继承

1.super与this不同。this是对象引用，super只是一个指示编译器调用超类的特殊关键字、

2.子类不能直接访问父类中的私有域，需要调配父类方法。同样，子类中对父类私有变量的初始化也需要在构造器中使用super调用父类构造器，并且该语句必须是子类构造器中的第一句。如果子类的构造器中没有显示地调用父类构造器，则会自动地调用父类默认的构造器。此时，如果父类没有无参构造器，编译器报错。

3.重载解析：

对语句X.f(param)，编译器将会一一列举出X类中名为f的方法及其父类中访问权限为public的f方法作为候选。

接下来，编译器会查看调用方法时提供的参数类型，选择类型完全匹配的方法。其中因为涉及到类型转换，所以过程很复杂。如果编译器没有找到与参数类型匹配的方法，或者发现经过类型转换后有多个方法与之匹配，就会报告一个错误。

4.动态绑定与静态绑定：

对于private方法，static方法，final方法或者构造器，编译器可以准确地知道应该调用哪个方法，这种情况叫静态绑定。、

调用的方法依赖于隐式参数(this, 在函数参数列表中的第一个，代表调用者对象，省略不写)的实际类型，并且在运行时实现动态绑定。

采用动态绑定时，虚拟机会调用与X所引用对象的实际类型最相符的那个类的方法。在本类中找不到，则会依次去寻找父类。为减小搜索开销，虚拟机预先为每个类创建一个方法表。

5.子类在覆盖父类的方法时，其可见性不能低于父类。Java中允许Parent parent=new Children();的多态表现形式，若此时允许子类方法权限小于父类，则会出现变量申明为Child类型时不允许访问，申明为Parent时则允许访问，这很显然逻辑不通。

6.内联：

方法调用过程中，每次遇到方法调用表达式，该方法都会进栈，程序的执行转移到该方法对应的内存地址中，执行完毕后，方法弹栈，转回。对一些方法体很短的方法来说，这种开销很大，于是有了内联。内联指的是在程序中，将方法表达式自动替换为方法体执行，这样就不涉及栈操作和内存地址跳转，用空间换来了时间。

在Java中，用final关键字指明一个方法可以做为内联方法，至于是否会被当做内联方法处理，则由编译器分析将final函数处理为内联和不处理为内联的性能比较决定。

7.父类对象引用可以自动接收子类的对象(向上转型)，但子类对象引用如果要接收父类对象，则必须要进行显示的向下类型转换。转换过程中有可能报告错误并抛出ClassCastException异常，因此在类型转换之前要加一个instanceof判断（如果子类引用允许接受父类对象，那么在调用子类才有的方法时，实际的对象是父类，并不存在这个方法，当然会报错）

8.equals和==

在比较对象是否相等时，==比较的是两个对象的引用是否相等，如果i对象相同则值一定相等。对于equals方法，在基类Objects中，是直接使用==实现，但是很多类都重写了equals方法，使得它变成了对值的比较，比如String类。

==可以比较基本数据剋型，而equals只能比较对象。