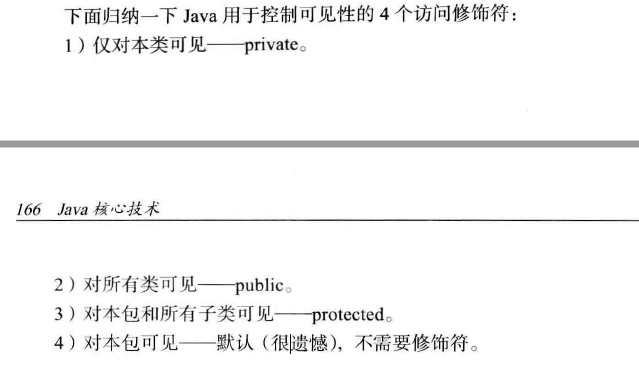
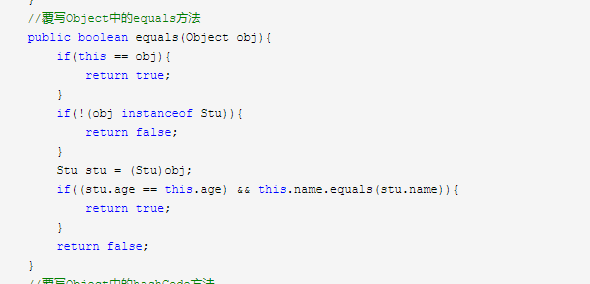
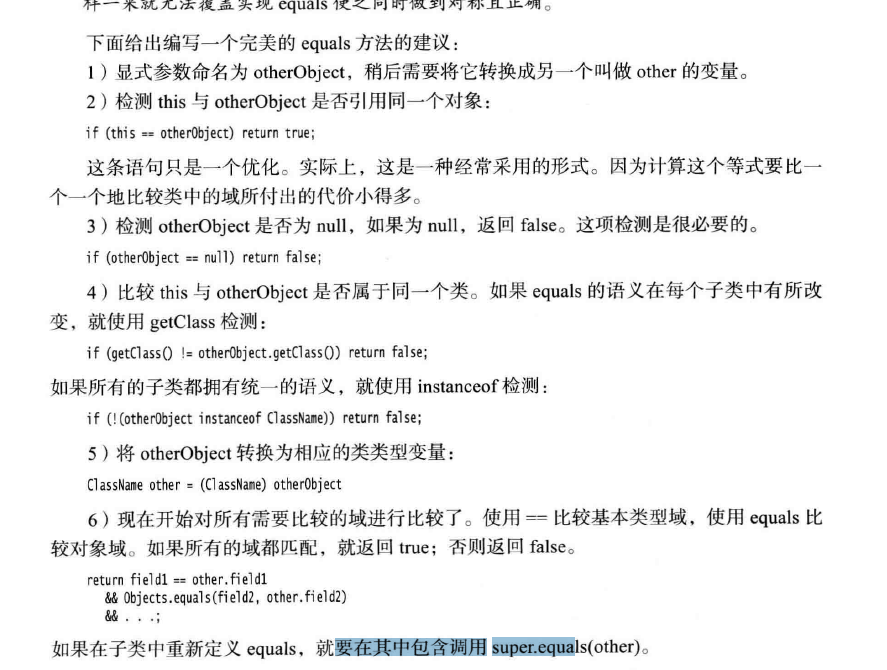
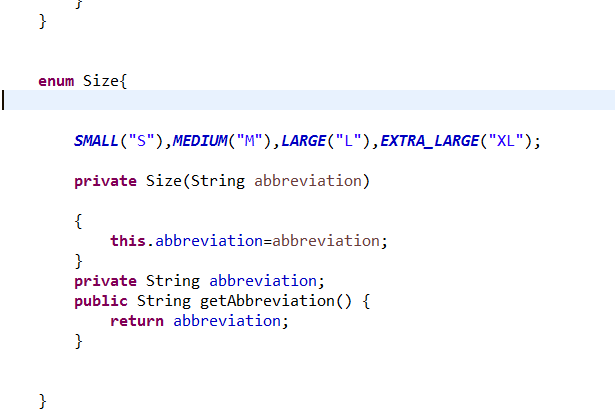
1. 一个对象变量可以指示多个实际类型的变量这种现象叫做多态 在运行时能动态的选择调用对象的哪个方法 是叫动态绑定
2. 子类会继承父类的属性 但唯独构造方法不会被继承 所以子类必须有自己的构造法方法 并且可以调用super方法去调用父类的构造方法，进行复用。子类有自己的属性的话要自己单独进行初始化
3. 用来判断是否适合用继承 就是判断两者关系是否符合is—a的关系
4. 超类不仅仅可以用自身来表示 也可以用子类来表示
5. 子类不可以调用父类的私有方法
6. 不是**private** 方法、**static** 方法或**final** 方法的方法 虚拟机会产生动态绑定的机制
7. 在覆盖一个方法的时候 子类的可见性一定不能比父类的可见性范围要小
8. 进行类型转化是因为 可以暂时忽略对象的实际类型 并可以用对象的所有功能
9. 在进行类型强制转化时 要考虑是否是在继承机制之类 而且将超类转换成子类时 要用instanceof 进行检测
10. 只要有一个抽象方法的类都要被声明为抽象方法 抽象类可以拥有具体的的数据属性和相应的方法
11. 类即使不含抽象方法，也可以将类声明为抽象类 将它变成不可实例化的效果
12. 抽象类将具体的类表现形式 上升到抽象的层面 给写代码也带来了一定的便捷性
13. 

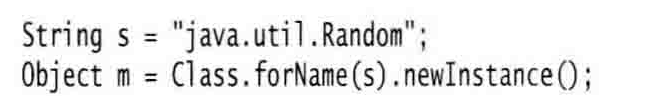
Public>protected>default>private

1. object的equals方法 用于检测两个对象是否相等 是否具有相同的引用
2. 
3. Equals方法需要注意的5点:
4. 自反性:对于任何非空引用x，x.equals(x)始终返回true
5. 对称性:对于任何引用x和y,当且仅当y.equals(x)返回true x.equals(y)才会返回true
6. 传递性:对于任何引用x,y,z如果x.equals(y) 返回true y.equals(z) 返回true x.equals(z)也返回true;
7. 一致性:如果x和y引用的对象没有发生变化，反复调用x.equals(y)应该返回同样的结果
8. 对于任意非空引用 x.equals(null) 应该返回false;
9. 
10. 重写tostring方法 可以用getclass().getName() 获取类名 这样可以共享给子类使用 子类只要调用super.toSting() 可以进行复用
11. 自定义类的两个对象是否相等 根据自己重写的equals方法来判断
12. ArrayList 增删改查 add remove set get
13. 类包装器:Integer Long Float Double Short Byte (派生自number类)Character Void Boolean
14. 可变参数的实现f(double...values)
15. Enum.valueof



1. 把能够分析类能力的程序叫做反射

反射可以用来:

1. 在运行时分析类的能力
2. 在运行时查看对象 例如编写一个toString方法供所有类使用
3. 实现通用的数组操作代码
4. 利用Method对象
5. 通过反射获取一个实例 e.getClass().newInstance();原理：调用默认的无参构造方法进行创建对象
6. 反射检测类的结构
7. 接口中的所有方法自动的属于public 在接口中定义方法中可以不用写public
8. 接口中的域会被自动设置为 public static final 书写接口时 只要按照返回值+方法就行了 这种也是比较推荐的模式