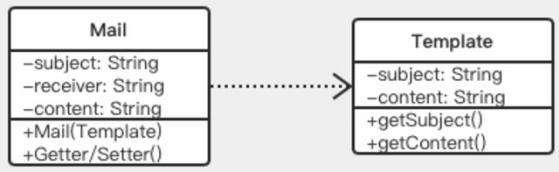
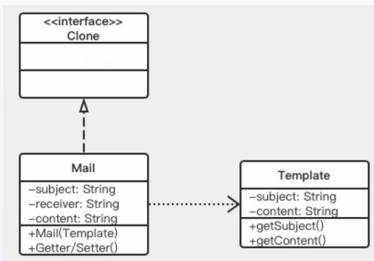
- 给用户发送邮件
- 假设发送一个邮件0.02s
- 1000w：50h以上
- 解决方法之一：多线程
- 问题：线程1邮件没发送完，线程2已经改了信息



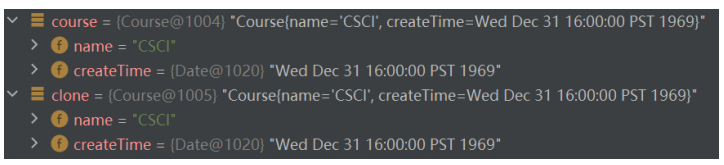
- 用一个已经创建的实例作为原型，通过复制该原型对象来创建一个和原型相同或相似的新对象

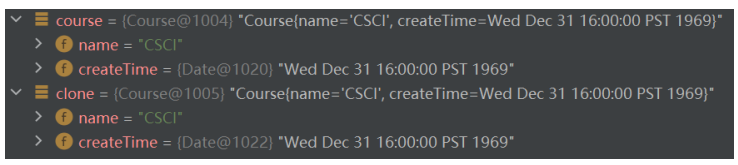


- 构造函数不会被执行
- 所有的类都会继承Object类
- Object类的clone方法的原理是从内存中以二进制流的方式进行拷贝

深克隆 v.s. 浅克隆

- 浅克隆：只clone本对象，对象内部元素不clone
- 深克隆：内部元素单独克隆





其他考点

- 单例模式：调用getInstance方法 (重写 clone（）方法时)
- final冲突：使用clone，不要用final

- Java 自带的原型模式基于内存二进制流的复制，在性能上比直接 new 一个对象更加优良
- 逃避构造函数的约束

- clone 方法位于类的内部，当对已有类进行改造的时候，需要修改代码，违背了开闭原则
- 当实现深克隆时，需要编写较为复杂的代码，而且当对象之间存在多重嵌套引用时，为了实现深克隆，每一层对象对应的类都必须支持深克隆，实现起来会比较麻烦

使用环境

- 一个对象多个修改者的场景
- 通过new产生一个对象需要非常繁琐的数据准备或访问权限