**中软国际考题-笔试**

1. 单选题

1、下列说法正确的的是（D）

1. 匿名类创建的对象必须要经过类声明的编写步骤
2. 我们不可以用匿名类直接创建一个对象
3. 我名可以用匿名类声明一个对象

假设有一个类A，A中一个方法a；又有一个类B，B中有方法b,我们可以在b中使用A a=new A(){public void a(){}}.a();来重写A类的a方法，并且进行调用a方法

1. 我名可以用匿名类声明一个对象

假设有一个类A，A中一个方法a；又有一个类B，B中有方法b,我们可以在b中使用new A(){public void a(){}}.a();来重写A类的a方法，并且进行调用a方法

2、一个匿名类不包含下列哪部分（B）

A. new操作符

B.：

C.()：这个括号表示构造函数的参数列表。

D.{...}

3、在一类中被定义为且用Static修饰的类，称为（A ）。

A．静态成员类 B．匿名类

C．成员类 D．局部类

4、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

5、不可能出现在匿名内部类中的是\_\_C\_\_\_。

A.继承其他的类

B.实现了某一接口

C.定义了构造方法

D.可以使用成员变量

6、下列正确描述是（C）

A. 匿名类不推荐使用

B. 匿名类可以在内部类中使用

C. 匿名类没有名字只能使用一次

D. 匿名类不用实现接口即可使用

7、下列描述中正确的是（B）

A. 匿名类严格来说不算java类

B. 匿名内部类没有类名

C. 匿名内部类在任何情况下都可以创建

D. 匿名内部类可以自定义其构造函数

8、下列关于静态类描述错误的是（B）

A. 匿名内部类可以直接调用方法传递参数

B. 匿名内部类传递形式参数可以不为常量

C. 匿名内部类不能定义静态代码块

D. 匿名内部类不能定义接口。

9、java中匿名类描述正确的是（D）

A.匿名类与匿名内部类是2种java类

B. Java中匿名类实现方式只有靠继承

C. 匿名内部类代码块无法被外部类调用

D. 匿名类和局部内部类一样，可以访问外部类的所有成员

10、public class BwfOuterClass {

      private int x = 1;

      private int y = 2;

      private classBwfInnerClass{

           private int x= 3;

           public voidprint(){

                 System.out.println("x+y="+(x+y) );

           }

      }

      public static voidmain(String[] args) {

           newBwfOuterClass().new BwfInnerClass().print();

      }

} （A）

1. x+y=5 B、x+y=3

C、x+y=4 D、程序不能编译

11、Outer类中定义了一个成员内部类Inner，需要在main()方法中创建Inner类实例对象，以下四种方式哪一种是正确的？D

A、Inner in = new Inner()

B、Inner in = new Outer.Inner();

C、Outer.Inner in = new Outer.Inner();

D、Outer.Inner in = new Outer().new Inner();

12. 下列叙述正确的是：B

A) 方法中可以定义另一个方法，称为：内部方法。

B) 类中可以定义另一个类，称为内部类。

C) 接口中可以定义另一个接口，称为：内部接口。

D) 对象中可以包含另一个对象。

E) 属性中可以包含另一个属性。

13. 请分析以下代码能否被放置在相应的位置上A

class Demo{

public void func(){

//位置1

}

class Inner{}

public static void main(String [] args){

Demo d=new Demo();

//位置2

}

}

A. 位置1：new Inner();

B. 位置2：new Inner();

C. 位置2：new d.Inner();

D. 位置2：new Demo.Inner();

14.下列代码编译运行结果是：C

public class Tree {

private static String tree = "tree";

String getTree() {

return tree;

}

public static class Elm extends Tree {

private static String tree = "elm";

public static void main(String[] args) {

new Elm().go(new Tree());

}

void go(Tree t) {

String s = t.getTree() + Elm.tree + tree + (new Elm().getTree());

System.out.println(s);

}

}

}

A. elm elm elm elm

B. tree elm elm elm

C. tree elm elm tree

D. elm elm tree elm

1. 多选题
2. 下列说法正确的是？（A、B）
3. 匿名了可以继承类的方法;
4. 匿名类可以重写类的方法
5. 匿名类不一定是内部类;
6. 匿名类不可以访问外嵌类中的成员变量;

2、匿名类中可以包含的东西有（A、B、C、D）

1. 字段
2. 方法
3. 实例初始化代码
4. 本地类

3、匿名内部类可以分为以下哪种类型（A、B、C）：

A . 继承式的匿名内部类  
B. 接口式的匿名内部类

C. 参数式的匿名内部类

D. 多态式的匿名内部类

4、下面哪几种陈述是成立的（ AB ）

A.一个内部类可声明为静态的（static）

B.一个匿名内部类可声明为公共的（public）

C.一个匿名内部类可声明为私有的（private）

D.一个匿名内部类可继承一个抽象类

E.一个匿名内部类可声明为保护型（protected）

1. 以下选项中关于匿名内部类的说法正确的是（ BC ）

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;

6、以下关于匿名类说法错误的是？（A、C）

A. 内部类一定是匿名内部类

B. 全局类无法定义为匿名类

C. 一个全局类只能定义一个匿名类

D. 内部类与匿名类可以分别创建

7、下面的五个选择中哪两个描述是正确的？（ C E ）

A、static inner class requires a static initializer

B、 A static inner class requires an instance of the enclosing class

C、A static inner class has no reference to an instance of the enclosing class

D、A static inner class has accesss to the non-static member of the other class

E、static members of a static inner class can be referenced using the class

     name of the static inner class

8、在使用匿名内部类时，要记住以下几个原则：（ABD）

A. 不能有购置方法，只能有一个实力。

B. 不能定义在任何静态成员、静态方法。

C. 能是public.protected.private.static.

D. 一定是在new的后面，用其隐含实现一个接口或一个类。

1. 填空题
2. 匿名类不可声明\_\_\_\_\_\_static（静态）\_\_\_\_成员变量和\_\_\_\_static（静态）\_\_\_\_\_方法
3. 匿名内部类也就是没有名字的\_\_\_\_\_\_\_\_\_\_ 。(内部类)

3、匿名类的一般格式是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

new ClassName(){

……

}

4、编写事件监听器的时候，采用\_\_\_内部类\_\_\_\_和\_\_\_匿名类\_\_\_\_\_编写可以很容易实现。

5、interface Inter{

void func();

}

class Demo{

public static void main(String [] args){

//使用匿名内部类补足代码

}

void show(Inter inter){ t.func(); } }

(

new Demo().show(new Inter(){

public void func(){

System.out.println("hello world");

}

});

)

6、定义匿名类格式：\_\_\_\_\_\_\_\_\_ 父类构造器 （\_\_\_\_\_\_\_\_\_）

{

//匿名内部类的类体部分

} (new 参数 )

* 1. 匿名类的声明是在 编译时 进行的，实例化在 运行时 进行
  2. 接口解决了部分问题，而内部类有效地实现了\_\_多重继承\_\_。
  3. 内部类之所以能获取到外部类的成员，是因为他能获取到外部类的引用，外部类名\_\_.this\_\_\_
  4. 是指在一个外部类的内部再定义一个类,类名不需要和\_\_文件夹\_相同

1. 判断题
2. 匿名类是一个子类（T）
3. 匿名类一定是内部类（T）
4. 如果某个方法的参数是接口，那么我们可以使用接口名和类体组合创建一个匿名对象传递给方法的参数，类体可以不实现接口中的全部方法（F）（类体必须要实现接口中的全部方法）
5. 使用匿名内部类还有个前提条件：必须继承一个父类或实现一个接口。（ T ）
6. **内部类访问的外部变量必须定义为 final 或 static 类型（T）**
7. 匿名类总是隐式的final。**（T）**
8. 匿名类可以继承类的方法，也可以重写类的方法。（T）
9. 匿名类可以声明static成员变量和static方法。 （F）
10. 可以用匿名类声明一个对象。 （F）
11. 匿名类可以访问外嵌类中的成员变量和方法。 （T）
12. 匿名类是指没有类名的内部类。( T )
13. 匿名类就是一个没有类申明的类体，因为没有名称，所有匿名类不可以申明对象，但却可以创建对象。（ T ）
14. 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 ）
15. 由于匿名内部类没有自己的类名，所以类的定义及其对象的创建（实例化）是同步完成的。（T）
16. 匿名内部类是唯一一个没有构造方法的类（T）
17. 匿名内部类是没有名字的，所以我们没办法获得其类型，而只能把它当作超类或接口类型来使用。（ T ）
18. 匿名类可以写在java文件类中。（ N ）
19. 匿名类只能访问局部final变量（ T ）
20. 匿名内部类没有显式构造器（T）
21. 每个内部类都能独立地承受来自一个（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。（ T ）
22. 匿名内部类不是局部的，所以不是所有限制都对其有效。（F）
23. 成员内部类可以无条件访问外部类的所有成员属性和成员方法（T）

五、简答题

# 1、什么是匿名了？

答案：当使用类创建对象时，程序允许我们把类体与对象的创建组合在一起。也就是说，类创建对象时，除了构造方法还有类体，此类体被认为是该类的一个子类去掉类声明后的类体，此外假设有一个接口，java允许直接用接口名和一个类体创建一个匿名对象，此类体被认为是实现了这个接口的类去掉类声明后的类体，即匿名类，

2、匿名类的访问权限有哪些（5分）

答案：1. 访问外层Class里面的字段。   
2. 不能访问外层方法中的本地变量。除非变量是final。   
3. 如果内部类的名称和外面能访问的名称相同，则会把名称覆盖掉。   
4. 不能定义静态初始化代码块   
5. 不能在匿名类里面定义接口   
6. 不能在匿名类中定义构造函数

# 3、简述内部类、静态内部类、匿名内部类的区别

答：

内部类：

成员内部类可访问外部类所有的方法和成员变量。

不能有静态的方法和成员变量。

静态内部类：

    只能访问外部类的静态成员变量与静态方法。

    静态内部类的非静态成员可访问外部类的静态变量，而不可访问外部类的非静态变量。

匿名内部类：

    没有类名，没有class关键字也没有extends和implements等关键字修饰。

类的定义和对象的实例化同时进行。

4、与类有关的匿名类一定是该类的子类吗？与接口有关的匿名类一定是实现该接口的一个类吗？

答：一定，因为类创建对象时，除了构造方法还有类体，此类体被认为是该类的一个子类去掉类声明后的类体。即匿名类，匿名类就是一个子类。与接口的匿名类相同。一定是子类。

5、在使用匿名内部类时，要记住那几个原则？

答：

1. 使用匿名内部类时，我们必须是继承一个类或者实现一个接口，但是两者不可兼得，同时也只能继承一个类或者实现一个接口。

2. 匿名内部类中是不能定义构造函数的。

3. 匿名内部类中不能存在任何的静态成员变量和静态方法。

4. 匿名内部类为局部内部类，所以局部内部类的所有限制同样对匿名内部类生效。

5. 匿名内部类不能是抽象的，它必须要实现继承的类或者实现的接口的所有抽象方法。

6、匿名类的特点有哪些？

答：

（1）匿名类可以继承父类的方法也可以重写父类的方法。

（2） 匿名类可以访问外嵌类中的成员变量和方法，在匿名类中不能声明静态变量和静态方法。

（3）使用匿名类时，必须在某个类中直接使用匿名类创建对象。

（4）在使用匿名类创建对象时，要直接使用父类的构造方法。

7、Anonymous Inner Class (匿名内部类) 是否可以extends(继承)其它类，是否可以implements(实现)interface(接口)?

匿名的内部类是没有名字的内部类。不能extends(继承) 其它类，但一个内部类可以作为一个接口，由另一个内部类实现。

8、使用内部类的好处（5分）

答案：、内部类方法可以访问该类定义所在的作用域中的数据，包括私有的数据；

2、内部类可以对同一个包中的其他类隐藏起来；

3、使用匿名内部类定义回调函数，节省代码；

* 9、请简述匿名内部类的特点（5分）

答案： 1、使用匿名内部类时，我们必须是继承一个类或者实现一个接口，但是两者不可兼得，同时也只能继承一个类或者实现一个接口。

2、匿名内部类中是不能定义构造函数的。

3、匿名内部类中不能存在任何的静态成员变量和静态方法。

4、匿名内部类为局部内部类，所以局部内部类的所有限制同样对匿名内部类生效。

5、匿名内部类不能是抽象的，它必须要实现继承的类或者实现的接口的所有抽象方法。

# 10、局部内部类和匿名内部类的访问权限（5分）

答案：1、 俩者只能访问局部final变量（局部:俩者存在的作用域内的变量）

2、 局部内部类就像是方法里面的一个局部变量一样，是不能有public、protected、private以及static修饰符的，匿名内部类也是不能有访问修饰符和static修饰符的

11、什么是内部类？

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三种：成员内部类、局部内部类、匿名内部类。

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？

答案：

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？

答案：

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

六、程序实现题

1、创建一个类A，方法为a,在a中打印aaaa;创建一个带有main()方法得类B，在B的main()方法中使用匿名类重写A类a方法，使其打印bbbb;运行B类，控制台打印bbbb

|  |
| --- |
| **public** **class** A {  **public** **void** a(){  System.***out***.println("aaaa");  }  }  **public** **class** B {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **new** A(){  **public** **void** a(){  System.***out***.println("bbbb");  }  }.a();  }  } |

2、编程实现:**匿名内部类方法的访问**（10分）

|  |
| --- |
| new D(){  Public void ShowContext() { System.out.println("hello"); } }.ShowContext(); |

3、补全以下代码、要求在控制台输出”HelloWorld”

public class Test1 {

    public static void main(String[] args) {

        // <使用下面这句代码输出HelloWrold>

        Outer.method().show();

    }

}

interface Inter {

    void show();

}

class Outer {

    // <补全代码>

}

答：

    public static Inter method() {

        // 直接返回一个实现Inter类的匿名内部类 并重写内部类的show方法

        return new Inter() {

            public void show() {

                System.out.println("HelloWorld");

            }

        };

    }

4、补全代码，要求分别输出30，20，10。  
          
    注意：  
        1:内部类和外部类没有继承关系。  
        2:通过外部类名限定this对象  
            Outer.this

class Outer {  
    public int num = 10;  
    class Inner {  
        public int num = 20;  
        public void show() {  
            int num = 30;  
            System.out.println(填写代码);  
            System.out.println(填写代码);  
            System.out.println(填写代码);  
        }  
    }  
}  
class InnerClassTest {  
    public static void main(String[] args) {

          //补全代码。  
    }      
}

答：

//外部类名.内部类名 对象名 = 外部类对象.内部类对象;

Outer.Inner oi = new Outer().new Inner();

oi.show();

5、分析以下程序运行的结果。

public class Anony {

public void test(Device d){

System.out.println("购买了一个"+d.getName()+",花掉了"+d.getPrice());

}

public static void main(String[] args) {

Anony anony= new Anony();

anony.test(new Device() {

@Override

public double getPrice() {

return 56.7;

}

});

Device device =new Device()

{

{

System.out.println("匿名内部类的初始化块");

}

@Override

public double getPrice () {

return 56.2;

}

@Override

public String getName () {

return "键盘";

}

};

anony.test(device);

}

}

答：

购买了一个null,花掉了56.7

匿名内部类的初始化块

购买了一个键盘,花掉了56.2

6、分析以下程序运行的结果。

interface Show{

public void show();

}

class A{

void f(Show s){

s.show(); //s调用匿名类体中实现的接口方法（接口回调）

}

}

public class Example{

public static void main(String args[]){

A a=new A();

a.f(new Show(){

public void show(){

System.out.println("这是实现了接口的匿名类");

}

});

}

}

答：

这是实现了接口的匿名类

* 1. 补足代码；调用两个函数，要求用匿名内部类

interface Inter

{

void show(int a,int b);

void func();

}

class Demo

{

public static void main(String[] args)

{

Inter in = new Inter()

{

public void show(int a,int b)

{

}

public void func()

{

}

};

in.show(4,5);

in.func();

}

}

8、补足代码；(匿名内部类)

interface Test

{

void func();

}

class Demo

{

public static void main(String[] args)

{

new Demo().show(new Test()

{

public void func(){}

});

}

void show(Test t)

{

t.func();

}

}

9、使用匿名内部类的方式实现多线程。

（10分）

|  |
| --- |
| private static void Test3() {  　　　　new Thread() {  public void run() {  System.out.println("线程执行了。。。。。。。");  }  }.start();  } |

10、定义常量： 定义能够调用方法的匿名类（10分）

|  |
| --- |
| public abstract class Bird {  private String name;  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }    public abstract int fly();  }  public class Test {    public void test(Bird bird){  System.out.println(bird.getName() + "能够飞 " + bird.fly() + "米");  }    public static void main(String[] args) {  Test test = new Test();  test.test(new Bird() {    public int fly() {  return 10000;  }    public String getName() {  return "大雁";  }  });  }  } |

11、按照要求，补齐代码

interface Inter { void show();

}

class Outer {

//此处补齐代码

}

class OuterDemo {

public static void main(String[] args) {

Outer.method().show();

}

}

要求在控制台输出”HelloWorld” （10分）

|  |
| --- |
| 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();  }  } |

12、运用匿名内部类启动一个线程，并打印出字符串”Running“。

|  |
| --- |
| ipublic class Demo{  public static void main(String[] args) {    new Thread(new Runnable() {  @Override  public void run() {  System.out.println("running");  }  }) {  }.start();  }  } |

13、在方法中编写一个匿名内部类。

（10分）

|  |
| --- |
| package com.lzw;    public class AnonymityInnerClass {  }  class OuterClass4{        public OutInterface doit(final String s){          return new OutInterface(){              private int i=0;              public int getValue(){                  System.out.println(s);                  return i;                }              public void f(){                  System.out.println("f()");              }          };      }    }  interface OutInterface { // 定义一个接口  } |