Java 程序员面试宝典（重点有标记 推荐初级学者认真看最后一道面试题 我几回笔试都看到了类此的 必考题哦）

面向对象

面向对象其实是现实世界模型的自然延伸。现实世界中任何实体都可以看做是对象。对象之间通过消息相互作用。另外，现实世界中任何实体都可归属于某类事物，任何对象都是某一类事物的实例。如果说传统的过程式编程语言是以过程为中心、以算法为驱动的话，面向对象的编程语言则是以对象为中心，以消息为驱动。用公式表示，过程式编程语言为：程序=算法+数据；面向对象编程语言为：程序=对象+消息。

所有面向对象编程语言都支持 3 个概念，即封装、多态性和继承，Java 也不例外。现实世界中的对象均有属性和行为，映射到计算机程序上，属性则表示对象的数据，行为表示对象的方法（其作用是处理数据或同外界交互）。所谓封装，就是用一个自主式框架把对象的数据和方法连在一起形成一个整体。可以说，对象是支持封装的手段，是封装的基本单位。Java 语言的封装性较强，因为 Java 无全程变量，无主函数，在 Java 中绝大部分成员是对象，只有简单的数字类型、字符类型和布尔类型除外。而对于这些类型，Java 也提供了相应的对象类型以便与其他对象交互操作。

“这个世界是由什么组成的？”这个问题如果让不同的人来回答会得到不同的答案。如果是一个化学家，他也许会告诉你：“还用问吗？这个世界是由分子、原子、离子等化学物质组成的”。如果是一个画家，他也许会告诉你：“这个世界是由不同的颜色所组成的”。但如果让一个分类学家来考虑问题就有趣多了，他会告诉你：“这个世界是由不同类型的物与事所构成的”。好！作为面向对象的程序员，我们要从分类学家的角度去考虑问题！是的，这个世界是由动物、植物等组成的。动物又分为单细胞动物、多细胞动物、哺乳动物等，哺乳动物又分为人、大象、老虎……就这样分下去了！

现在，从抽象的角度，我们给“类”下个定义吧！我的意思是，从抽象的角度，你回答我“什

么是人类？”首先让我们来看看人类所具有的一些特征，这个特征包括属性（一些参数、数值）及方法（一些行为，他能干什么）。每个人都有身高、体重、年龄、血型等一些属性。还有人会劳动、人都会直立行走、人都会用自己的头脑去创造工具等方法。人之所以能区别于其他类型的动物，是因为每个人都具有人这个群体的属性与方法。“人类”只是一个抽象的概念，它仅仅是一个概念，它是不存在的实体。但是所有具备“人类”这个群体的属性与方法的对象都叫人。这个对象“人”是实际存在的实体。每个人都是人这个群体的一个对象。老虎为什么不是人？因为它不具备人这个群体的属性与方法，老虎不会直立行走，不会使用工具等，所以说老虎不是人。

由此可见，类描述了一组有相同特性（属性）和相同行为（方法）的对象。在程序中，类实际上就是数据类型，例如，整数、小数等。整数也有一组特性和行为。面向过程的语言与面向对象的语言的区别就在于，面向过程的语言不允许程序员自己定义数据类型，而只能使用程序中内置的数据类型。而为了模拟真实世界，为了更好地解决问题，我们需往往要创建解决问题所必需的数据类型。

面向对象编程为我们提供了解决方案。以下的考题来自真实的笔试资料，希望读者先不要看答案，自我解答后再与答案加以比对，找出自己的不足。

11.1 面向对象的基本概念

面试例题 1：对象与实例有什么区别？

解析：在 Java 的内存分配中，对象和实例是不同的，前者分配在内存堆里，后者分配在堆栈里，至于为什么要这样分，参考一下其他的资料。对象是不变的东西，它是对客观事物的抽象，实例是对操作对象的引用，你不能直接操作对象。

答案：对象和实例从宏观的角度看，区别是：对象是同类事物的一种抽象表现形式，而实

例是对象的具体化，一个对象可以实例化很多实例，对象就是一个模型，实例是照着这个模型生产的最终产品。实际上就是这样，一个对象可以实例化 N 个实例。就像根据一个模型可以制造多个实际的产品一样。

从内存分配的角度来看，对象是保存在堆中的，而实例是存储在栈中的，实例其实只是对象的一个引用，也就是指向对象的指针。

面试例题 2: Java 中所有的类都继承了 java.lang.Object 类，而在 C++中没有像 java.lang.Object 这样的类，这是为什么呢？都是面向对象，Java 这样做有什么好处呢？

解析：Java 采用的是单根结构，所有的类都继承了 java.lang.Object 类。对于 Java 这种纯面向对象的语言来说，这种设计具有很好的灵活性，比如对垃圾收集来说很有利，所有的类都具有 java.lang.Object 类具有的方法等。C++没有更多这样的方法，大概是为了向后兼容。向 C 兼容，满足它设计上最大的灵活性。

答案：C++的特点是指针，一个指针可以指向任何的对象、结构、基本类型、函数。

Java 没有指针类型，所以 Java 用 Object 基础类的共同特征来实现所有对象类型之间的转换。

面试例题 3：声明与定义之间的区别是什么？在 Java 里声明和定义能否分开？

解析：

声明：一种把一个名称引入或者重新引入到某个作用域的构造。

定义：它也是一种声明，但该声明必须给出被声明实体的细节。

对于变量而言，这里的细节是指：为被声明实体保留存储空间。

对于 class 类型和函数定义而言，指的是包含有一对花括号内容的声明。

对于外部变量而言，指的是前面没有关键字 extern 或者在声明时就进行初始化。

变量的声明有以下两种情况:

一种是需要建立存储空间的。例如，int a 在声明的时候就已经建立了存储空间。

另一种是不需要建立存储空间的，例如，extern int a。其中，变量 a 是在别的文件中定义的。前者是“定义性声明（defining declaration）”，或者称为“定义（definition）”，而后者是“引用性声明（referencing declaration）”。从广义的角度来讲，声明中包含着定义，但是并非所有的声明都是定义，例如 int a，它既是声明，同时又是定义。然而对于 extern a 来讲，它只是声明不是定义。在一般的情况下我们常常这样叙述，把建立空间的声明称为“定义”，而把不需要建立存储空间的声明称为“声明”。很明显在这里指的声明的范围是比较窄的，也就是说非定义性质的声明。

答案：

变量的定义和声明的区别在于是否分配内存，如果分配内存就是定义，否则就是声明。

类中的变量只是声明而不是定义，因为它只是类声明的一部分。不同于变量的定义，类的定义是一种新的类型的定义，只有实例化后才会分配内存。所以类成员变量只是声明而不是定义。

在 Java 中，利用 Interface，也可以将声明和实现分开。如下所示。

```
 //MyInterface.java
public   interface   MyInterface
{
void   method();
}

//MyImpl.java
public   class   implements   MyInterface
{
public   void   method()
{
//……
}
}
```

面试例题 4：Which is incorrect about the class？（关于类的描述下面哪个是错误的？）[金山公司 2005 年面试题]

```
A. A class is a blueprint to objects.
B. We use the keyword class to create a class construct.
C. Once a class is declared, the class name becomes a type name and
can be used to declare variables.
D. The class is same as the struct, and there are no different between
class and struct.
```

解析：这道题的考点是类的概念。

答案：D

面试例题 5：Which is incorrect about the OOP？（下面关于面向对象技术的叙述哪个是错误的?）[金山公司 2005 年面试题]

```
A. The central idea of OOP is to build programs using software objects.
B. The OOP focuses mainly on the step-by-step procedure as procedure-
oriented programing.
C. The OOP offers many advantages:
simplicity,modularity,modifiability,
extensibility, and so on.
D. The key concept of object orientation is the attachment of procedure
to data.
```

解析：OOP 的概念面试例题。面向对象和面向过程不能混为一谈。

答案：B，D

11.2 类和对象

面试例题 1：以下代码编译时会产生错误的是_____。[Trend 公司 2005 年面试题]

```
class reverseIt4
{

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

```
EnclosingClass jb2;                 //-----1
System.out.println(jb2.m);      //-----2
}


}

    class EnclosingClass           //--------3
{
public int m = 6;
class InnerClass         //-------4
{
int msquare;
InnerClass()
{
msquare = m*m;
}
}
}
```

A．语句 1 B．语句 2 C．语句 3 D．语句 4

解析：语句 3 和语句 4 显然是正确的，尽管它们的描述不是那么规范（存在一个类中的类）。语句 1 声明了一个类，但是没有做定义，于是问题就出现了。声明好比只是告诉编译器有一个人，但是如果不定义，这个人就是个抽象的人，没有身高、体重、年龄、职业的“空”人。所以定义对象必须在声明的同时给它定义。正确的程序如下。

```
class reverseIt4
{

public static void main(String[] args)
{
EnclosingClass jb = new EnclosingClass();

System.out.println(jb.m);

}



}

    class EnclosingClass
{
public int m = 6;
class InnerClass
{
int msquare;
```

```
InnerClass()
{
msquare = m*m;
}
}
}
```

答案：该题是问编译在哪儿出现问题，尽管问题出在 1 处，但编译器不会发现，编译器只有在 2 处才会发现问题。所以答案选 B。

面试例题 2：Object 是所有类的父类，任何类都默认继承 Object。Object 类到底实现了哪些方法？

答案：

1．clone 方法

保护方法，实现对象的浅复制，只有实现了 Cloneable 接口才可以调用该方法，否则抛出 CloneNotSupportedException 异常。

2．getClass 方法

final 方法，获得运行时类型。

3．toString 方法

该方法用得比较多，一般子类都有覆盖。

4．finalize 方法

该方法用于释放资源。因为无法确定该方法什么时候被调用，很少使用。

5．equals 方法

该方法是非常重要的一个方法。一般 equals 和==是不一样的，但是在 Object 中两者是一样的。子类一般都要重写这个方法。

6．hashCode 方法

该方法用于哈希查找，重写了 equals 方法一般都要重写 hashCode 方法。这个方法在一些具有哈希功能的 Collection 中用到。

一般必须满足 obj1.equals(obj2)==true。可以推出 obj1.hash- Code()==obj2.hashCode()，但

是 hashCode 相等不一定就满足 equals。不过为了提高效率，应该尽量使上面两个条件接近等价。

7．wait 方法

wait 方法就是使当前线程等待该对象的锁，当前线程必须是该对象的拥有者，也就是具有该对象的锁。wait()方法一直等待，直到获得锁或者被中断。wait(long timeout)设定一个超时间隔，如果在规定时间内没有获得锁就返回。

调用该方法后当前线程进入睡眠状态，直到以下事件发生。

（1）其他线程调用了该对象的 notify 方法。

（2）其他线程调用了该对象的 notifyAll 方法。

（3）其他线程调用了 interrupt 中断该线程。

（4）时间间隔到了。

此时该线程就可以被调度了，如果是被中断的话就抛出一个 InterruptedException 异常。

8．notify 方法

该方法唤醒在该对象上等待的某个线程。

9．notifyAll 方法

该方法唤醒在该对象上等待的所有线程。

11.3 嵌套类

面试例题 1：请说明 static nested class 和 inner class 的不同。

答案：

1．nested（嵌套）class（一般是 C++的说法）

nested class 是合成型聚集关系（Composite Aggregation）的另一种表达方式，也就是说 nested class 也可以用 Aggregation 表达出来。但是，nested class 更加精确地表达了一种专用的、紧耦合的关系，尤其在代码生成时，nested class 在 Java 中映射成 inline class。比

如，计算机专用开关电源类可以作为计算机类的 nested class，但是，电池组类就不一定适合作为计算机类的 nested class，因为，电池组类表述的是一个过于通用的对象，可能还被包含（Aggregation）于模型中的其他设备对象。class A nested in class B，则说明 A 是一个 nested class，一般 A 是用来完成 B 中的某种重要功能的。

2．inner class（一般是 Java 的说法）

Java 内部类与 C++嵌套类最大的不同就在于是否有指向外部的引用上。

静态内部类（inner class）意味着：创建一个 static 内部类的对象，不需要一个外部类对象；不能从一个 static 内部类的一个对象访问一个外部类对象。

面试例题 2：关于下面类的定义，以下哪种说法是正确的？[研华科技 2005 年 11 月面试题]

```
public class Droitwich{
class one{
private class two{
public void main(){
System.out.println("two");
}
}
}
}
```

A．此代码不能编译成功，因为该类嵌套了不止一层

B．此代码不能编译通过，因为 class two 是私有的

C．此代码可以编译通过，并且在运行的时候输出字符串 two

D．此代码编译没有错误

解析：

内部类或嵌套类在类层级上没有限制，所以选项 A 是错误的。

内部类可以是私有类，所以选项 B 也是错误的。

选项 C 的 main 方法不是 public static void main 的，并且假设命令行是 java Droitwich，它不能在任何地方被调用。

答案：D

第 12 章　继承与接口

接口在实际语言，如 Delphi、Java、C++等中，都有广义和狭义之分，这很重要，以前就是因为没明白接口的广义和狭义之分，始终没能真正理解接口的真正意义。

广义接口从一般意义上说，凡是一个类提供给外部使用的部分都可以被称为接口。但是在引入继承和抽象类之前，这个广义接口并没有太大意义。广义接口的真正意义是在类的继承中体现多态的功能，这种接口又被称为抽象类接口。

狭义接口是指特定的函数集合，一般是用 interface（Delphi）声明的，它表示一个方法集合，这个集合被称为一个命名接口。一个命名接口中的方法必须在一个类中实现后才能被使用，一个类继承实现一个接口，称为这个类实现了该接口，一个接口可以被多个类实现，一个类也可以继承多个接口，这样就形成了一种灵活的接口调用方式，从而实现更加灵活和节省资源的多态。

从上述认识来看，接口实际上是结合着多态而来的，它的最大的任务就是实现多态。而多态又是面向对象最精华的理论，掌握了多态，也就掌握了面向对象的精髓。但掌握多态必须先理解和掌握接口，只有充分理解接口的意义，才能更好地应用多态。

在面试过程中，各大企业会考量你对虚函数、纯虚函数、私有继承、多重继承等知识点的掌握程度。因此，这是本书比较难掌握的一章。

12.1　基础知识

面试例题 1：下面哪一项说法是正确的？

A．在一个子类中一个方法不是 public 的就不能被重载

B．覆盖一个方法只需要满足相同的方法名和参数类型就可以了

C．覆盖一个方法必须需要相同的方法名参数和返回类型

D．一个覆盖的方法必须有相同的方法名、参数名和参数类型

解析：

对于在同一可访问区内被声明的几个具有不同参数列（参数的类型、个数、顺序不同）的同名函数，程序会根据不同的参数列来确定具体调用哪个函数，这种机制叫重载，重载不关心函数的返回值类型。覆盖是指派生类中存在重新定义的函数，其函数名、参数列、返回值类型必须同父类中的相对应被覆盖的函数严格一致，覆盖函数和被覆盖函数

只有函数体（花括号中的部分）不同，当派生类对象调用子类中该同名函数时会自动调用子类中的覆盖版本，而不是父类中的被覆盖函数版本，这种机制就叫做覆盖。

成员函数被重载的特征如下。

（1）相同的范围（在同一个类中）；

（2）函数名字相同；

（3）参数不同；

（4）virtual 关键字可有可无。

覆盖的特征如下。

（1）不同的范围（分别位于派生类与基类）；

（2）函数名字相同；

（3）参数相同；

（4）基类函数必须有 virtual 关键字。

答案：C

面试例题 2：下面的说法中哪项是正确的？

A．静态方法不能被覆盖成非静态的方法

B．静态方法不能被声明成私有的

C．私有的方法不能被重载

D．一个重载的方法在基类中不通过检查不能抛异常

解析：JDK 1.1 版本会发布这样一个提示信息：静态的方法不能被覆盖。选项 B 和 C 的说法并不合理，没有合理的理由来说静态的方法不能被声明成私有的，或私有的方法不能被重载。选项 D 是对于一个覆盖方法异常限制的混杂版本来说的。

答案：A

面试例题 3：给定下面的代码。

```
 class Base {}

class Agg extends Base{
public String getFields(){
String name =  "Agg";
return name;
}
}


public class Avf{
public static void main(String argv[]){
Base a = new Agg();
//Here
}
}
```

What code placed after the comment //Here will result in calling the getFields method resulting in the output of the string "Agg"?（下面哪个选项的代码替换到//Here，会调用 getFields 方法使输出结果输出字符串“Agg”？）

A． System.out.println(a.getFields());

B．System.out.println(a.name);

C．System.out.println((Base) a.getFields());

D．System.out.println(((Agg) a).getFields());

解析：Base 类型要引用 Agg 类的实例需要把 Base 类显示转换为 Agg 类，然后调用 Agg 类中的 getFields()方法。如果 a 是 Base 类的一个实例的话，它要调用 getFields()方法，那此方法在 Base 类中是不存在的，必须把 a 转换为 Agg 类的一个实例，这样才可以调用它里面的方法。

答案：D

面试例题 4：如果在下列代码中的 Here 处添加一段代码，问哪一个选项不能通过编译[Trend 公司 2005 年 10 月面试题]

```
public class Upton{
public static void main(String argv[]){
}
public void amethod(int i){}
//Here

}
```

A．public int amethod(int z){}

B．public int amethod(int i,int j){return 99;}

C．protected void amethod(long l){}

D．private void anothermethod(){}

解析：选项 A 不能通过编译。一个方法是显式地返回一个 int 值的方法，另一个是在同一个类中上述方法的一个重定义。方法中参数从 i 换做 z 对一个方法并没有任何影响。一个方法不能在同一个类中被覆盖。

答案：A

面试例题 5：下面代码的输出结果是多少？[Trend 公司 2005 年 10 月面试题]

```
class A {
public static void prt() {
System.out.println("1");
}

    public A() {
System.out.println("A");
}
}

    public class B extends A {
public static void prt() {
```

```
System.out.println("2");
}

    public B() {
System.out.println("B");
}

    public static void main(String[] args) {
A a = new B();
a = new A();
}
}
```

解析：每新建一个对象，都会产生一个构造函数，因为产生构造函数的顺序是 A，B，A，所以结果是 A，B，A。

答案：A，B，A

面试例题 6：下面代码的输出结果是多少？[Trend 公司 2005 年 10 月面试题]

```
class Parent {

    protected String value = "123";

    public String getValue() {
return value;
}
}

    public class Child extends Parent {

    protected String value = "456";

    }
```

解析：父类里的东西也可以理解成你自己的东西。也就是说在程序里面有两个 value，一个是 123，另一个是 456。而现在要输出的是父类里面的那个 value，所以就是 123。原因是在输出语句中使用的是 getValue()方法，而这个方法就是父类里面的方法。它的返回值是父类里面定义的 value，从父类继承来的没有被子类覆盖的方法操作的是继承于父类的被隐藏的变量，也就是 123。

答案：123

12.2 Super

面试例题 1：以下代码的输出结果是下列哪个选项？[Sun 公司 2005 年面试题]

```
class Base{
Base(){
System.out.println("Base");
}
}

    public class Checket extends Base{
Checket(){
System.out.println("Checket");
super();

}
public static void main(String argv[]){
Checket c = new Checket();
//  super();
}


}
```

A．Compile time error

B．Checket followed by Base

C．Base followed by Checket

D．runtime error

解析：这是一个考查 super 构造函数的面试例题。子类的构造函数如果要引用 super 的话，必须把 super 放在函数的首位，不然会出现这样的报错：

```
Checket.java:10: call to super must be first statement in
constructor
super();
```

如果一定要引用 super 构造函数，则必须把 super()放在前面，代码如下。

```
class Base{
Base(){
System.out.println("Base");
}
}

    public class Checket extends Base{
Checket(){
```

```
super();
System.out.println("Checket");
}
public static void main(String argv[]){
Checket c = new Checket();
//  super();
}


}
```

答案：会出现编译报错，选项 A

面试例题 2：Java 里在类中用 super 调用父类构造函数时，为什么调用语句必须是子类的第一条语句？[Siemens 公司 2005 年面试题]

答案：如果想用 super 继承父类构造的方法，但是没有放在第一行的话，那么在 super 之前的语句，肯定是为了满足自己想要完成某些行为的语句，但是又用了 super 继承父类的构造方法。那么以前所做的修改就都回到以前了，就是说又成了父类的构造方法了。如下面的程序所示。

```
class Father
{
public Father()
{String name=null;
int age=0;}
}
class Son extends Father
{
public Son()
{String name="学生";
super();
}
}
```

扩展知识（Java 中的 super 类）

在 Java 中，有时还会遇到子类中的成员变量或方法与超类（有时也称父类）中的成员变量或方法同名。因为子类中的成员变量或方法名优先级高，所以子类中的同名成员变量或方法就隐藏了超类的成员变量或方法，但是我们如果想要使用超类中的这个成员变量或方法，就需要用到 super。请看下面的类。

```
class Country
  {
String name;
void value()
{
  name="China";
}
  }
```

在下面的子类中，子类的成员变量和方法隐藏了超类的成员变量 name 和方法 value()。

```
class City extends Country
String name;
void value()
{
  name="Hefei";
  super.value();
  System.out.println(name);
  System.out.println(super.name);
  }
```

为了在子类中引用超类中的成员变量 name 和方法 value()，在代码中使用了 super、super.name 和 super.value()，所以显示的结果为：

Hefei

China

如果想要使用超类的构造函数，则应当使用 super(参数列表)的形式。

面试例题 3：给定下面的代码，哪个选项在替代“//Here”后可以被编译并且改变变量 oak 的值？

```
class Base{
static int oak=99;
}

public class Doverdale extends Base{
```

```
public static void main(String argv[]){
Doverdale d = new Doverdale();
d.amethod();
}
public void amethod(){
//Here
}
}
```

A．super.oak=1;

B．oak=33;

C．Base.oak=22;

D．oak=50.1;

解析：因为变量 oak 被声明是静态的，如果它存在只能有一个本体，则它可以通过本类的名字或者通过定义本类的任何一个实例被改变。

答案：A、B、C

面试例题 4： 当编译和运行下列代码时会发生下列哪种情况？

```
class Base{
Base(){
System.out.println("Base");
}
}

public class Checket extends Base{
public static void main(String argv[]){
Checket c = new Checket();
super();
}

Checket(){
System.out.println("Checket");
}
}
```

A．Compile time error

B．Checket followed by Base

C．Base followed by Checket

D．runtime error

解析：

用 Sun 的 JDK 运行会出现下列出错信息。

"Only constructors can invoke constructors"

Checket 作为一个构造方法应该在调用时从最老的祖先类开始向下调用，调用 super 会引起程序在编译和运行时间上的错误。

Java 中的关键字 super：调用父类的属性，一个类中如果有 int x 属性，如果其子类中也定义了 int x 属性的话，在子类中调用父类的 x 属性时应用 super.x ＝ 6，表示该 x 是引用的父类的属性，而要表示子类中的 x 属性的话，使用 this.x。this 和 super：在 Java 中，this 通常指当前对象，super 则指父类的对象。若想要引用当前对象的某种东西，比如当前对象的某个方法，或当前对象的某个成员，便可以利用 this 来实现这个目的。当然，this 的另一个用途是调用当前对象的另一个构造函数。如果想引用父类的某种东西，则非 super 莫属。

Java 里在子类中用 super 调用父类构造函数时，调用函数必须放在子类的第一条语句的位置，如果想用 super 继承父类构造的方法，但是没有放在第一行的话，那么在 super 之前的语句，也许是为了满足自己想要完成某些行为的语句，但是又用了 super 继承父类的构造方法，以前所做的修改就都回到以前了，也就是说又成了父类的构造方法了。

答案：A

12.3 this

面试例题 1：下面程序的结果是什么？[Trend 公司面试题]

```
class Tester
{
int var;
Tester(double var)
{
this.var = (int)var;

}

Tester(int var)
{
this("hello");
}

Tester(String s)
{
this();
```

```
System.out.println(s);
}
Tester()
{
System.out.println("good-bye");
}

public static void main(String[] args)
{
Tester t = new Tester(5);
}
}
```

答案：

good-bye

hello

扩展知识（变量的内存分配情况）

在 Java 中有两个非常特殊的变量：this 和 super，这两个变量在使用前都是不需要声明的。this 变量使用在一个成员函数的内部，指向当前对象，当前对象指的是调用当前正在执行方法的那个对象。super 变量是直接指向超类的构造函数，用来引用超类中的变量和方法。因此它们都是非常有用的变量。下面介绍一下 this 的使用方法。

先看下面的一段代码。

```
class PersonInformation
{
  String name,gender,nationality,address;
  int age;
  void PersonInformation(String p_name,String
p_gender,String
      p_nationality,String p_address,int p_age)
  {
name=p_name;
gender=p_gender;
nationality=p_nationality;
address=p_address;
age=p_age;
  }
}
```

在 PersonInformation()函数中这个对象的方法提示可以直接访问对象的成员变量，而且在同一个范围中，定义两个相同的名字的局部变量是不允许的。如果确实想使类的成员变量与方法的参数或方法自己定义的局部变量同名的话，就需要想一种方法使成员变量与跟它同名的方法参数或局部变量区分开来，这就要使用到 this 变量。下面改写一下上

面的代码，使 PersonInformation 类的构造函数的每个参数都有与对象成员变量相同的名字，而成员变量的初值由参数给出。

```
class PersonInformation
  {
String name,gender,nationality,address;
int age;
void PersonInformation(String name,String gender,String
nationality,String address,int age)
{
this.name=name;
this.gender=gender;
this.nationality=nationality;
this.address=address;
this.age=age;
}
  }
```

从上例中可以看出，该构造函数中必须使用 this。this 在方法体中用来指向引用当前正在执行方法的那个对象实例。this 变量的类型总是为包含前执行方法的类。在上例中，我们要区别参数 name 和成员变量 name，写成 name=name 显然是不允许的。在参数或局部变量名与类成员变量名相同的时候，由于参数或局部变量的优先级高，这样在方法体中参数名或局部变量名将隐藏同名的成员变量，因此，为了指明成员变量，必须使用 this 显式地指明当前对象。

有时候会遇到这种情况，全面地访问当前对象，而不是访问某一个个别的实例对象，此时也可以使用 this，并利用 Java 中的 toString()方法（它能够返回一个描述这个对象的字符串）。如果把任何一个对象传递到 System.out.println 方法中，这个方法调用这个对象的 toString 方法，并打印出结果字符串，所以，可以用 System.out.println(this)方法来打印出固有参数的当前状态。

this 还有一个用法，就是构造函数的第一个语句，它的形式是 this(参数表)，这个构造函数就会调用同一个类的另一个相对的构造函数。请看下面的例子。

```
class UserInfo

    {
  public UserInfo(String name)
  {
this(name,aNewSerialNumber);
  }
  public Userinfo(String name,int number)
  {
userName=name;
userNumber=number;
  }
}
```

如果调用 UserInfor newinfotable = new UserInfo("Wayne Zheng")，就会自动调用 UserInfo(String name,int number)构造函数。

可见，熟练掌握 this 在 Java 程序设计过程中是非常重要的。

面试例题 2：以下代码的运行结果是　　。[Advantech 公司 2005 年 12 月面试题]

```
class Base{

int i;

    Base(){
add(1);
System.out.println(i);
}

void add(int v){
i+=v;
System.out.println(i);
}

void print(){
System.out.println(i);
}
}
class MyBase extends Base{
MyBase(){
add(2);
}
void add(int v){
i+=v*2;
System.out.println(i);
}
}
public class TestClu {
public static void main(String[] args) {

go(new MyBase());
//System.out.println();
}
static void go(Base b){
b.add(8);
//b.print();
```

```
}
}
```

A．12　B．11　C．22　D．21

解析：

程序流程是这样的：在主函数中，首先执行 new MyBase()，在这个过程中，子类会首先调用父类的构造函数；在父类的构造函数 Base()中执行 add()方法。这里需要注意，这个 add 方法由于是在新建 MyBase 对象时调用的，将会首先查找 MyBase 类中是否有此方法。所以，Base()函数中的 add(1)实际上是：

```
 void add(int v)
{
i+=v*2;
System.out.println(i);
}
```

此时 i 的值即为 2。在打印两个 2 之后，父类构造函数执行完毕，执行子类的构造函数，即 MyBase()，这里的 add(2)当然也是子类的。i 的值就变为了 6。

最后执行 go 函数，i 即为 22。

答案：C