今天看了《Java编程思想》的前7章，晚上看了《Effective Java》的第一章

收获如下：

1、终止方法finalize()在垃圾回收执行前调用，其使用条件是只负责回收通过某种创建对象之外的方法为对象分配的存储空间，通常就是指调用C/C++“native方法”为对象分配的空间，如果本地方法中没有调用free()释放堆空间，就需要在finalize()中调用;

2、弄清楚了Java垃圾回收的基本原理：从堆栈和静态存储区开始，遍历所有对象引用，找到所有“活”的对象，需要进行整理；程序内存空间中剩余的对象为“非活”对象，需要被回收。垃圾回收通常只对上次回收动作之后新分配的块进行整理（对短寿命对象很有帮助）；同时垃圾回收定期执行完整的回收动作，通过监控碎片数量，自适应地在“停止-复制”和“标志-清除”两种回收方式进行切换（为了降低回收处理花销）;

3、《Effective Java》第一章介绍了对象创建与销毁的7条程序设计规则，其基本思想是可以看懂的，但还是由于缺少Java编程经验的原因，其实自己还未遇到需要使用其中一些规则的情形，尽管看了之后遇到对应情形会遵照这些规则，但对于这些规则的必要性以及带来的好处还不能深刻的理解。

4、对“规则1:使用静态工厂方法替代构造器”这一规则的理解。

应当使用静态工厂方法的情形：

1. 某个类需要有多个带标签的构造器，用静态工厂方法的函数名进行区分更清晰；
2. 不需要每次都生成类的新对象实例而只需使用类的可重用实例的情况，使用静态工厂方法实现更为简洁；
3. 对于需要的对象类型与参数值有关的情形，使用静态工厂方法可以灵活的返回原返回类型的子类型；
4. 创建参数化类型实例时使用静态工厂方法，借助类型推导可以让声明代码更简洁；
5. 需要通过API获取非公有类的实例对象，使用静态工厂方法实现会使API非常简洁（还理解不了）；
6. 应用于服务提供框架的情形（还理解不了）。

不应该使用静态工厂方法的情形：

当某个类需要子类化时，不能只提供静态工厂函数，而不提供非私有构造器。