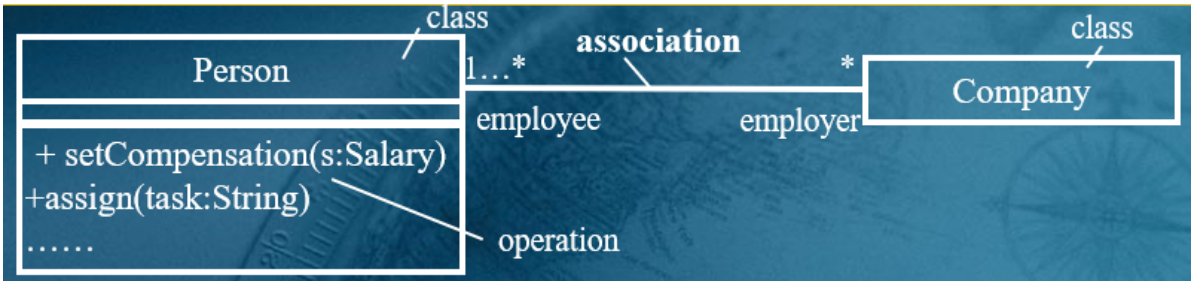
UML notes 03 Interaction Diagram

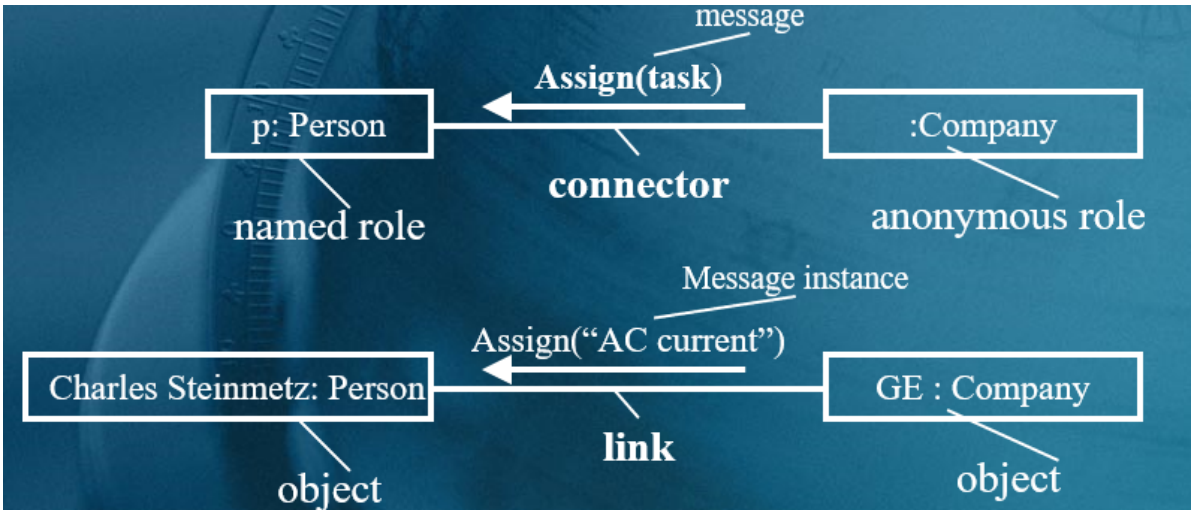
collected by wxb

1 Intro of Interaction Diagram

- interaction 交互: 表示上下文中的一组对象之间交换消息，以实现某个目的的行为
- interaction diagram 交互图: 用来描述对象之间、对象与actor之间的动态合作关系，描述协作过程中的行为顺序
    - sequence diagram 顺序图：侧重于信息的时序
    - collaboration diagram 协作图：侧重于在对象的某些结构组织的上下文中的消息排序

  交互图描述的是对象之间的信息交互，不是类之间的信息交互
- message 信息：预期活动发生时，对象之间传递的信息
- roles 角色：代表类、接口、组件、节点和用例的原型实例
    - 它们的动态方面被可视化、指定、构建和记录为控制流 flow of control
- Link 连接：对象之间的语义连接，是 association 的实例
    - link 指定了一个对象给另一个对象/自己发送消息的路径
    - 修饰词：association, self, global, local, parameter
- connector：是 link 的原型





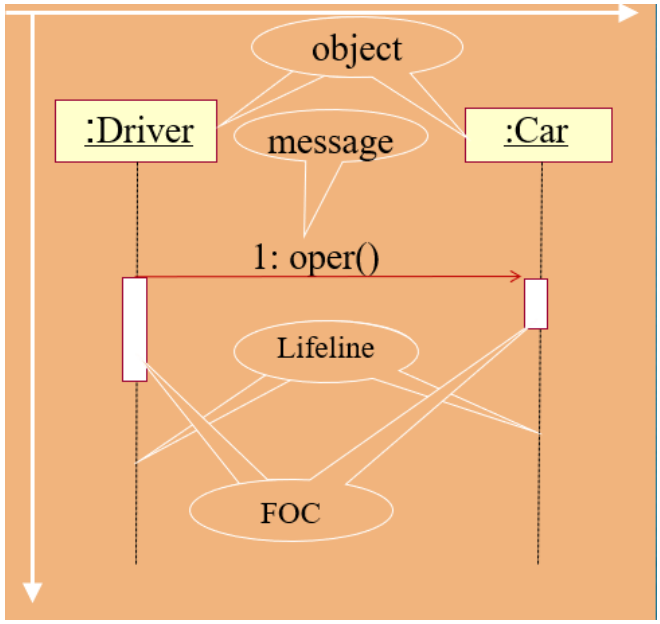
- context 上下文

2 Sequence Diagram

- 定义
  - 显示对象如何根据消息序列相互通信
  - 指示与这些消息相关的对象的生命周期
- 表现方式
  - 平行垂直线：不同的进程或对象同时存在
  - 平行箭头：不同的进程或对象之间的消息传递（按照发生顺序）
- 特征：从上到下表示时间流逝

绘图方式

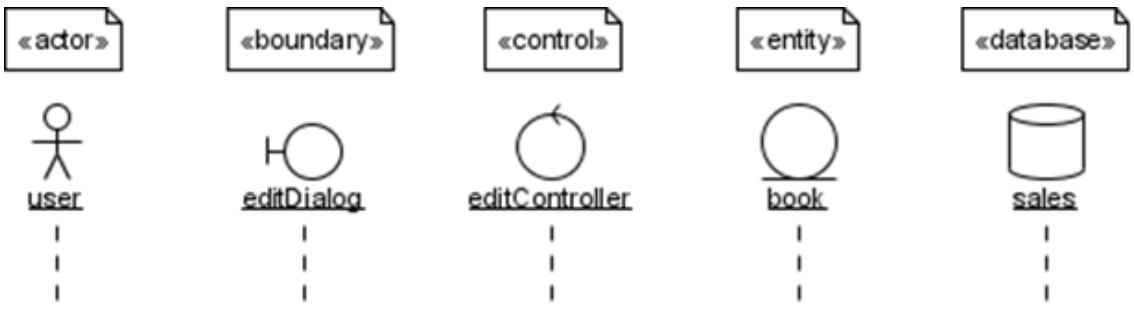
四个基础的建模元素：object　lifeline　focus of control: FOC　message



object

- 命名规则
  - `objectName: ClassName`
  - `: ClassName`
  - `objectName`

  使用 object name的情况
  - 希望在交互中引用指定对象时，要加上`objectName`
  - 没有提到 type / class
  - 相同类型已经存在其他的匿名对象，只能通过命名进行区分
- 除了由类实例化得到的object以外，actor, boundary, control, entity, database 也可以作为对象

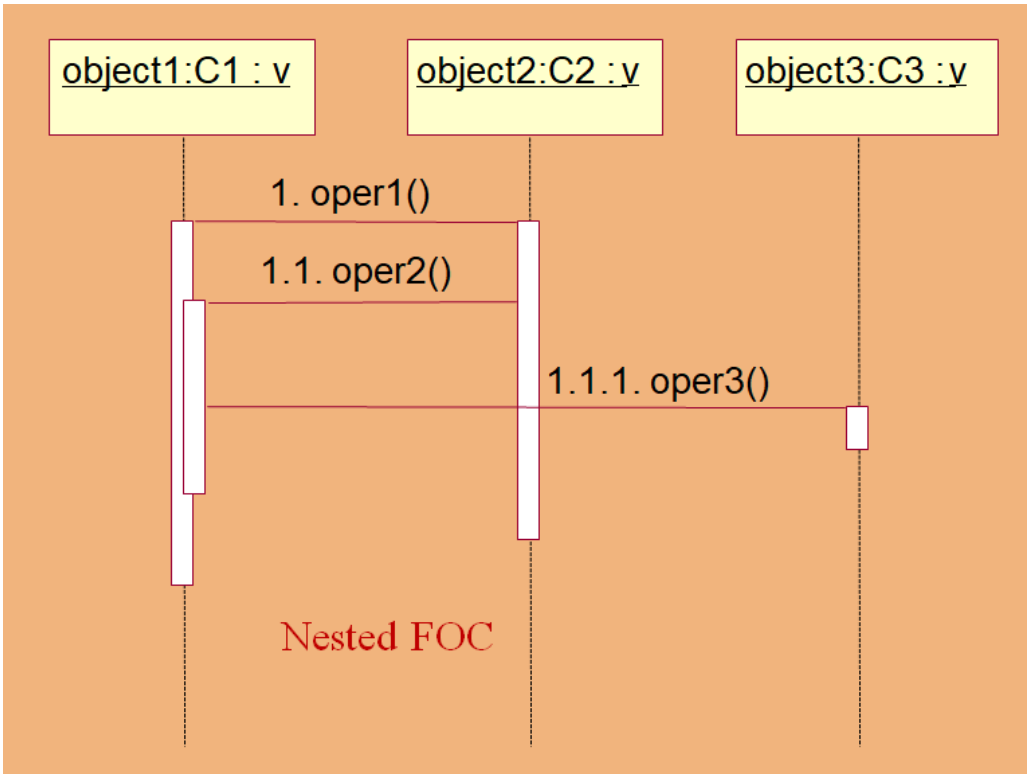


lifeline

- 表示对象的生命范围
- 用点线表示，从object出发，从上到下

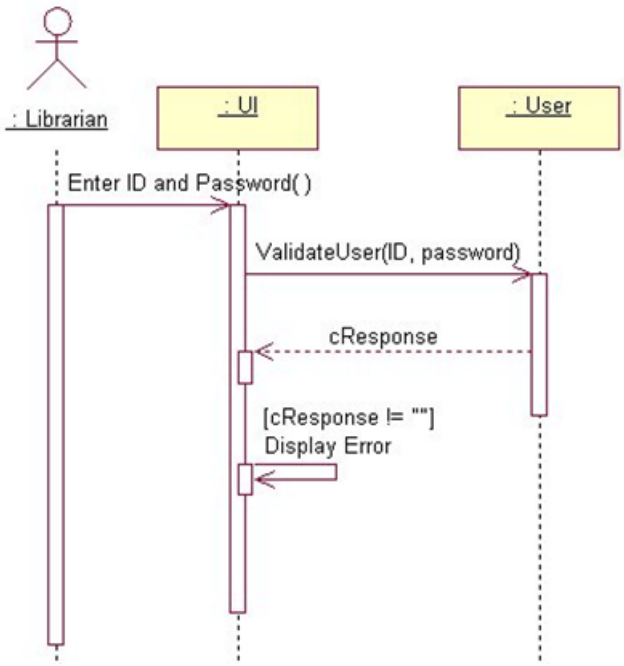
FOC

- 表示某个对象在执行操作的一段时间
- 用细长方形表示，要画在lifeline上
- FOC的顶部和底部用于消息的发送和接收操作
- FOC 可以嵌套，以更准确地表示消息的开始和结束位置



message

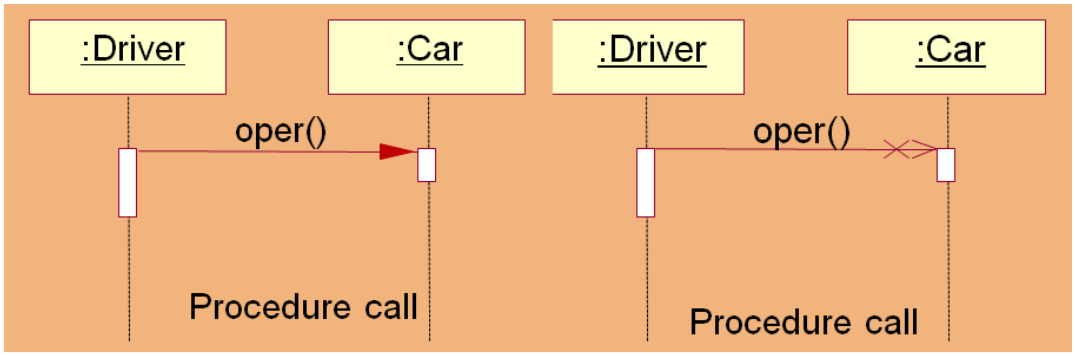
- sequence diagram中最重要的元素
- 表示一个对象何时调用另一个对象（或自身）的操作，还可以用于表示返回值
- message 从左上角的对象（通常是actor）开始，沿着垂直轴从一个对象流向另一个对象
- 当message被发送时，消息文本需要指明接收消息的对象要被调用的方法



- 分类
  - procedure call 过程调用

    消息发送方在接收到返回前会处于wait状态

    使用封闭的实心箭头表示

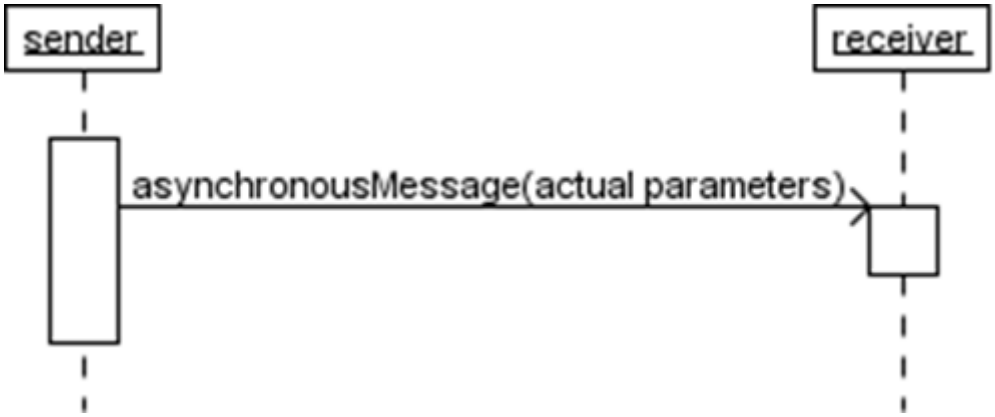


  - asynchronous message 异步消息

    消息发送方在发送消息后继续执行，不等待接收方的处理

    一旦消息被接受，发送方和接收方会同步工作

    使用半箭头表示

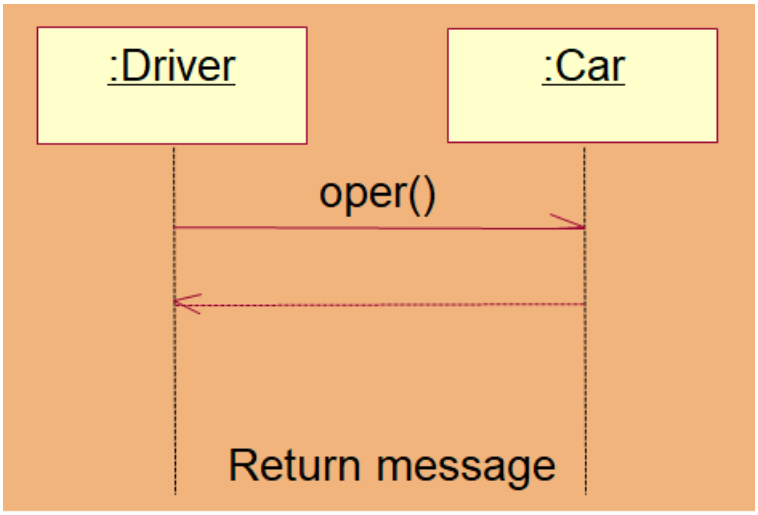


  - return message 返回消息

    表示从 procedure call返回的消息，返回值可以被隐藏

    如果不是procedure call，且有返回值，则必须画出来

使用虚线箭头返回



- syntax 语法

  [predecessor] [guard-condition] [message-expression] [return-value:=]

  message-name ([argument-list])

  - predecessor: 必须在前面发送的消息列表

    `sequence-number "," ... "/"`

  - guard-condition: 只有保护条件被满足，消息才会被发送

    `'[' boolean-expression ']'`

  - message-expression:

    sequence-term `'.' ... ':'`

    `[integer | name] [recurrence]`

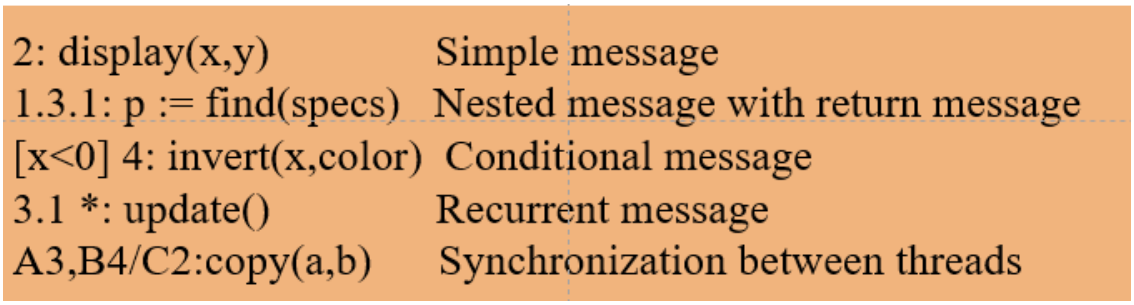
    integer：表示message no

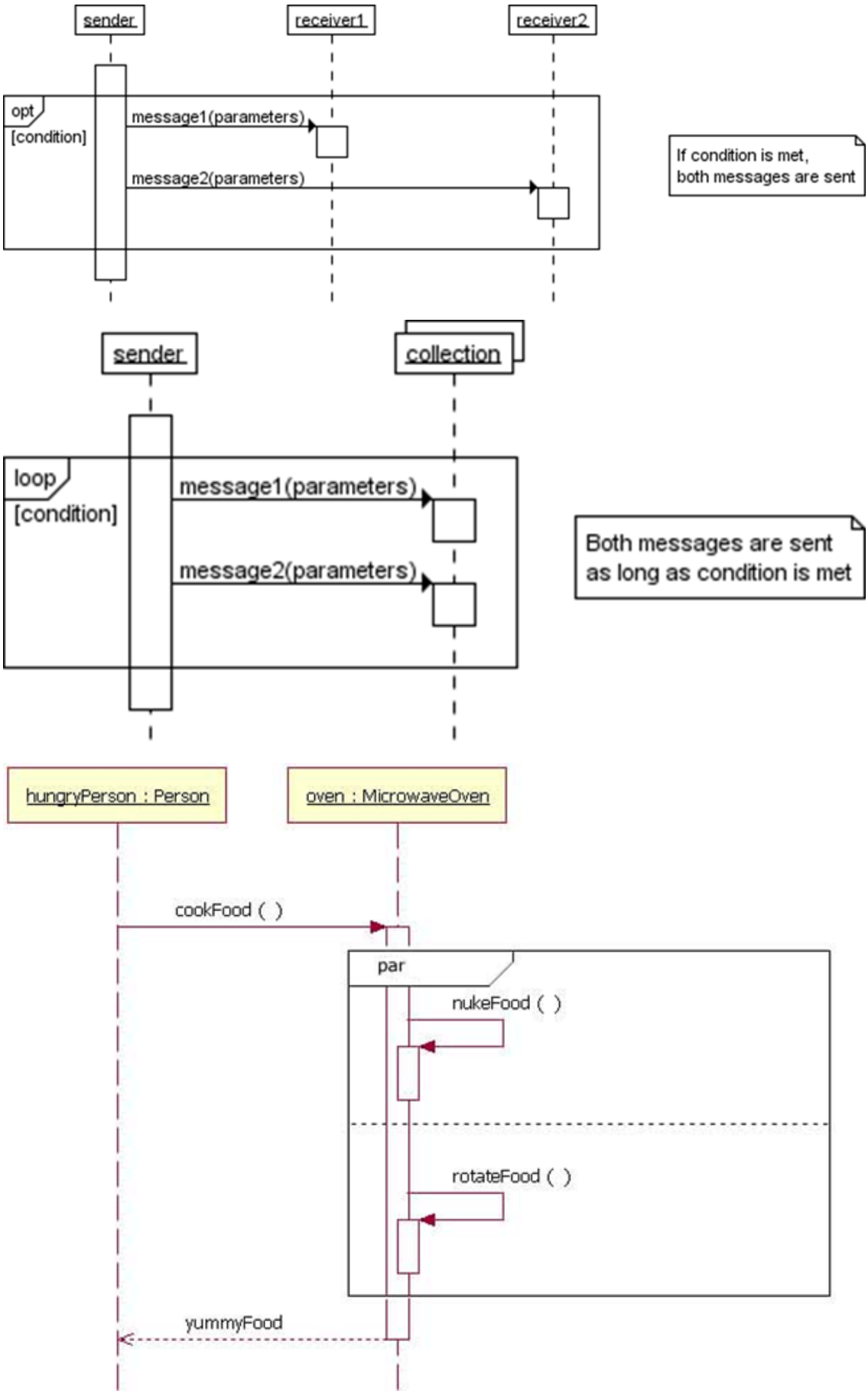
    name: concurrent control thread 并发控制线程

    recurrence：表示条件执行或循环执行

    - 循环：`*[i:=1...n]`
    - 条件：`[x>y]`

  - return-value: 参数列表





3 Collaboration / Communication Diagram

- 根据有序消息描述对象之间的交互，描述了系统行为是如何被组件协作实现的
- 表示方法：object + link + message

- message 要写在link附近，一个link可以有多条message
- message 用时间顺序编号，从 1. 开始