

软件工程

大连理工大学软件学院



第5章 类的分析与设计

- 对未来系统的功能进行总体上的概括并使用UML的类图进行表达。
- 在开始阶段粗略的对模型进行构建，后续再通过迭代逐级具体化，是一个逐步求精的设计过程。
- 初始类图要覆盖所有需求的功能，并通过优化尽量保持业务结构的稳定，然后通过修订和丰富细节逐渐过渡到详细设计，并最终转化为成功的物理实现。

基本类的确定

- 设计阶段的主要任务是从需求分析阶段的规格说明出发，对系统进行模型表示并优化。
- 面向对象的概要设计首先寻找系统中参与业务处理的对象和类。
- 然后使用类图（**Class Diagram**）将系统中不同的类抽象出来描述系统的静态结构，包括类以及它们之间的关系。

类及其种类


- 在系统分析与设计阶段，类通常分为**实体类（Entity Class）**、**控制类（Control Class）**和**边界类（Boundary Class）**：
 - 实体类**：对应需求中的实体，通常需要永久保存，一般使用数据库表或文件来记录，既包括存储和传递数据的类，还包括操作数据的类。（**名词、POJO**）
 - 控制类**：用于体现应用程序的执行逻辑，提供相应的业务操作，抽象控制类可以降低界面和数据库之间的耦合度。控制类有时也称为管理类。（**动宾**）
 - 边界类**：边界类用于对外部用户与系统之间的交互对象进行抽象，主要包括界面类以及与外部系统的数据交换类(如同步、缓存等)。
- 在分析设计初始，通常首先识别出实体类，绘制初始类图，也可称为**领域模型**。

类的识别


- 类的寻找和细化是迭代的过程，不断补充新类及信息并逐渐扩展，最后发展为更多的类和实例变量。
- 需求规格说明书是寻找业务类的直接来源。
 - 一种比较快速而实用的分析方法是按照语法分析的方式将**名词作为对象的候选**，**形容词作为属性（实例变量）的候选**进行重点关注。
- 业务术语词汇表也是类信息的重要来源，这些与业务术语相关的类通常为实体类。

举例

- 需求R1.1“项目创建”的描述：在项目编辑中，系统必须提供给用户新项目的创建以及为其指定具体项目信息的功能。
 - 词汇“项目信息”：自动生成的唯一项目编号、项目名称、项目起止时间、预计工作量。
- 在首次的迭代中主要关注的是类及其属性。
 - 通过以上需求和词汇描述的分析，下面的内容会被首先识别出来：项目类，含有项目编号、项目名称、项目起止时间以及预计工作量等属性。

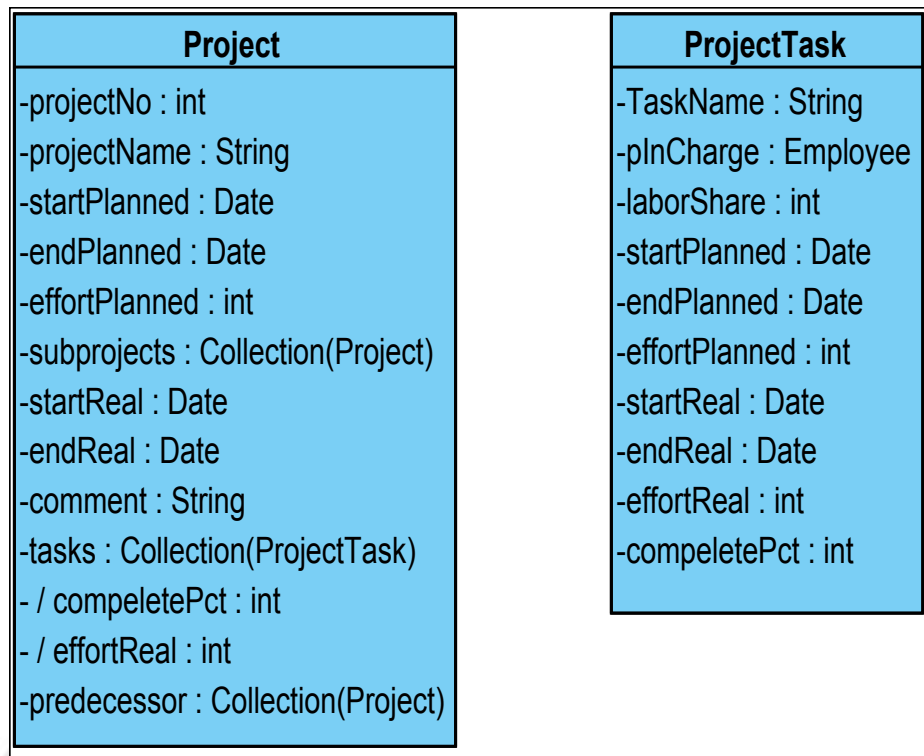
- 
- 需求R1.4子项目创建：选定项目后，系统需要提供给用户为所选项目创建子项目的功能。
 - 子项目为项目的一个实例变量，并最终可能成为一个新的子项目类，可以将其暂记为一个备选类。
 - 需求R1.5子项目与项目：在项目编辑过程中，系统对子项目的处理方式与项目应该是一样的，对项目提供的编辑功能，子项目也必须具有。
 - 子项目与项目是同义词，所以子项目不需要单独设置一个类而存在。
 - 对于同义词有“异形同义”的情况，还有“同形异义”的情况，这在需求分析阶段已经进行了标识。

- 需求R1.6项目信息编辑：选定项目后，系统必须提供给用户对该项目数据编辑的功能，包括实际开始时间、最新计算出的结束时间、预计工作量以及项目备注等。
 - 可发现以下信息：项目类的实例变量还应包括实际开始时间、最新计算的结束时间、备注。
- 需求R1.7项目任务添加：选定项目后，系统必须提供给用户对该项目添加具体任务的功能，包括任务名称、计划开始和结束时间、人员安排以及该任务的预计工作量等内容。
 - 词汇“项目任务”：项目中包含的原子任务，具有名称以及具体的责任人对应，具有可量化的工作量比例，具有计划与实际工作量、计划与实际的开始和结束时间以及完成进度等属性，是不可再分的项目管理单元。

- 
- 词汇“完成进度”：每次编辑操作后对项目任务的完成进度通过百分数进行标识。此数字在一般情况下应呈一种递增的线性的增长方式。项目的进度是根据其子项目以及任务的进度，以预计工作量值作为权重计算出来的。
 - 词汇“工作量”：每次编辑操作作为项目任务记录此任务花费的时间（小时）。整个项目的工作量根据每个子项目和任务的工作量进行核算。

- 发现的信息：项目类的“任务”属性、项目任务新类及其属性：名称、责任人、工作量比例、计划的工作量、实际工作量、计划和实际的开始时间、计划和实际的结束时间、完成进度比例。
- 项目类的实例对象同样也具有完成进度比例以及实际工作量等属性，即使它的值能够通过其它相关子项目或者子任务完全计算出来。把这样的属性称为**依赖属性**。对于项目属性“任务”的另外的特殊之处在于其取值的数量可以是多个或者在少数的时候取空值，不像其它属性只能取一个单一的值。

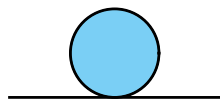
初始类图



- 类名
- 实例变量
- 可见性
 - +, -, *, ~
- 依赖（计算）属性
- 类型（可忽略）
 - UML预定义
 - 编程语言提供

实体类的便捷表示

- **<<entity>>**



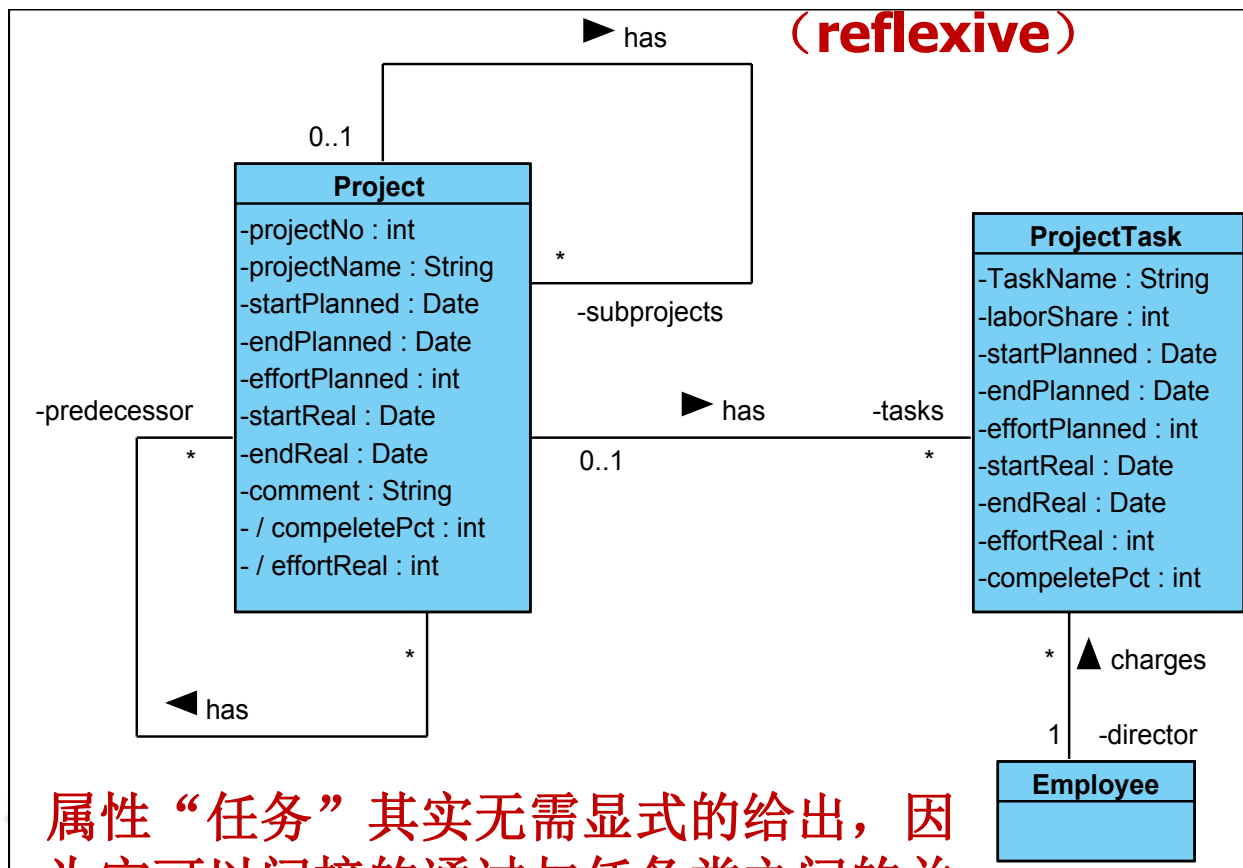
Project

- projectNo : int
- projectName : String
- startPlanned : Date
- endPlanned : Date
- effortPlanned : int
- subprojects : Collection(Project)
- startReal : Date
- endReal : Date
- comment : String
- tasks : Collection(ProjectTask)
- / completePct : int
- / effortReal : int
- predecessor : Collection(Project)

类的关系

自反关联

(**reflexive**)



属性“任务”其实无需显式的给出，因为它可以间接的通过与任务类之间的关联关系进行体现。

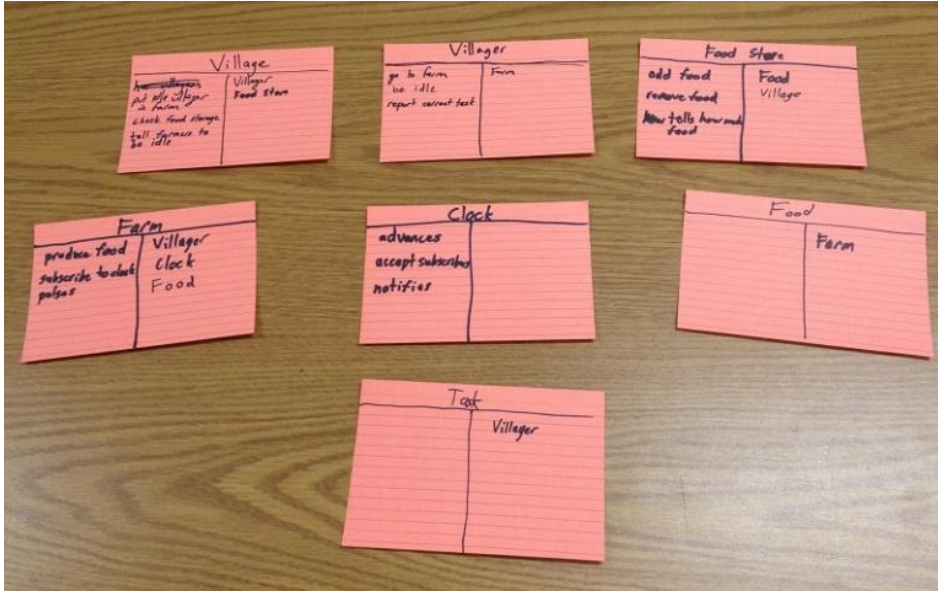
Class or String?

- 关联关系，**Association**，静态，拥有
- 导航方向，**Navigation**
- 依赖关系，**Dependency**
- 应避免双向依赖

关联关系的基数（多重性）

- “*”：任意多个（包括0个）对象
- “1”：只有1个对象
- “3”：正好3个对象
- “1..*”：最少1个，也可能为多个对象
- “3..*”：至少3个，也可能为多个对象
- “0..1”：0或1个对象
- “3..7”：3到7个对象

CRC方法

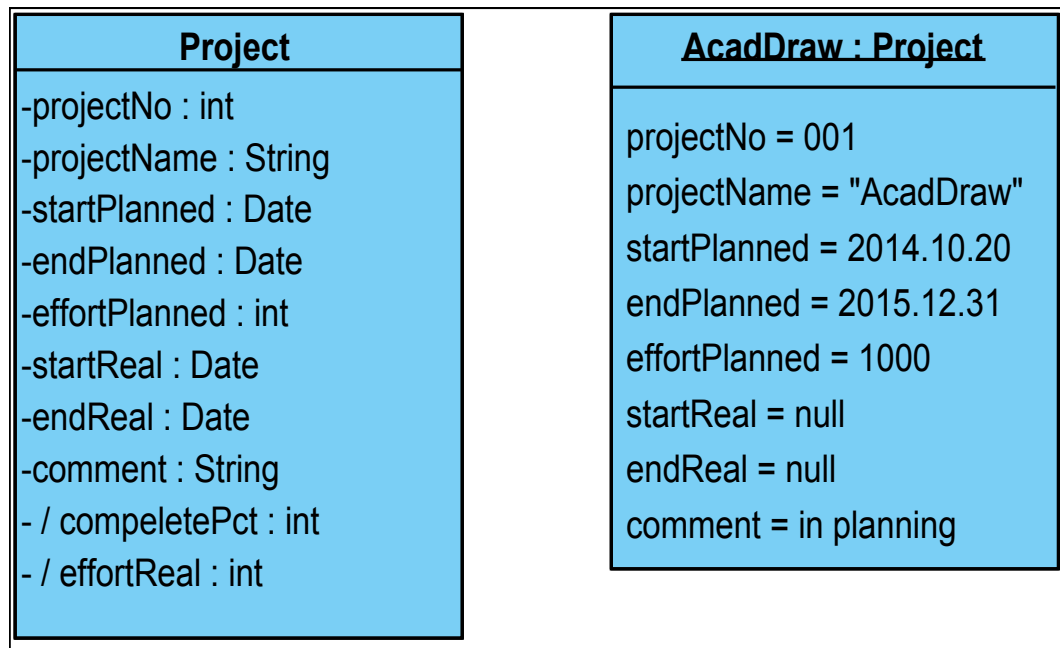


- CRC(Class Responsibility Collaborators)
- 类图的创建和调整是一个高度动态的过程
- 卡片上主要记录类的名字、含有的属性以及主要职责、有消息交互的其他类等
- 类之间的关系也便于绘制，可将不同版本的类图拍照存档

对象与类

- 系统中的每个对象在表示上具有唯一的**标识ID**以及通过其属性进行描述。
- 比如一个具体的项目名字为“考勤系统”，项目经理名“王楠”，项目开始日期“2010年10月20日”。这些属性称为**实例变量（instance variable）或属性（attribute）**。
- 同类对象的共同结构可通过类进行说明，除了类名外，所有的实例变量都可包含在类中作为类的初始信息。因此对于“项目”类来说，它是所有具体“项目对象”的一个**模板**。

对象与类的表示



- 连接类与对象间的实线，表示类图中关联关系的实例化
- 对象名:类的类型
- 实例变量的初始值

类的细化

- 在下一轮的迭代中，将重新审视并分析需求陈述和词汇表中提到的功能与对象之间的对应关系。
- 除了实例变量的说明，类中还包含方法，又称为操作或对象功能，它们为业务计算或对实例变量值的读写提供了服务。
- 一个对象中所有实例变量值的组合构成了该类的状态集合。

方法和管理类

- 访问和修改方法，不涉及业务，在分析模型中通常不考虑，实现阶段再考虑。
- 对象通常还提供了只需通过内部信息，如实例变量，对业务数据进行计算的方法。
 - 如方法`computeAllocatedEffort()`，计算已经对项目中的任务和子项目分配的工作量。
- 对于同类对象的协调和管理通常使用一个管理类，主要负责对对象的创建、代理访问其它对象的信息等。
- 管理类必须能够提供所管辖所有对象统一的处理方式。

控制类

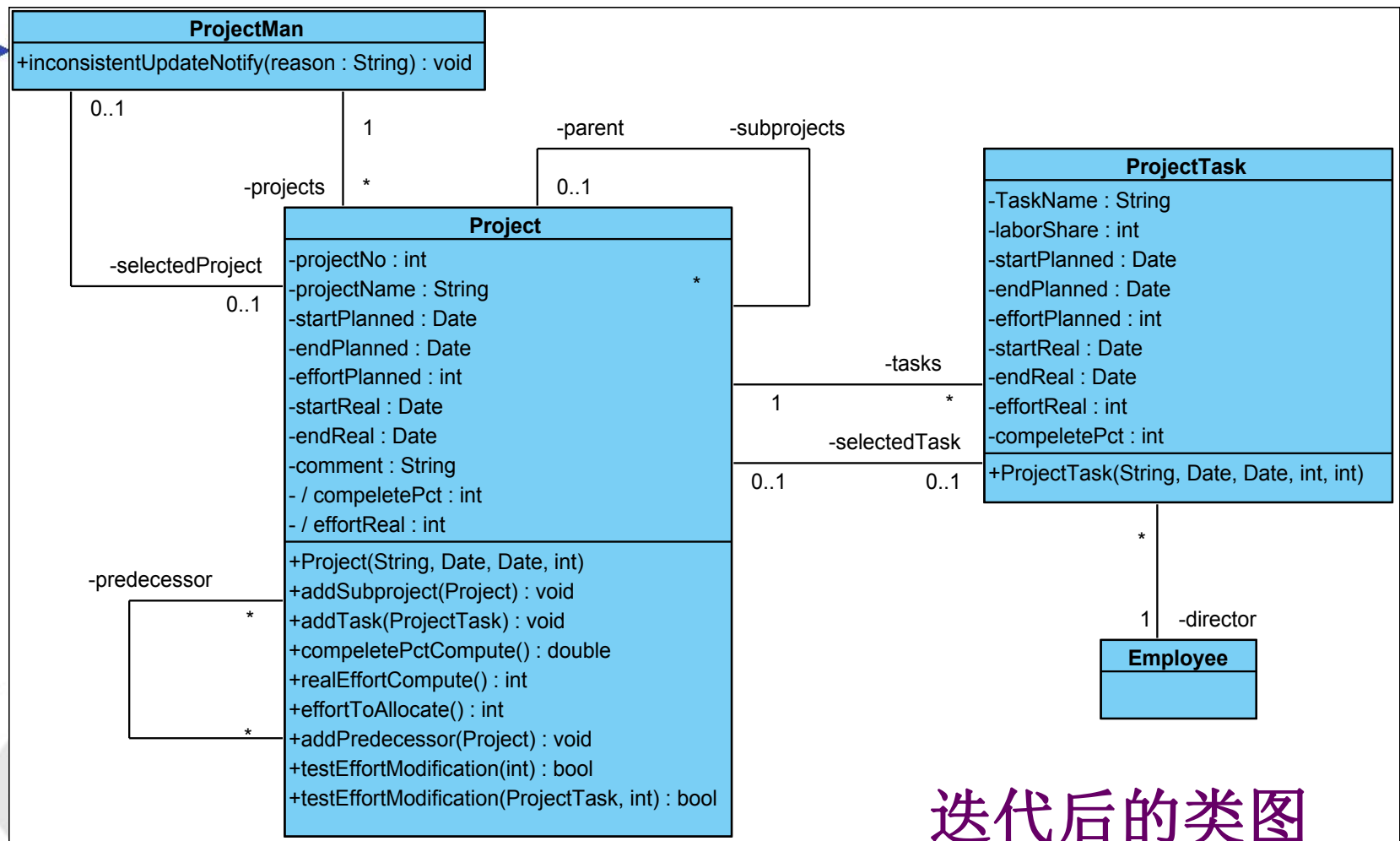
- 控制类通常控制和协调不同对象的行为，用来封装用例的特有行为。复杂用例一般都需要一个或多个控制类。
- 借助控制类可将边界对象与实体对象分开，让系统更能适应其边界内发生的变更。
- 控制类还将用例所特有的行为与实体对象分开，使实体对象在用例和系统中具有更高的复用性。

控制类的识别

- 一般方法：先对所有的用例进行分析，对每个用例对应产生一个控制类，用来对该场景中需要的对象进行管理和协调。
- 控制类每次考虑一个任务，只向控制类添加与该任务相关的方法和该方法需要的实例变量。
- 类与类之间尽可能保持较少的联系，这样可以降低接口的数量。

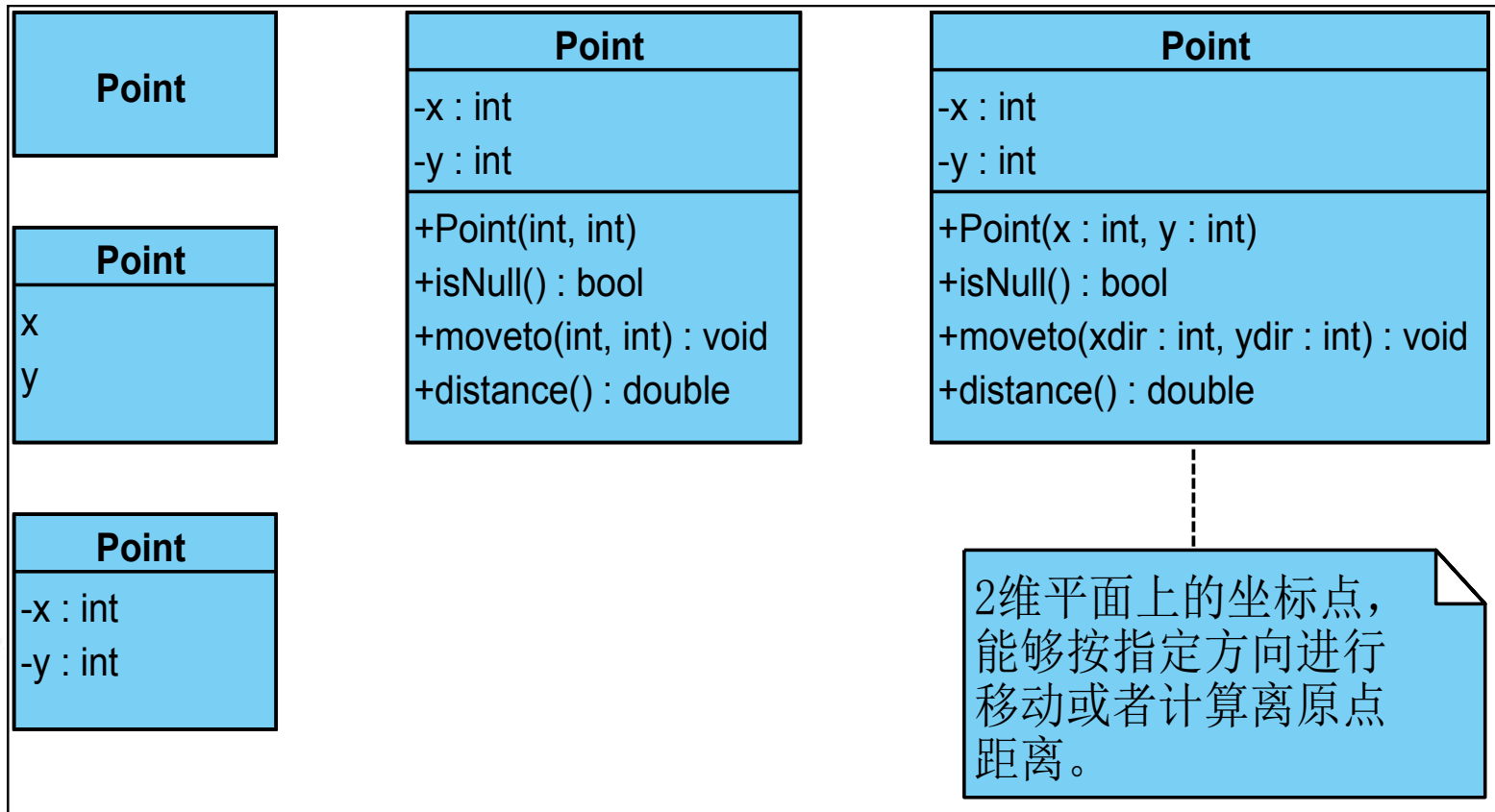
进一步识别和补充类及方法

- **R1.4 子项目创建**: 选定项目后, 系统需要提供给用户为所选项目创建子项目的功能。
- 集合的常用操作**add**和**delete**方法对元素添加和删除。对于本需求, 需要在项目类中加入一个函数
addSubproject(Project): void, 并约定如果返回值为空值表示创建子项目的过程没有成功完成。
- **R1.10 对其他项目的依赖**: 选定项目后, 系统必须提供给用户对该项目与其他(子)项目的依赖关系的编辑功能, 说明它们之间的逻辑关系。
- 项目类应具有方法**addPredecessor(Project):void**。

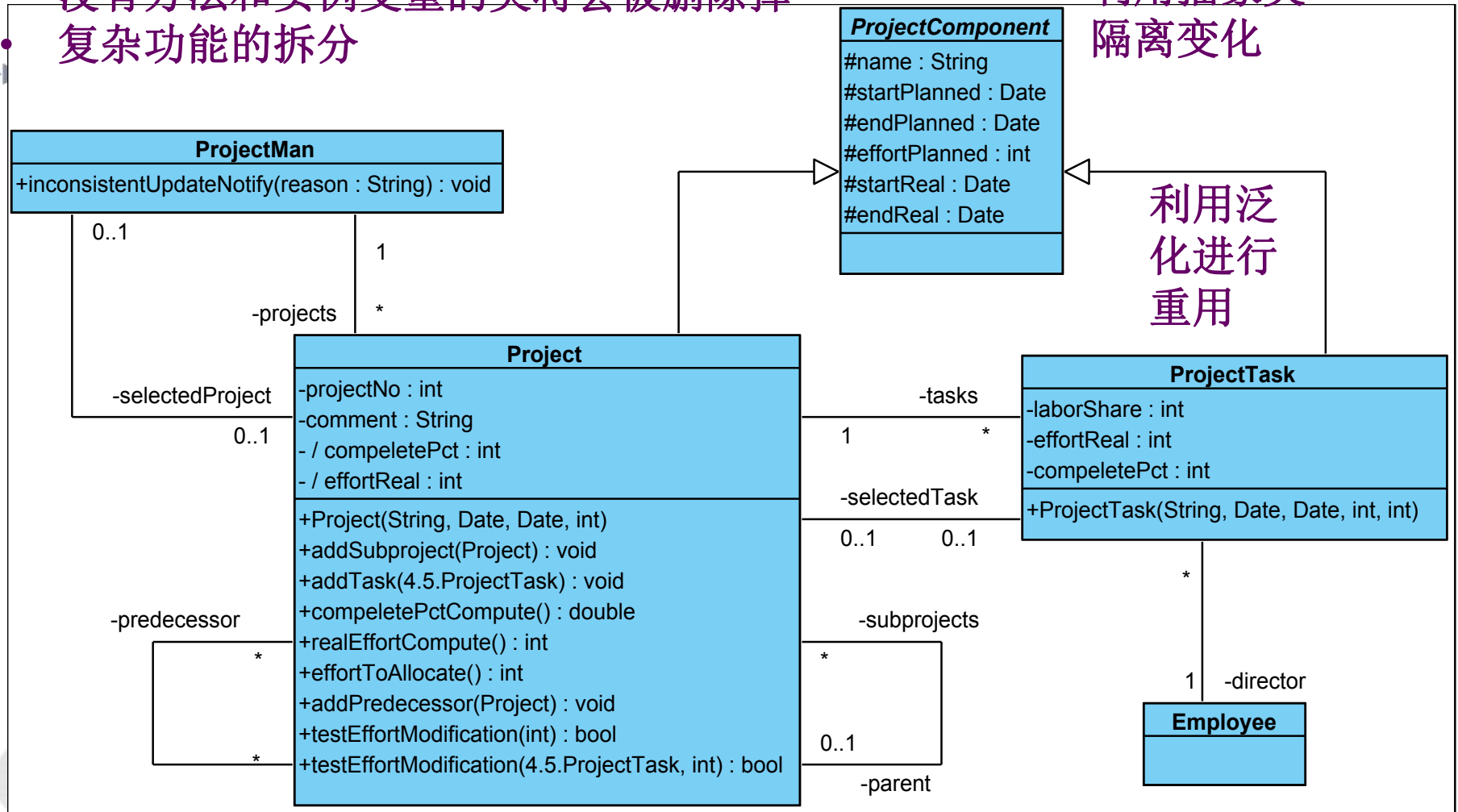


迭代后的类图

类图的不同表示方式



- 优化可以提高模型的易理解性
- 没有方法和实例变量的类将会被删除掉
- 复杂功能的拆分

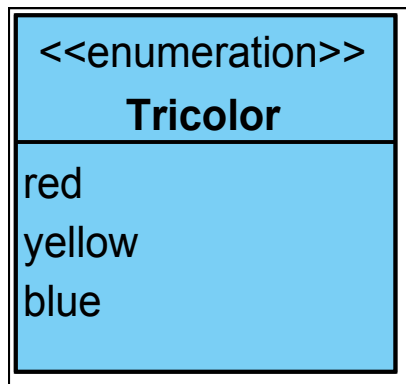


利用抽象类
隔离变化

利用泛
化进行
重用

设计优化

枚举类



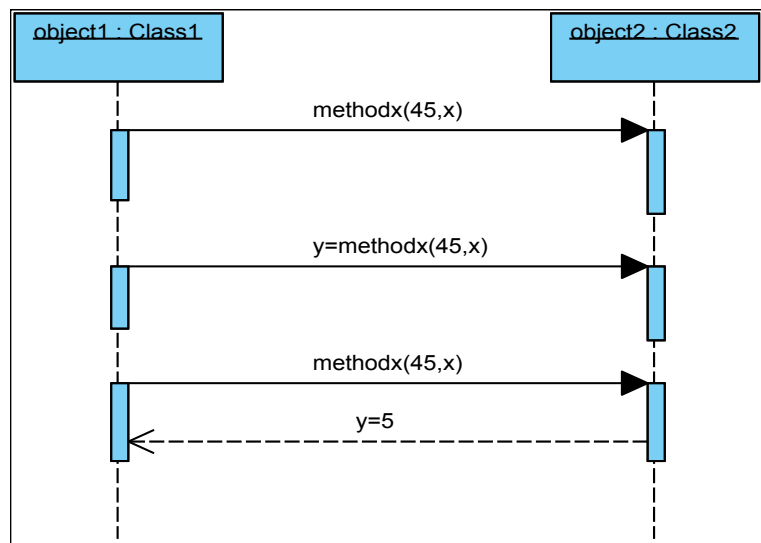
- 如果一个变量的取值是某个有限集合中的数据，如“红色”、“黄色”，“绿色”等，应该使用一种叫做枚举的类型而不是直接使用String类型。
- 如图中所示的枚举类，其具有一个构造型`<<enumeration>>`描述，在它的实例变量部分列举的数据为该类型可能的取值。

补充和确认

- 当初始版本的分析类图完整的构建出来后，需要确认是否需求中的所有信息在模型中都得到了体现而没有遗漏。
- 可使用UML中的顺序图对需求场景中涉及到的不同对象之间的交互过程进行建模。
- 类图在UML中是一种静态图，因为描述了系统的功能侧面，而基于类图的顺序图可以用来设计对象之间的动态交互过程，描述对象之间的过程调用顺序和关系。
- 通过顺序图可以用来检验类图中说明的功能是否能够实现活动图中描述的功能需求。

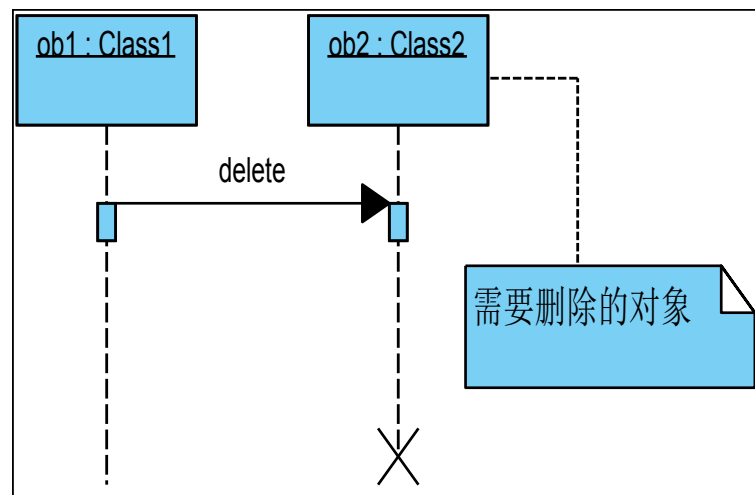
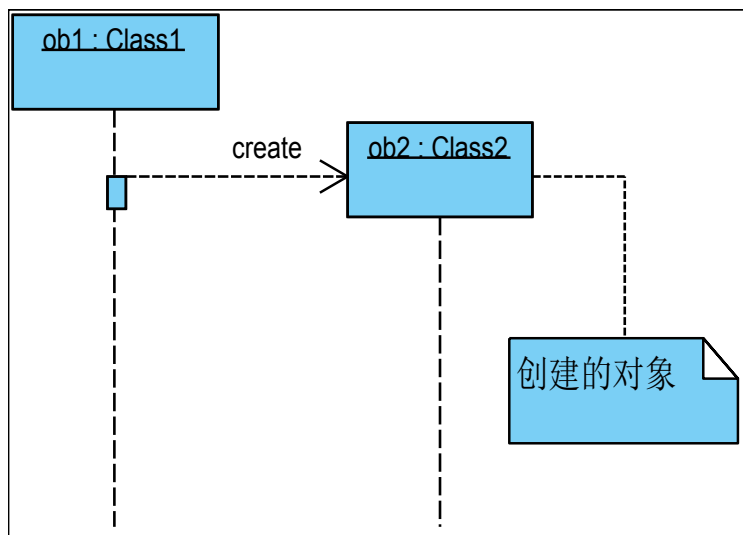
顺序图

- 对象轴、时间轴
- 同步调用的方式和表示
- 生命线、控制焦点（激活区域）
- 同样可以使用通信图（协作图）进行交互建模

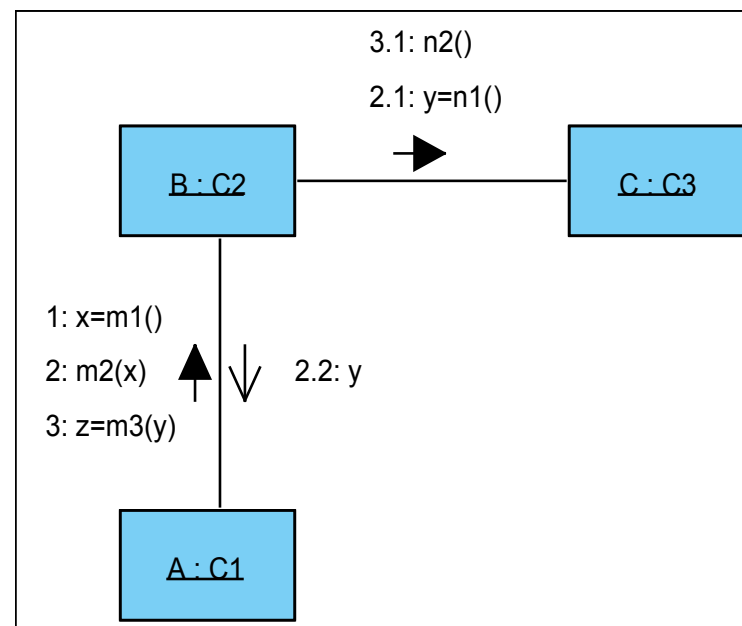
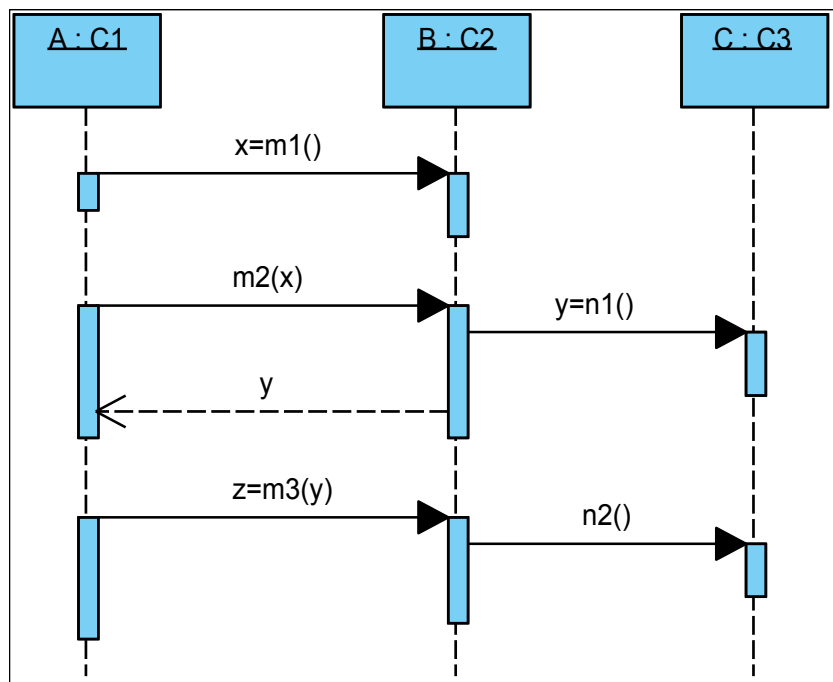


对象的创建与删除

- 左图是对象的创建过程，箭头所指是一个新创建的对象，注意此对象名并不在最上方的位置出现。
- 右图是对象的删除过程，通过在生命线上的X符号表示对象在内存中被回收。



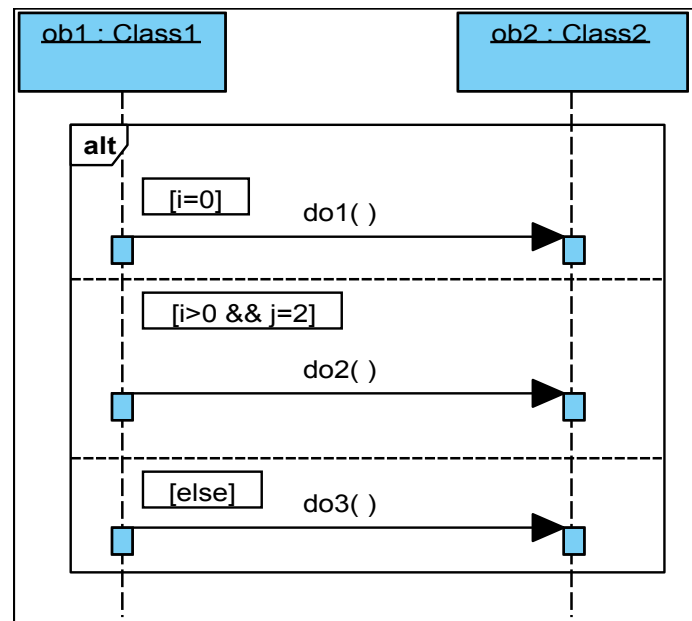
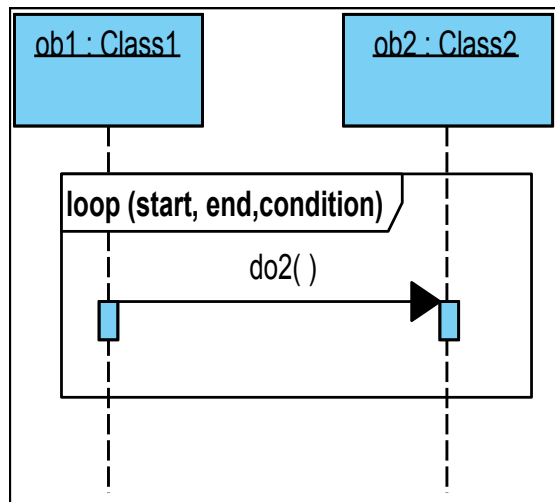
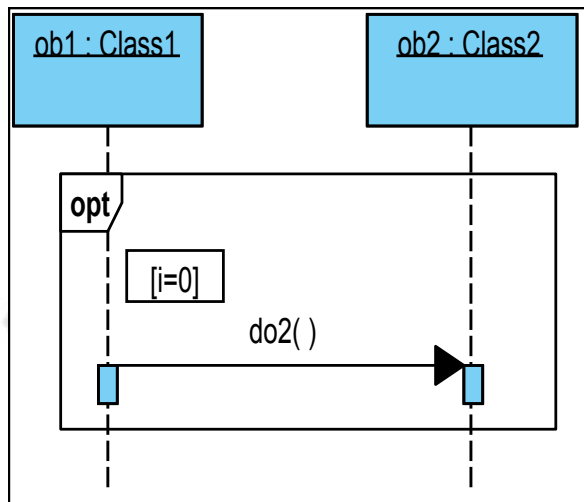
通信图



- 对象表示方式相同，对象间的关联
- 消息发送的顺序、嵌套和表示
- 关注对象间关联结构，与顺序图等价

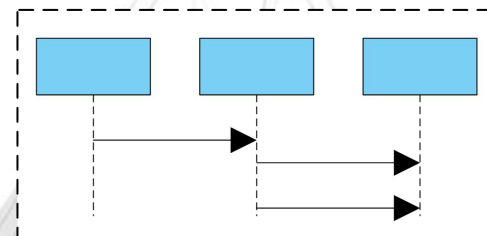
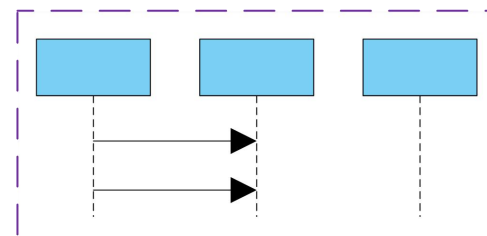
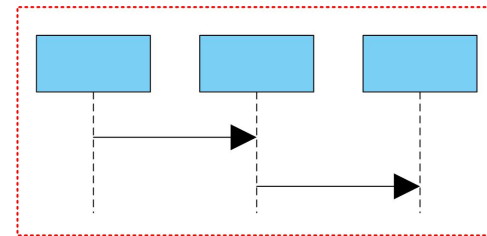
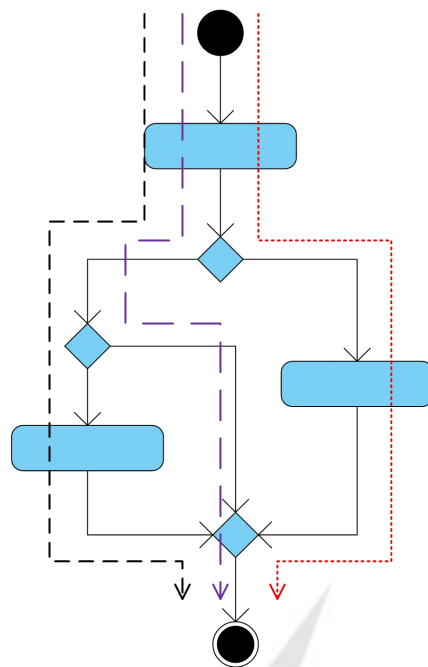
结构表示

- 顺序图某些部分使用矩形框封闭描述，左上角指定一种处理方式。
- “opt”为可选的内容，表示在满足方括号条件的情况下，对应部分就会被执行，否则跳过。
- “alt”对多分支的条件进行选择，矩形框内各分支用虚线分割。每个分支一个布尔条件，应彼此排斥。
- “loop”为循环结构，这里必须清楚的给出循环执行的参数，如循环次数和结束条件。



场景模拟

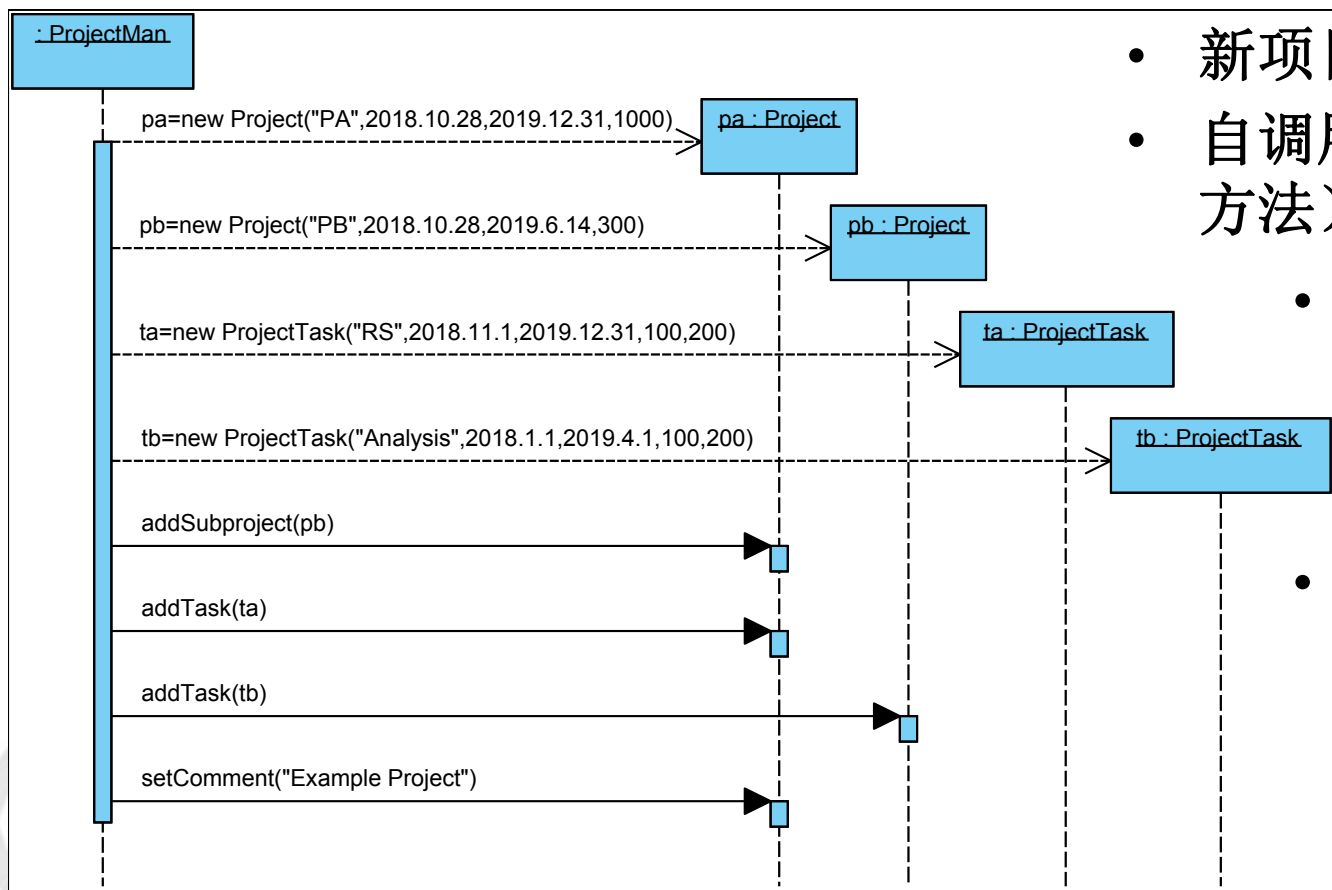
- 对于每个活动图的过程尝试使用一个顺序图进行描述。
- 图中活动图存在3个过程，验证的目标是要确保每条活动图中的边都要被执行到。
- 为使子过程不必多次在每个顺序图中重复描述，可以对一个或多个动作创建子图。



验证的作用

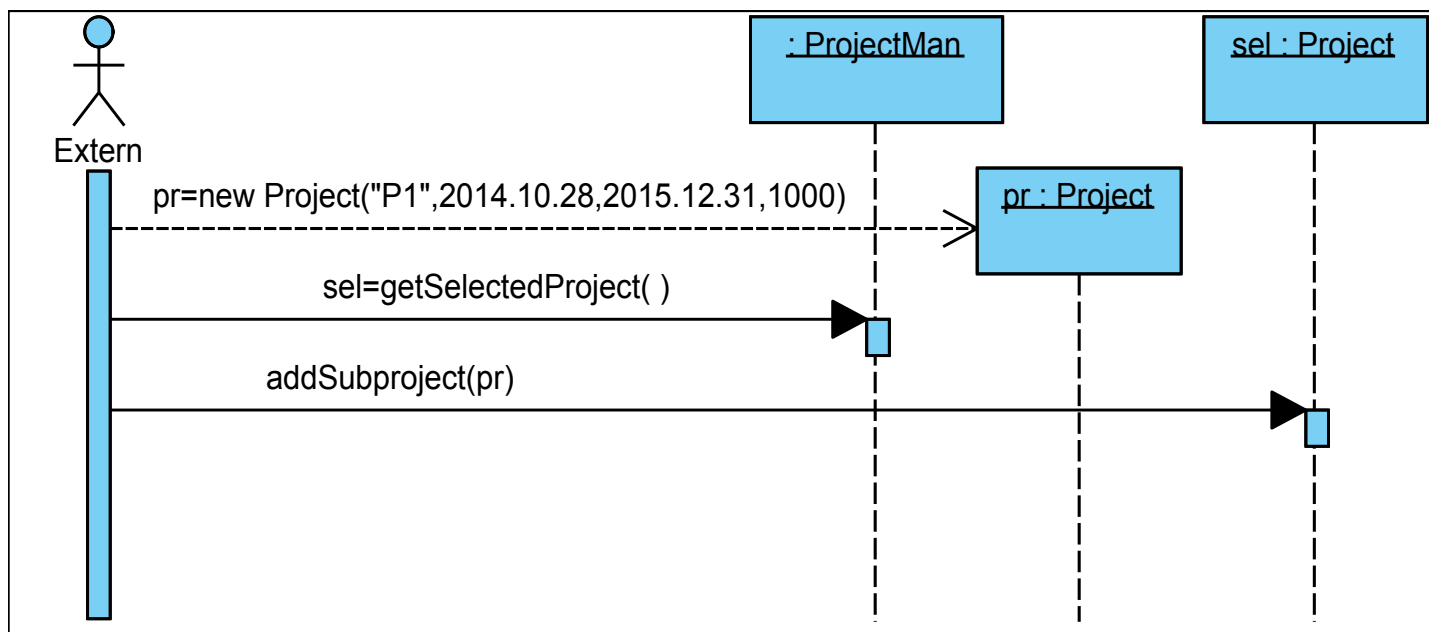
- 将需求中重点描述的过程进行建模，包括对新对象的创建过程。
- 确保所有的功能需求在分析模型中都得到了体现，而非功能性需求则主要是在系统架构中体现。
- 分析类模型和顺序模型的构建也是迭代的过程，完善顺序图时可能会发现新的必要的类方法，从而对类图也进行了补充。

举例

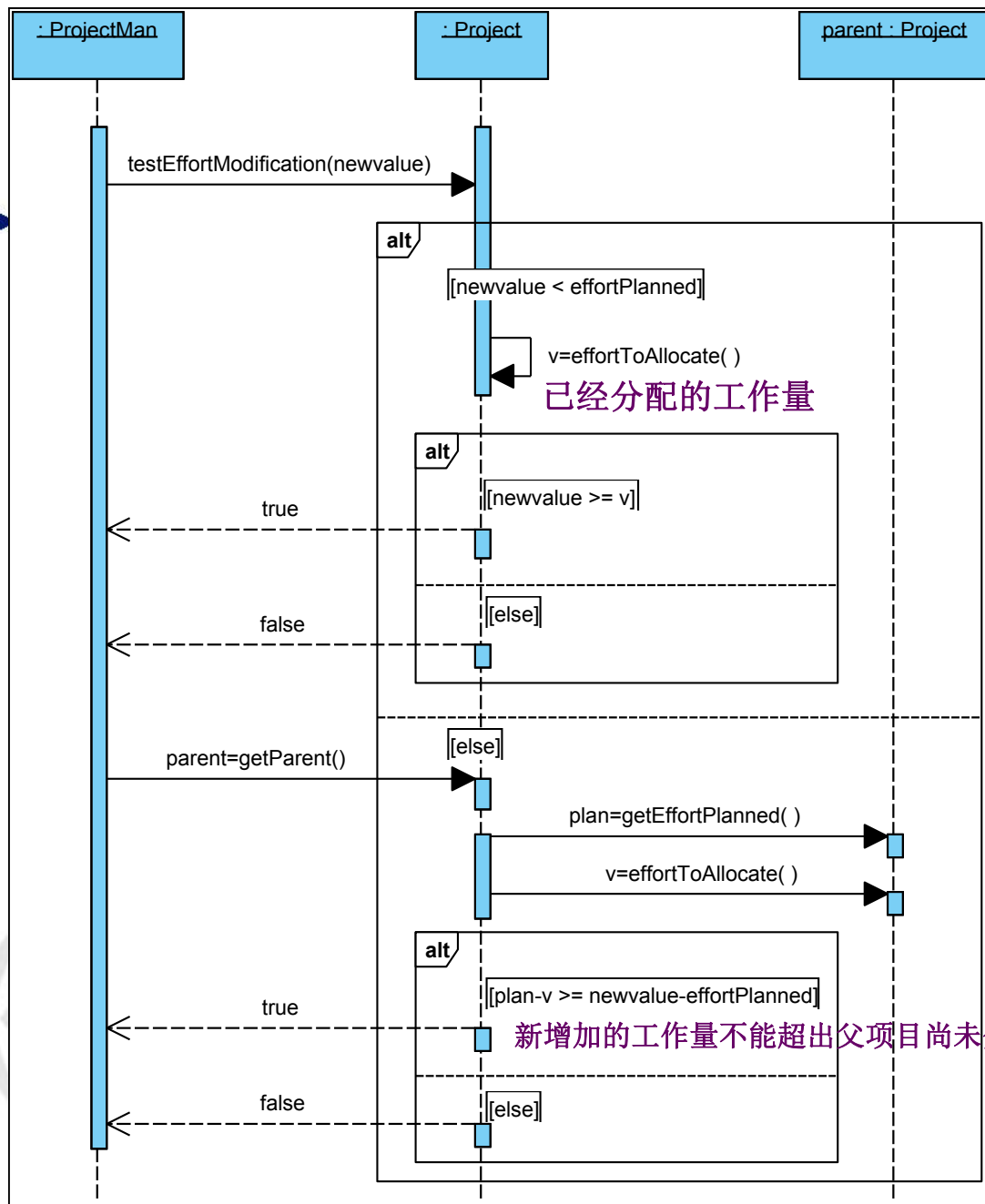


- 新项目的创建过程
- 自调用（类内的私有方法）
- 项目管理对象可以是包含在图形界面中的一个对象
- 图形界面中的对象逐步补充，因为与编程语言相关

举例

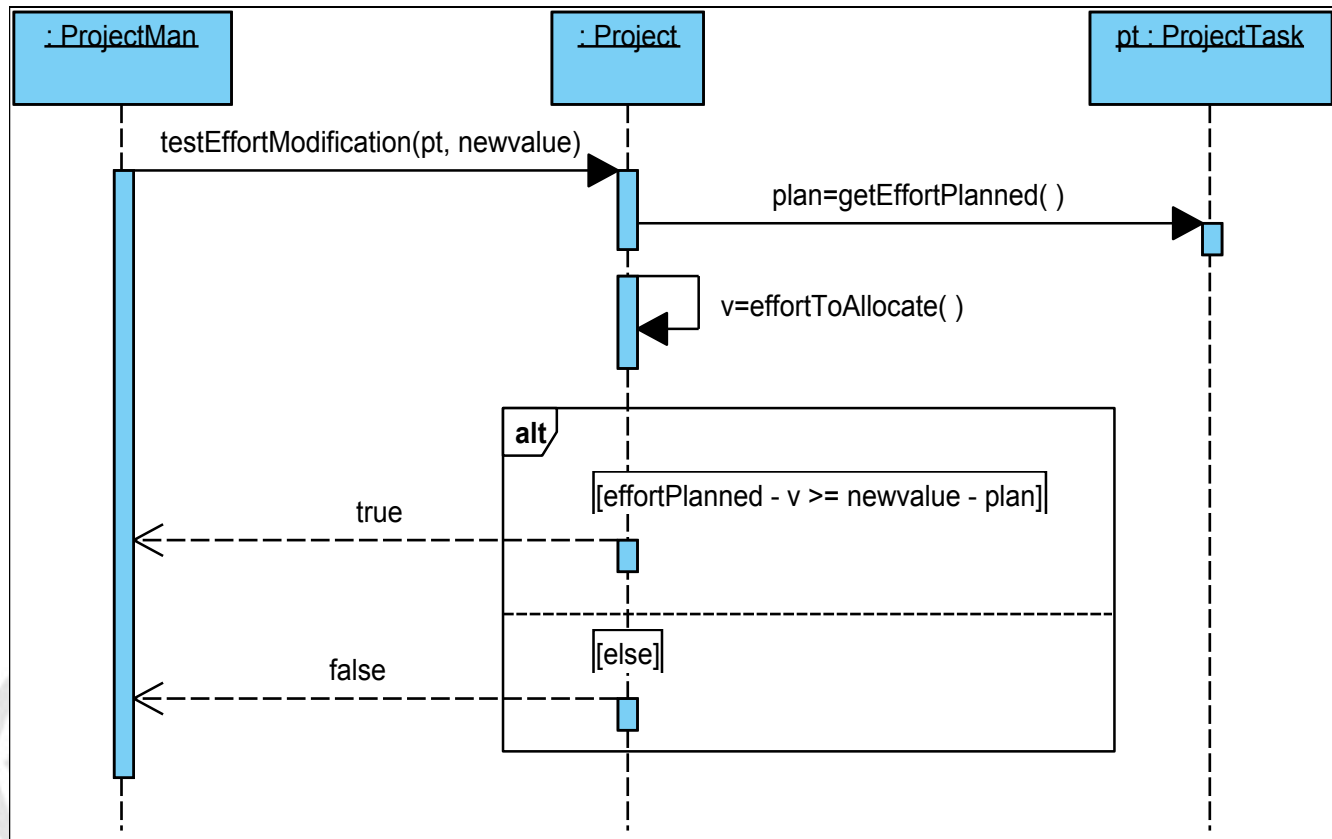


- 由外部推动的项目创建
- 无法准确命名和识别动作的对象，用“extern”对象表示
- 外部extern类使用了角色的图标，这是为了能够清晰的说明此外部类并不属于系统相关的业务类。



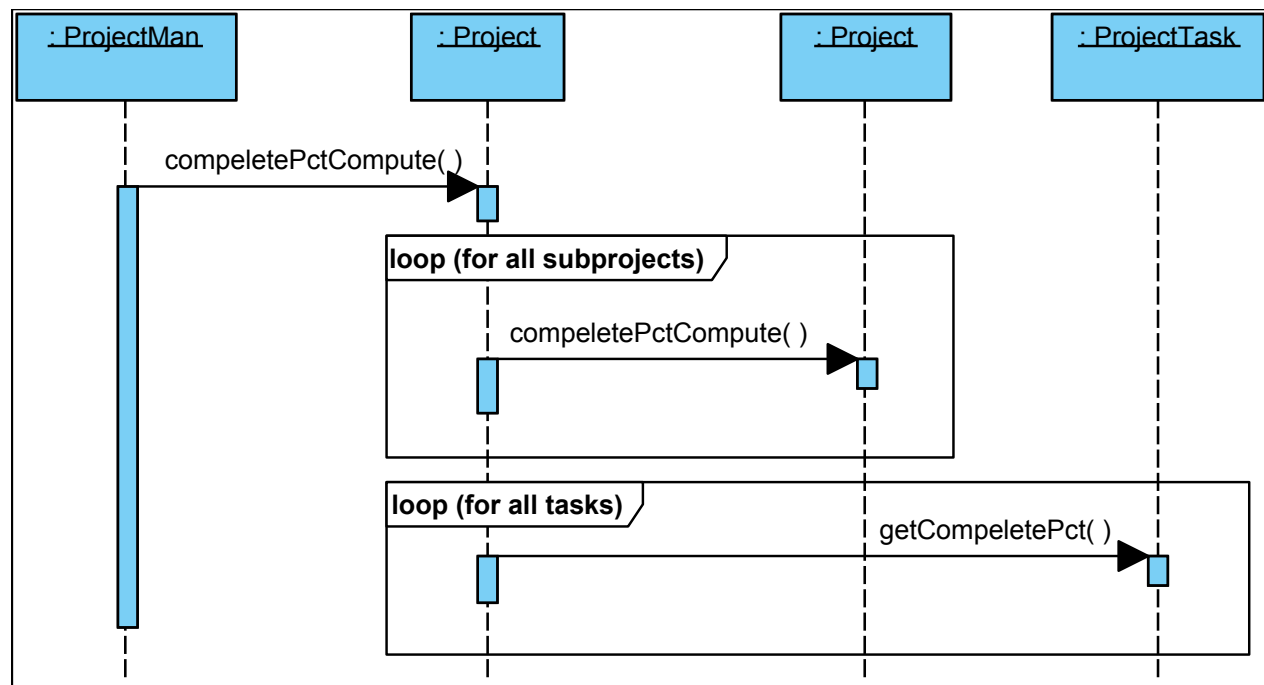
- 项目工作量的修改
- 减少项目的工作量，要对其子项目和任务的工作量进行检查
- 首先总体上进行确认，工作量的新值是否小于原始的计划工作量值

举例



- 任务工作量的修改

举例



完成进度的计算
effortToAllocate()

1. 为计算整个项目的完成进度，需要逐个确定每个子项目的完成进度。
2. 递归的使用在这里没有问题，因为项目只能作为唯一项目的子项目，是一个树结构而不是一个图结构。

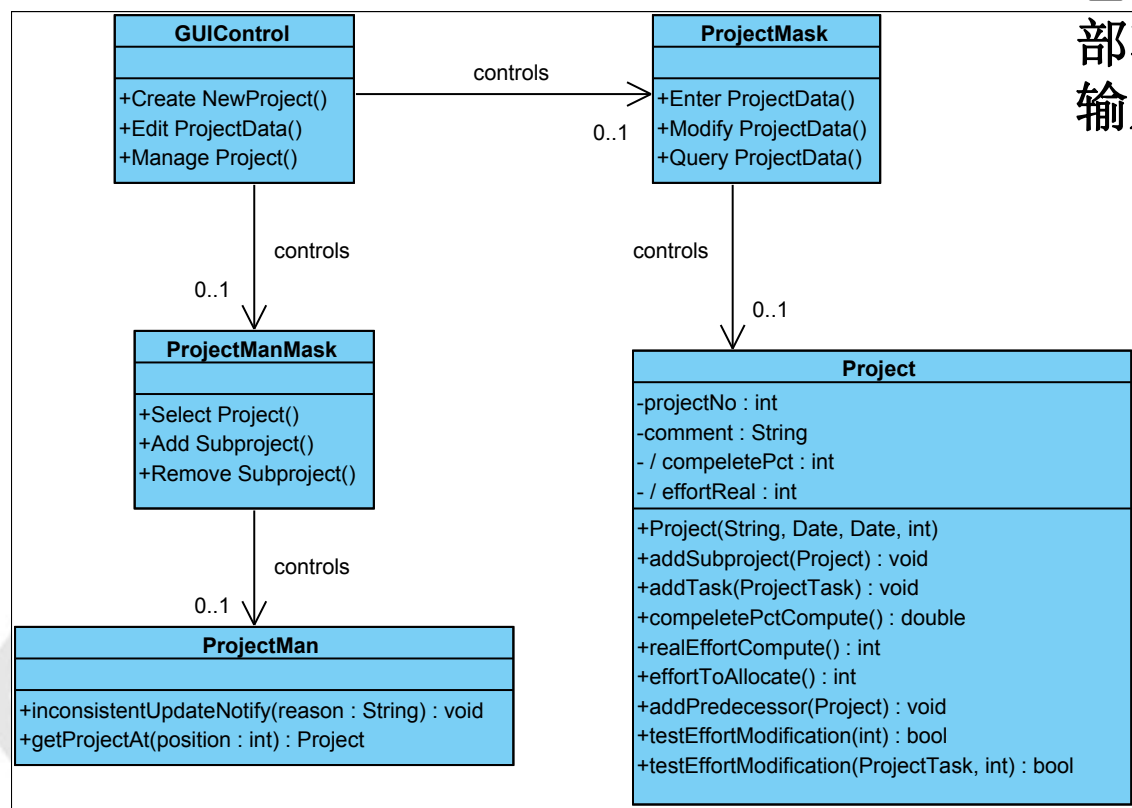
3. 下一步确定每个任务的完成进度以及由这些返回的结果确定项目总体的完成进度。其返回值没有显式的给出。

界面类设计

- 界面设计的基本要求是：通过界面使得模型中含有的类的某些部分对外部可见，比如用户通过界面可进行业务内容的修改或访问，即包括人机交互界面。
- 界面类设计通常可推迟进行，因为可以直接应用现成的类库中的模型，采用不同的类库对整体的类设计会有很大的影响。
- 对现有的类模型补充对应的界面描述，一个直接的方法就是对于每个类补充一个对应的接口，使得它向外部提供可访问的信息。
- 对于项目类可设置一个ProjectMask界面，对外提供项目创建和修改的操作；使用一个界面控制类GUIControl，控制当前哪个类的界面类处于使用状态。



- 用例“项目编辑”中的界面类
- 界面类的方法没有参数，只有一个简要的描述，对应着该对象的职责。
- 也暗示需要的参数主要是在外部功能中进行的初始化，比如输入框。



- 界面布局相关的类没有加入。
- 界面类主要来源为类型2的需求，即针对用户交互的描述。
- 界面类又称边界类，另外一种边界类对应类型3的需求，及为其它系统提供服务的方法。

作业

- 习题1~3

