public static void main(String[] args){

Outer.method().show();

}

}

要求在控制台输出”HelloWorld”。