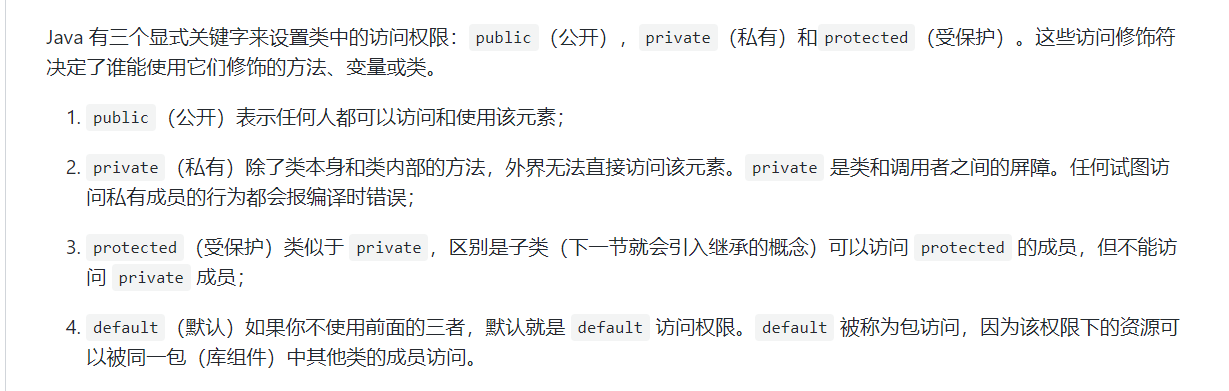
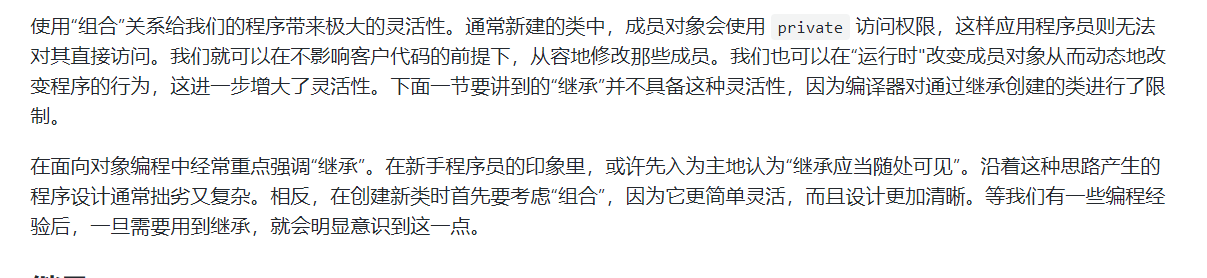
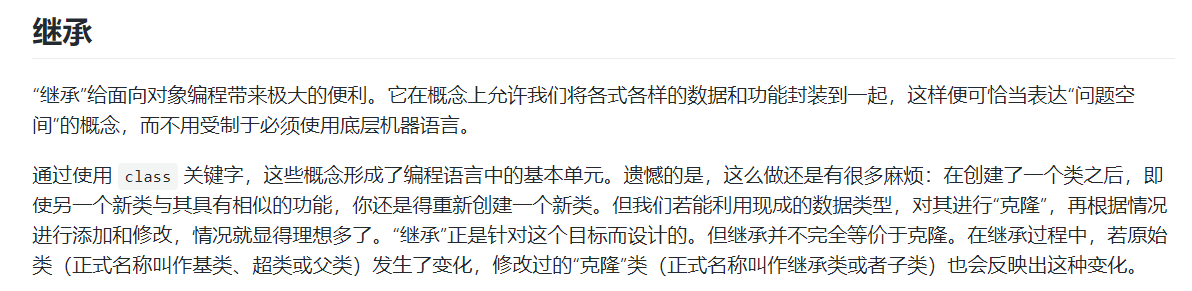
1.



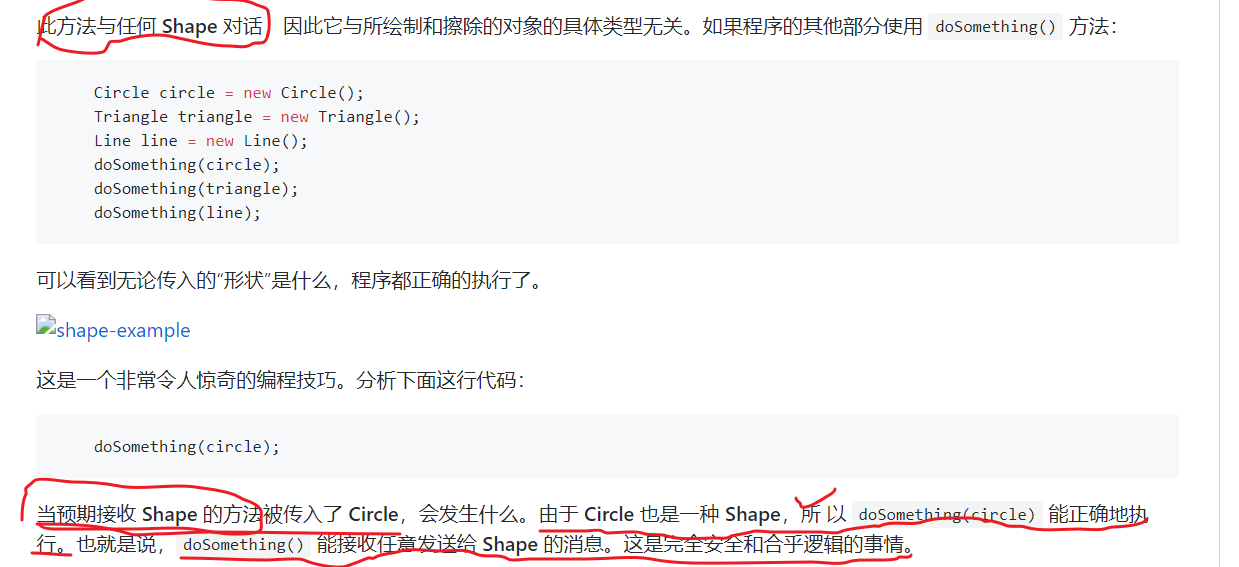
继承是为了实现复用，也是一种轮子思想



多态就是直接对基类如shape定义方法（函数），然后能作用于众多派生类如circle。

“通过继承，程序直到运行时才能确定代码的地址，因此发送消息给对象时，还需要其他一些方案。为了解决这个问题，面向对象语言使用**后期绑定**的概念。当向对象发送信息时，被调用的代码直到运行时才确定。编译器确保方法存在，并对参数和返回值执行类型检查，但是它不知道要执行的确切代码。

为了执行后期绑定，Java 使用一个特殊的代码位来代替绝对调用。这段代码使用对象中存储的信息来计算方法主体的地址（此过程在多态性章节中有详细介绍）。因此，每个对象的行为根据特定代码位的内容而不同。当你向对象发送消息时，对象知道该如何处理这条消息。在某些语言中，必须显式地授予方法后期绑定属性的灵活性。例如，C++ 使用 **virtual** 关键字。在这些语言中，默认情况下方法不是动态绑定的。在 Java 中，动态绑定是默认行为，不需要额外的关键字来实现多态性。”



**Java 单继承结构 优选！**

**Java的集合就是C++的STL**

在 C++ 中，“集合”是其标准库的一部分，通常被称为 STL（Standard Template Library，标准模板库）

不同类型的集合对应不同的需求：常见的有 List，常用于保存序列；Map，也称为关联数组，常用于将对象与其他对象关联；Set，只能保存非重复的值；其他还包括如队列（Queue）、树（Tree）、栈（Stack）、堆（Heap）等等

Java 的异常处理机制在编程语言中脱颖而出。Java 从一开始就内置了异常处理，因此你不得不使用它。这是 Java 语言唯一接受的错误报告方法。如果没有编写适当的异常处理代码，你将会收到一条编译时错误消息。这种有保障的一致性有时会让程序的错误处理变得更容易。值得注意的是，异常处理并不是面向对象的特性。尽管在面向对象的语言中异常通常由对象表示，但是在面向对象语言之前也存在异常处理。