1 接口 interface implement

接口的所有方法自动地属于public。因此，在接口中声明方法时，不必提供关键字public。

接口可以包含多个方法，接口中可以定义常量。接口中的域将被自动设为public static final。

SwingConstants就是只定义了常量，而没有定义方法，任何实现了SwingConstant接口的类都自动地继承了这些常量，并可以在方法中直接地引用一些常量。

* 1. 接口的特性

接口不是类，尤其不能使用new运算符实例化一个接口

但是可以声明接口变量，必须引用实现了接口的类对象。

使用instanceof检查一个对象是否属于某个特定类一样，也可以使用instance检查一个对象是否实现了某个特定的接口。

每个类只能够拥有一个超类，但却可以实现多个接口

* 1. 接口与抽象类

抽象类也是只定义方法，但是不实现

总点在于继承只能继承一个，实现接口可以实现多个。

1. 对象克隆

当拷贝一个变量时，原始变量与拷贝变量引用同一个对象。也就是说，改变一个变量所引用的对象将会对另一个变量产生影响。

拷贝与克隆



clone方法

clone方法时Object类的一个protected方法，也就是说，在用户编写的代码中不能直接调用它。

1默认的clone方法是否满足要求

2默认的clone方法是否能够通过调用可变子对象的clone得到修补

3是否不应该使用clone

3是默认的

如需要选择1和2

a 实现Cloneable接口类

b 使用public访问修饰符重新定义clone方法

3接口和回调

回调callback是一种常见的程序设计模式。在这种模式下，可以指出某个特定发生应该采取的动作。

4内部类

inner class 是定义在另一个类中的类

需要内部类的原因：

a 内部类方法可以访问该类定义所在作用域的所有数据，包括私有数据

b 内部类可以对同一个包中的其他类隐藏起来

c 当想要定义一个回调函数且不想编写大量代码时，使用匿名内部类比较便捷

4.1 使用内部类访问对象状态

访问外围类的实例类

内部类既可以访问自身的数据域，也可以访问创建它的外围类对象的数据域

内部类的对象总有一个隐式引用（outer），它指向创建它的外部类对象。

4.2 内部类的特殊语法规则

访问外围类引用的正规语法复杂，表达式OuterClass.this 表示外围类引用。

在外围类的作用域之外，可以这样引用内部类

OuterClass.InnerClass

4.3 内部类是否有用、必要和安全

编译器会把内部类翻译成用（$）美元符号分隔外部类名域内部类名的常规类文件。

如果内部类访问了私有数据域，就有可能通过附加在外围类所在包中的其他类访问它们，但做这些事情需要高超的技巧和极大的决心。

4.4 局部内部类

局部类不能用public和private访问说明符进行声明。它的作用域被限定在声明这个局部类的块中。

局部类有一个优势，即对外部世界可以完全隐藏起来。

4.5 由外部方法访问final变量

与内部类相比，不仅能够访问包含它们的外部类，局部类还可以访问局部变量。那些局部变量必须是final

在内部类被首次提出时，原型编译器对内部类中修改的局部变量自动地进行转换。

4.6 匿名内部类

假如只创建这个类的一个对象，就不必命名了。这种累叫做匿名内部类。

4.7 静态内部类

使用内部类只是为了把一个类隐藏在另一个类的内部，可以将内部类声明为static，以便取消产生的引用

5 代理 proxy java1.3

利用代理可以在运行时创建一个实现了一组给定接口的新类。

假设有一个表示接口的Class对象，它的确切类型在编译时无法知道。要想构造一个实现这些接口的类，就需要使用newInstance方法或反射找出这个类的构造器。但是不能实例化接口，需要在程序处于运行状态时定义一个新类。

代理机制是一种解决方案，代理类可以在运行时创建全新的类。这样的代理类能够实现指定的接口。

指定接口所需要的全部方法

Object类中的全部方法，

然而不能再运行时定义这些方法的新代码，而是要提供一个调用处理器（invocation handler）。调用处理器是实现了invocationHandler接口的对象。在这个接口中只有一个方法：

Object invoke(Object proxy,Method method,Object[] args)

无论何时调用代理对象的方法，调用处理器的invoke方法都会被调用，并向其传递Method对象和原始的调用参数。调用处理器必须给出处理调用的方法。

要想创建一个代理对象，需要使用Proxy类的newProxyInstance方法。这方法有三个参数

一个类加载器 作为java安全模型的一部分，对于系统类和从因特网上下载下来的类，可以使用不同的类加载器。

一个Class对象数组，每个元素都是需要实现的接口

一个调用处理器。

这有两个需要解决的问题

如何定义一个处理器?

能够用结果代理对象做些什么》？

路由对远程服务器的方法调用

在程序运行期间，将用户接口事件与动作关联起来。

为调试，跟踪方法调用

代理类的特性

代理类是在程序运行过程中创建的。

所有的代理类都扩展于Proxy类，一个代理类只有一个实例域——调用处理器，它定义在Proxy的超类中。

所有的代理类都覆盖了Object类中的方法toString、equals和hashCode。如同所有的代理方法一样，这些方法仅仅调用了调用处理器的invoke。Object类的其他方法都没有被重新定义。

没有定义代理类的名字，sun虚拟机中的Proxy类将生成一个以字符串$Proxy开头的类名。

代理类一定是public 和 final 。如果代理类实现的所有接口都是public，代理类就不属于某个特定的包，否则，所有非公有的接口都必须属于同一个包，同时，代理类也属于这个包。

可以通过Proxy类中的isProxyClass方法检测一个特定的Class对象是否代表一个代理类。