面向对象的设计

- 使用一组交互的对象和对象类来设计系统

目标

- 为什么软件设计可以表现为一组交互的对象，这些对象管理自己的状态和操作
- 面向对象设计过程中的活动
- 用于描述面向对象设计的各种模型
- 这些模型如何用UML来表示

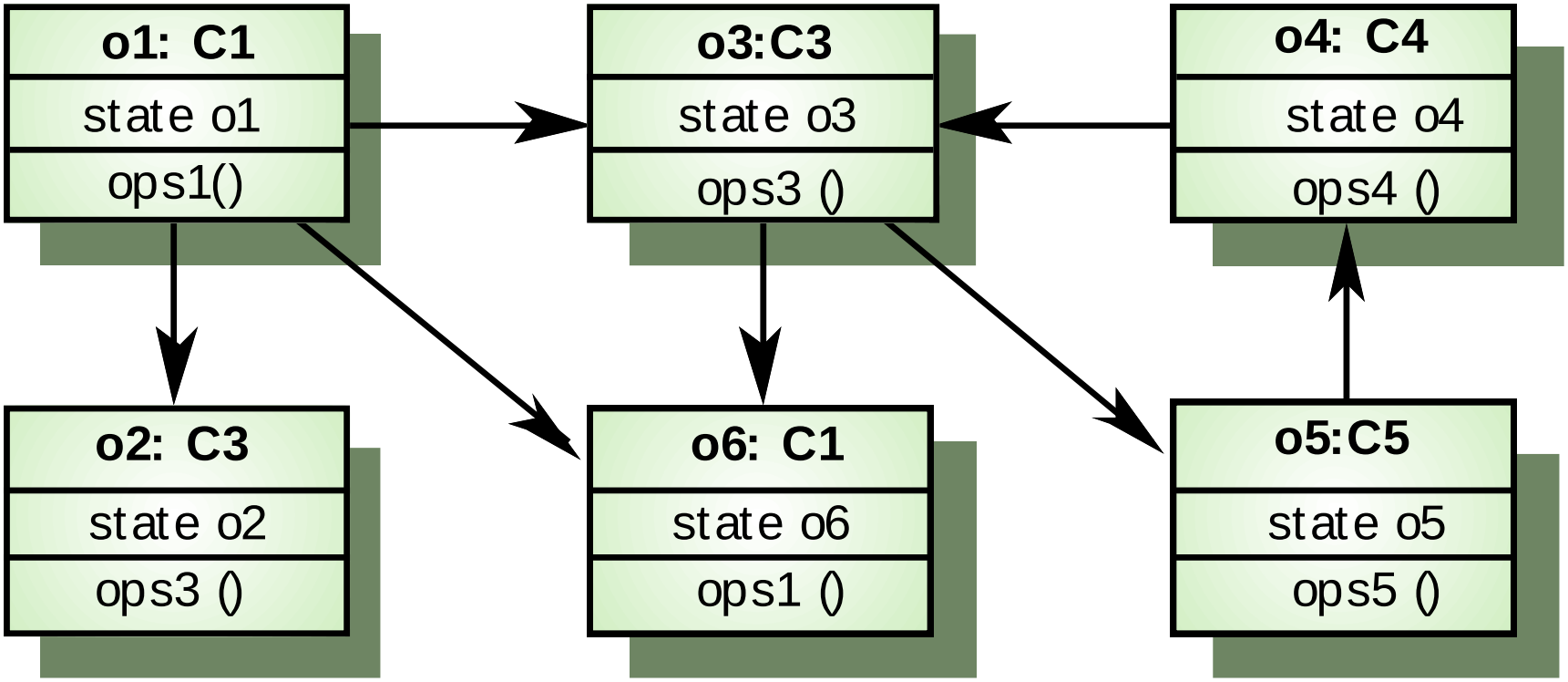
内容

- 对象和对象类
- 面向对象设计过程
- 设计进化

面向对象设计(OOD)的特点

- 对象是真实世界或系统实体的抽象，对象自己管理自己
- 对象是独立的，封装了状态和继承信息
- 系统功能由对象服务来表达
- 没有共享数据区。对象通信通过消息传递
- 对象可以是分布式的，可以顺序执行或者并行执行

对象的交互


o1: C1
state o1
ops1()
o3:C3
state o3
ops3 ()
o4: C4
state o4
ops4 ()
o2: C3
state o2
ops3 ()
o6: C1
state o6
ops1 ()
o5:C5
state o5
ops5 ()

面向对象设计的优点

- 更容易维护. 对象可理解为独立的实体
- 对象是可重用的组件
- 对于某些系统，从真实世界到系统对象有着明显的一一对应

面向对象开发

- 面向对象的分析、设计、编程既是互相联系的，又是互相独立的
- 面向对象分析：建立应用领域的面向对象模型
- 面向对象设计：建立面向对象的系统模型，以实现需求
- 面向对象编程：使用面向对象的编程方法来实现一个面向对象的软件设计。如Java、C++

对象和对象类

- 对象是软件系统中的实体，代表真实世界或系统中的实体
- 对象类是对象的模板，用来生成对象
- 对象类可以从其它对象类继承属性和服务

对象

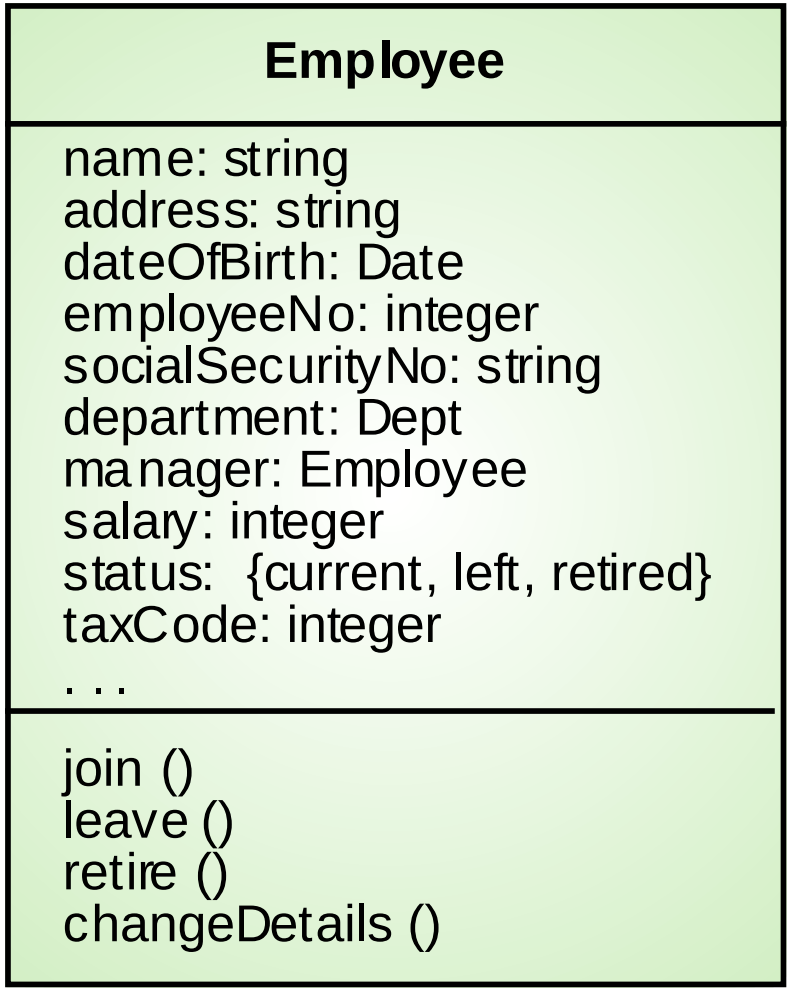
一个对象是由状态和在此状态上的一组操作构成的一个实体。状态由一组对象属性来表示。与对象相关的操作提供给其他对象（客户机）相应的服务，当这些对象在计算过程中需要这些服务的时候，就向该对象请求这些服务。

对象是依照对象类定义创建出来的。对象类定义就是用来创造对象的模板。它包括对所有属性和操作的声明。

统一建模语言

- 1980年代和90年代提出了描述面向对象设计的一些不同的符号
- 统一建模语言是这些符号的集合
- 统一建模语言为多种不同的模型提供了描述符号，这些模型用来生成面向对象的分析和设计
- 统一建模语言已经成为面向对象建模的事实上的标准

雇员对象类(UML)


Employee
name: string
address: string
dateOfBirth: Date
employeeNo: integer
socialSecurityNo: string
department: Dept
manager: Employee
salary: integer
status: {current, left, retired}
taxCode: integer
. . .
join ()
leave ()
retire ()
changeDetails ()

对象通信

- 从概念上说，对象通过消息传递通信
- 消息
  - 调用对象所需的服务的名称
  - 执行服务所需的信息，以及存放服务结果的变量名称
- 从实现上来说, 消息通常通过过程调用来实现
  - 名称 = 过程名.
  - 信息 = 参数列表.

消息的举例

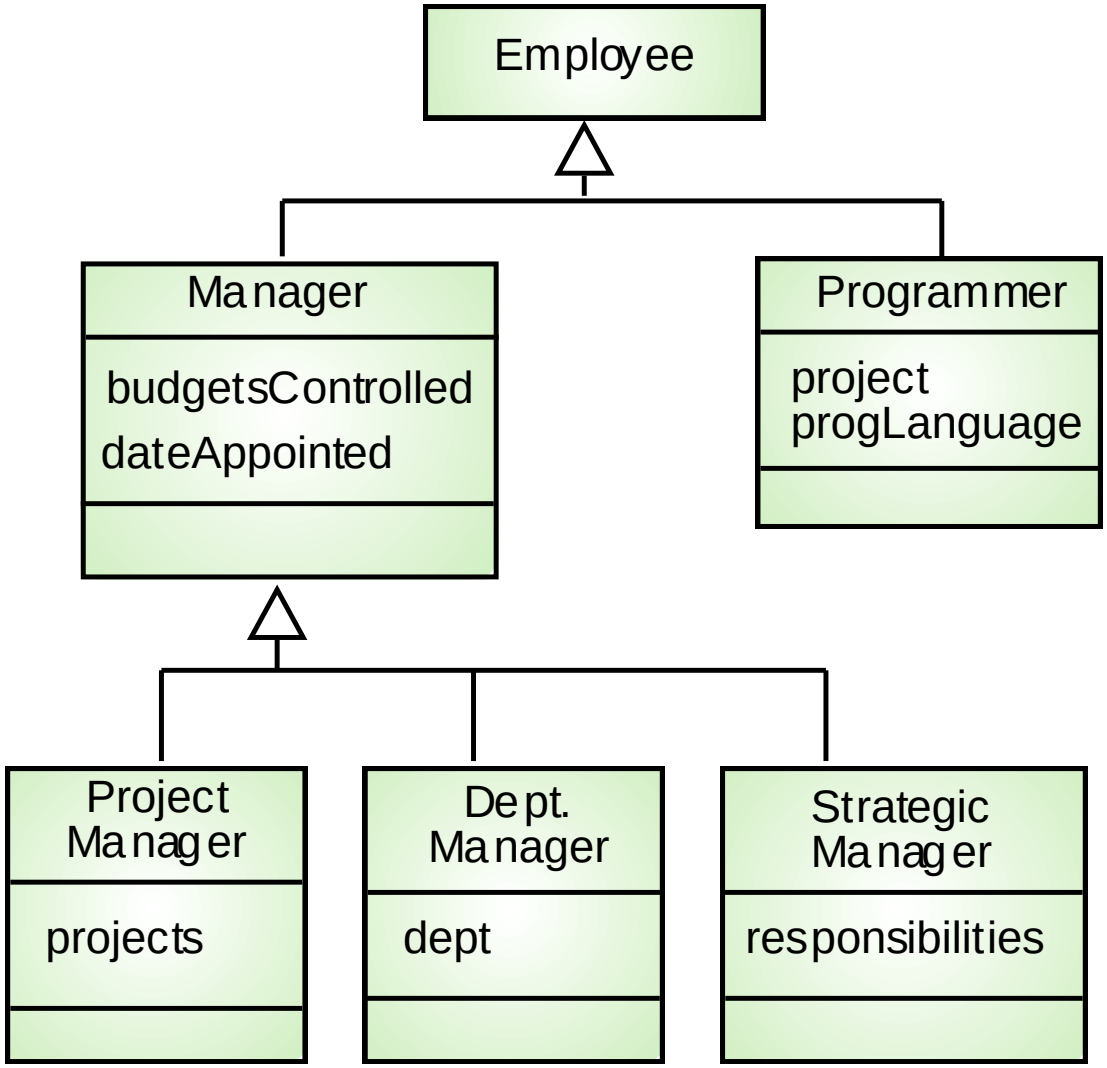
```
// Call a method associated with a buffer
// object that returns the next value
// in the buffer
        v = circularBuffer.Get () ;

// Call the method associated with a
// thermostat object that sets the
// temperature to be maintained
        thermostat.setTemp (20) ;
```

泛化和继承

- 对象类定义了属性和操作
- 类形成一个类层次结构，一个超类（父类）是一个或多个子类的概括
- 一个子类从它的超类（父类）继承属性和操作，也可以增加新的属性和方法
- UML中的泛化，通过面向对象编程中的继承来实现

泛化/继承的层次结构


Employee
Manager
budgetsControlled
dateAppointed
Programmer
project
progLanguage
Project Manager
projects
Dept. Manager
dept
Strategic Manager
responsibilities

继承的优点

- 可用来给实体分类的抽象机制
- 设计和编程层面上的重用机制
- 继承层级图可作为领域和系统的知识源

继承的问题

- 对象类不是独立的。如果不参考其超类，将无法理解对象类
- 设计者会倾向于使用分析阶段生成的继承分层结构图。这会导致效率低下。
- 分析、设计和实现的继承分层结构图具有不同的功能，应该被分别维护

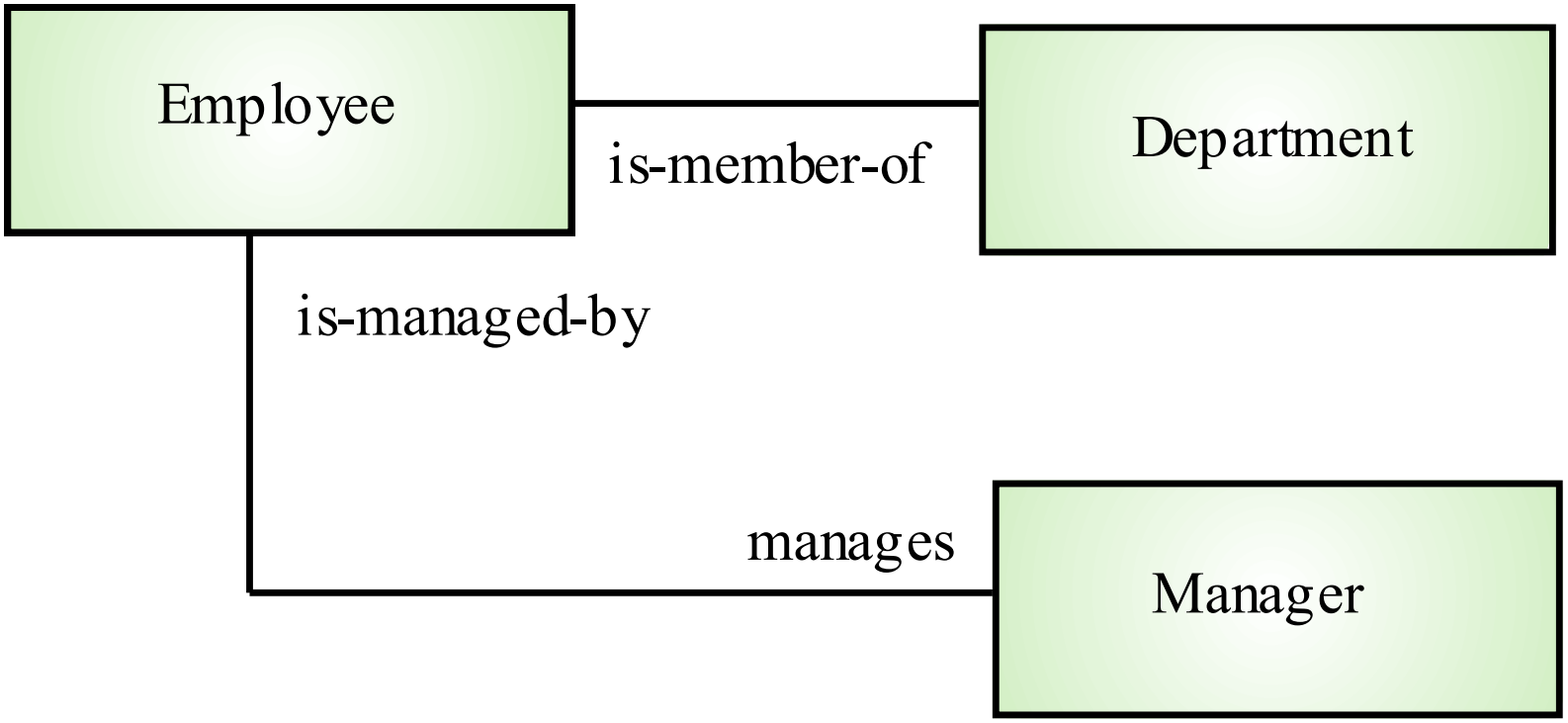
继承与面向对象设计

- 关于继承是否是面向对象设计所必需的，有两个观点：
  - 观点 1. 识别出继承分层结构是面向对象设计的基础。明显地这只能用面向对象编程语言来实现。
  - 观点 2. 继承允许属性和操作的重用，是很有用的。但在设计阶段识别继承分层结构会给实现带来不必要的限制。
- 继承引入了复杂性，这是不希望的，特别对于严格的系统。

UML的关联

- 对象、对象类与其他对象、对象类之间存在关联
- 在UML中, 通过关联来表示，是对象之间的连线
- 可以在连线上附加说明信息
- 关联描述一个对象是另外一个对象的属性或者一个对象方法的实现依赖于相关联的对象。

关联模型


Employee
is-member-of
Department
is-managed-by
manages
Manager

并发对象

- 对象是独立的实体这一本质特征，使得对象适合于并发实现
- 如果对象运行在分布式系统的不同处理机上，对象通信的消息传递模型可以直接实现

服务器和主动对象

- 服务器
  - 对象被实现为一个并行进程（服务器），它的方法对应为定义的对象操作。如果没有对它调用，对象挂起，等待进一步的服务请求
- 主动对象
  - 对象用并行进程实现，对象内部状态的改变由对象的操作完成，而不是直接的外部调用

“询答机”主动对象

- 主动对象通过操作来改变属性，也可以通过内部操作来更新属性
- 飞机上的询答机对象利用卫星导航系统，周期性的更新飞机的位置信息。

“询答机”主动对象

```
class Transponder extends Thread {

        Position currentPosition ;
        Coords c1, c2 ;
        Satellite sat1, sat2 ;
        Navigator theNavigator ;

        public Position givePosition ()
        {
                return currentPosition ;
        }

        public void run ()
        {
                while (true)
                {
                        c1 = sat1.position () ;
                        c2 = sat2.position () ;
                        currentPosition = theNavigator.compute (c1, c2) ;
                }

        }

} //Transponder
```

Java线程

- Java中的线程是实现并发对象的一种简单架构
- 线程中必须包含一个run()方法，这个方法由Java运行时系统启动。
- 典型的主动对象包含一个无限循环，这样它们就可以一直执行下去

面向对象设计过程

- 定义上下文和系统的使用模式
- 设计系统体系结构
- 识别出系统中的主要对象
- 开发设计模型
- 描述对象接口

气象系统描述

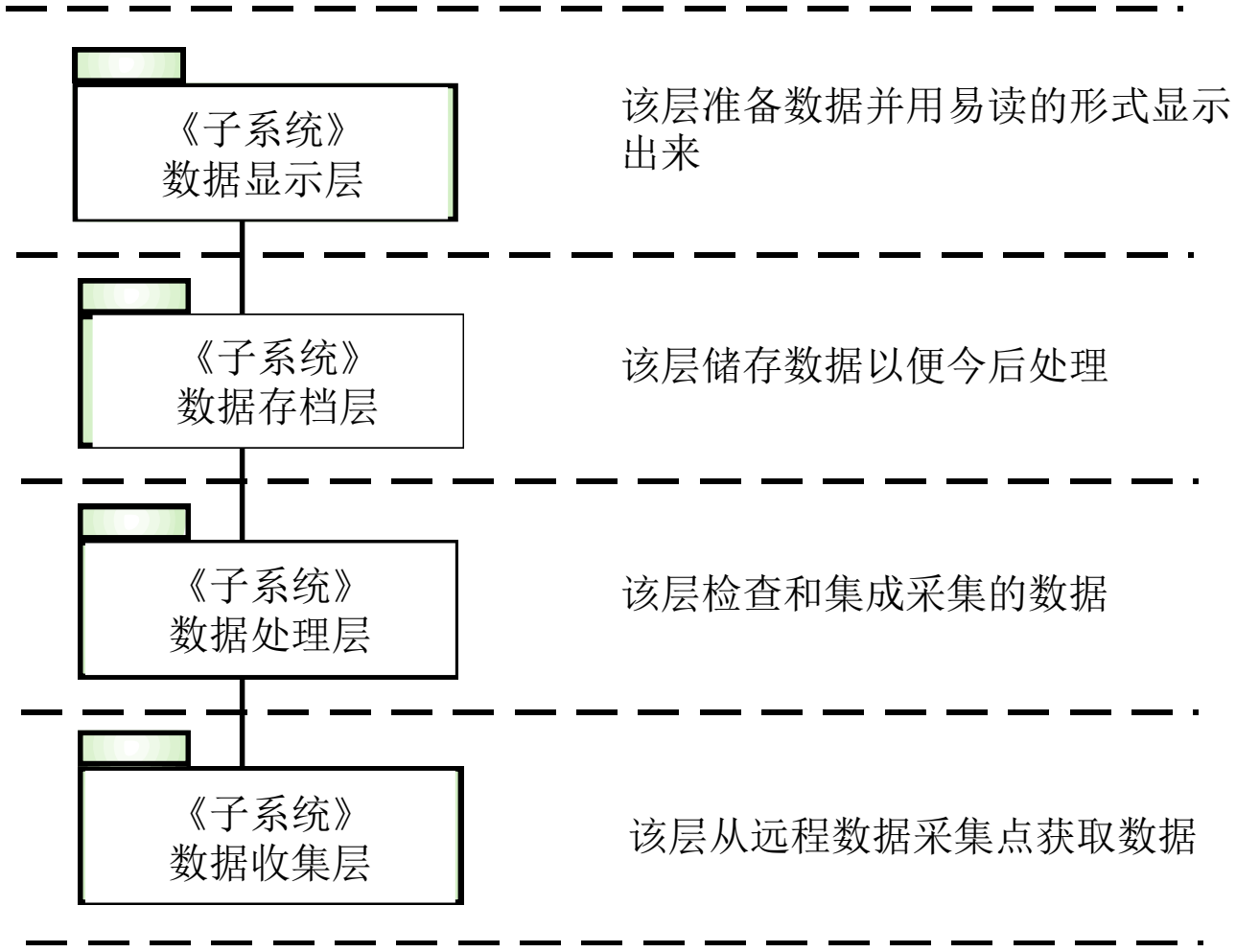
气象制图系统通过收集远方无人看守气象台得到的信息和其他数据源如气象观察者、气球和卫星发来的信息，根据气象学原理来生成气象图。气象台在得到来自区域计算机的请求后发送数据到区域计算机，由区域计算机系统来确认收集到的数据并集成来自不同数据源的数据。将集成数据存档，根据这些集成数据和数字化的地图数据库来生成一组局部气象图。地图可以用专用打印机打印出来以便分发或用不同格式显示出来。

气象站的描述

气象站是一组软件控制的仪器，负责收集数据、数据处理以及发送数据以便做进一步处理。仪器包括空气和地面温度计、风速计、风向标、气压计和雨量测量器。数据每5分钟收集。

当发布了一个传送气象数据命令时，气象站处理和总结收集的数据。如果收到请求，总结的数据传送给绘图计算机。

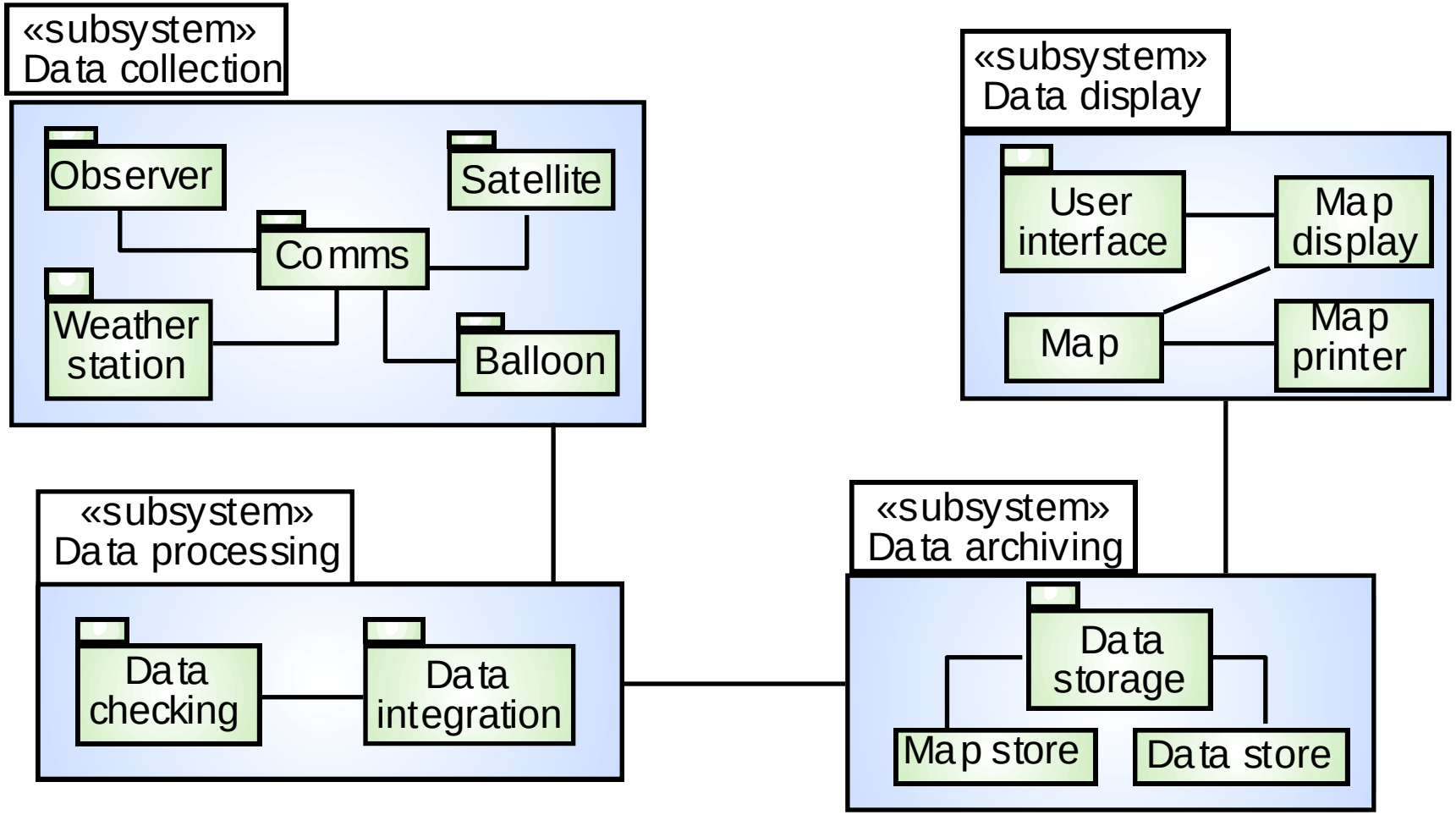
分层体系结构


《子系统》
数据显示层
该层准备数据并用易读的形式显示出来
《子系统》
数据存档层
该层储存数据以便今后处理
《子系统》
数据处理层
该层检查和集成采集的数据
《子系统》
数据收集层
该层从远程数据采集点获取数据

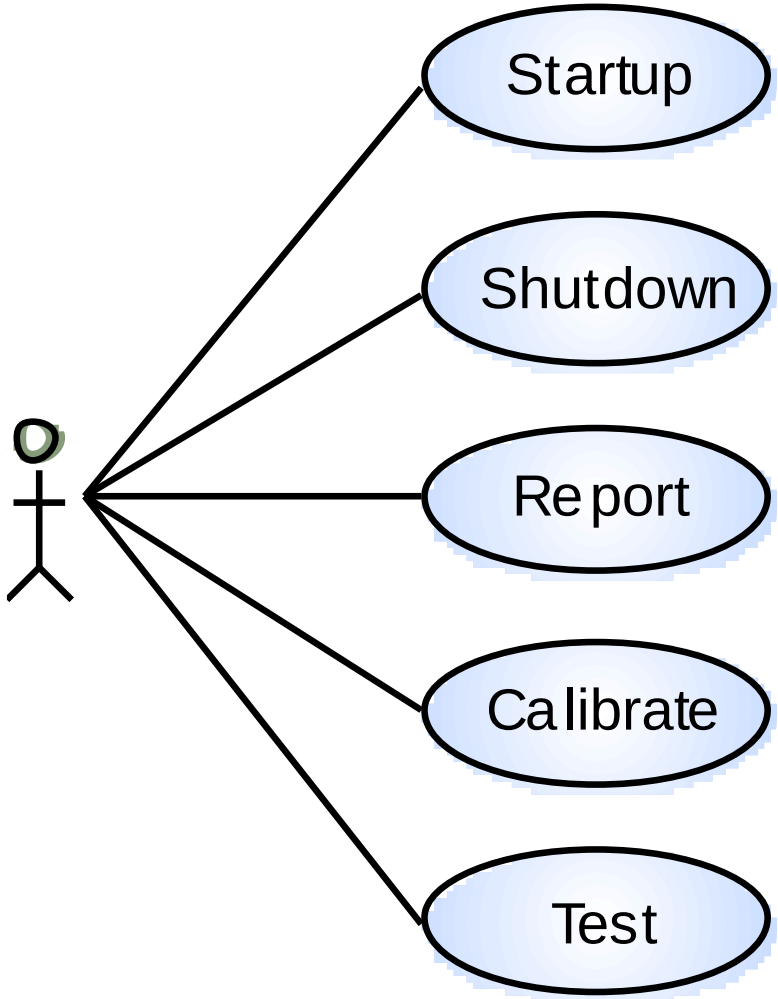
系统上下文和使用模型

- 了解待设计软件和外部环境之间存在的关系
- 系统上下文
  - 是一个静态模型，描述环境中的其他系统。用子系统来表示其他系统。
- 系统使用模型
  - 是一个动态模型，描述系统实际上是如何与环境交互的。使用用例来表示交互。

气象制图系统中的子系统



气象站的用例



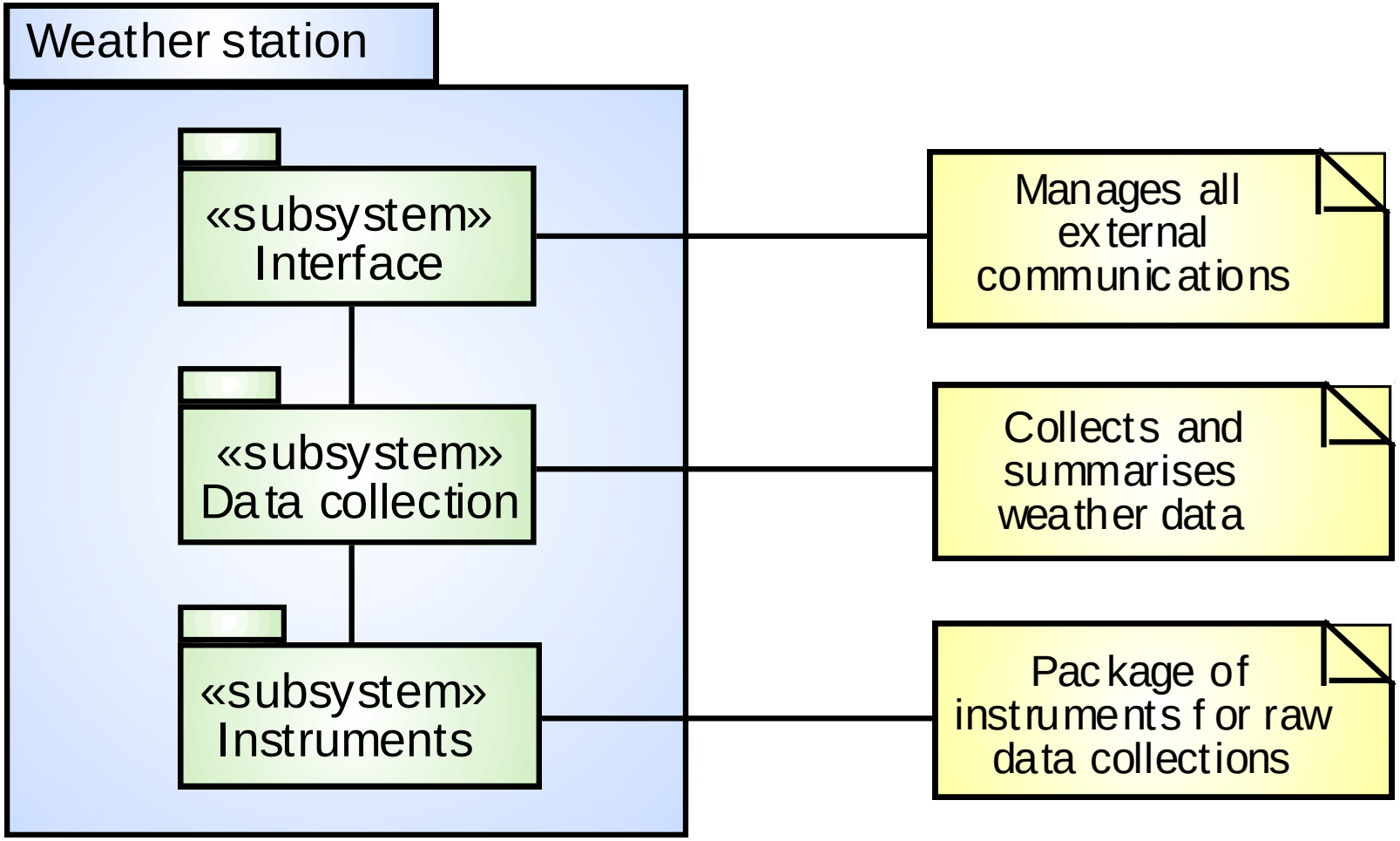
用例描述

System	Weather station
Use-case	Report
Actors	Weather data collection system, Weather station
Data	The weather station sends a summary of the weather data that has been collected from the instruments in the collection period to the weather data collection system. The data sent are the maximum minimum and average ground and air temperatures, the maximum, minimum and average air pressures, the maximum, minimum and average wind speeds, the total rainfall and the wind direction as sampled at 5 minute intervals.
Stimulus	The weather data collection system establishes a modem link with the weather station and requests transmission of the data.
Response	The summarised data is sent to the weather data collection system
Comments	Weather stations are usually asked to report once per hour but this frequency may differ from one station to the other and may be modified in future.

体系结构的设计

- 当理解了系统与其环境的交互后，可使用这个信息设计系统的体系结构
- 对气象站来说分层体系结构是合适的
  - 接口层：处理通信
  - 数据采集层：管理仪器
  - 仪器层：采集数据
- 在一个体系结构模型中不应超过7个实体。

气象站体系结构


Weather station
«subsystem»
Interface
«subsystem»
Data collection
«subsystem»
Instruments
Manages all external communications
Collects and summarises weather data
Package of instruments for raw data collections

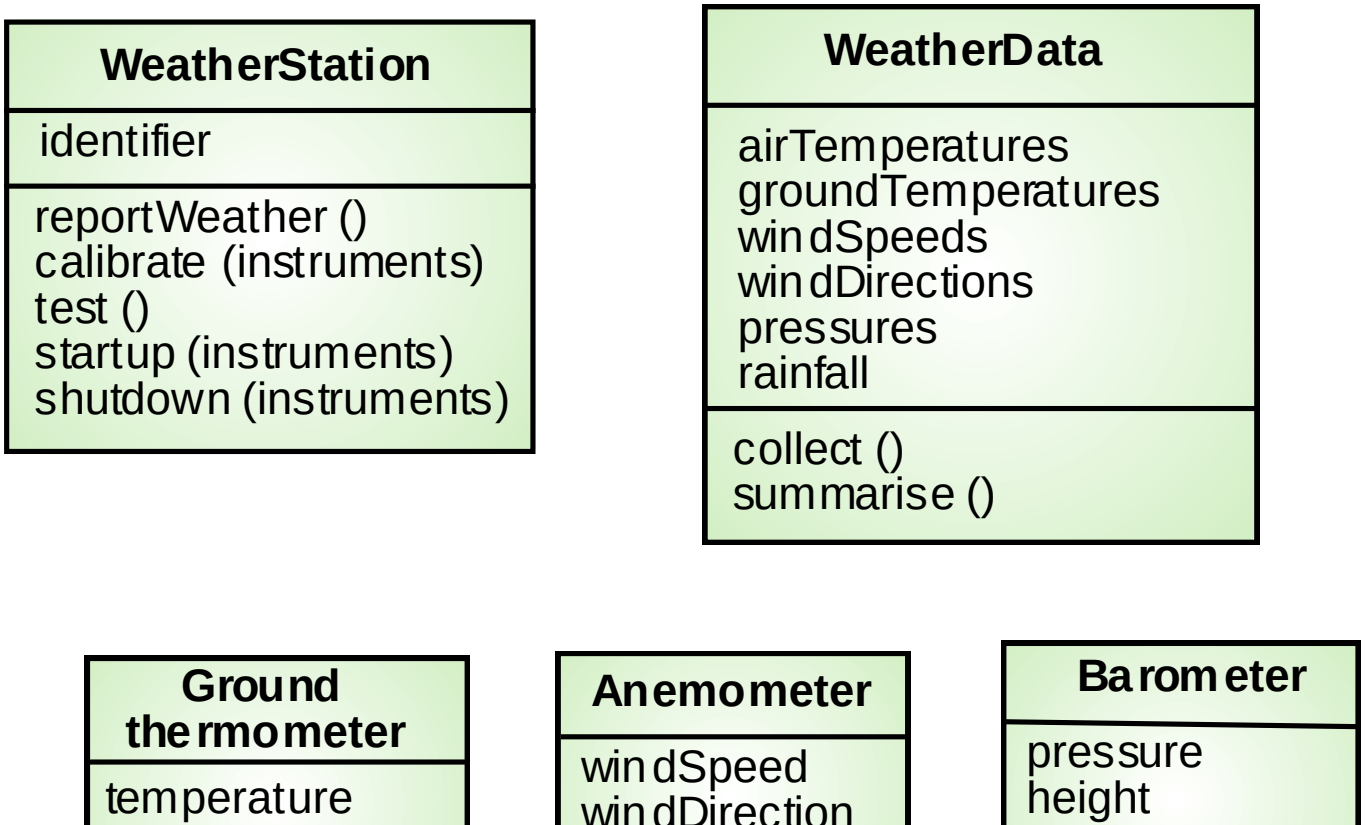
对象识别

- 识别对象（或对象类）是面向对象设计中最困难的部分
- 依赖于系统设计者的技术、经验和领域知识。
- 对象识别是一个反复的过程。第一次不一定能正确。

识别的方法

- 对系统的自然语言描述做文法分析。对象和属性是名词，操作或服务是动词。
- 使用应用领域中的真实实体
- 使用行为方法，了解系统的全部行为，谁参与了什么行为。
- 使用基于情景的分析。在每个情景中识别出对象，属性和方法。

气象站对象类


WeatherStation
identifier
reportWeather ()
calibrate (instruments)
test ()
startup (instruments)
shutdown (instruments)
WeatherData
airTemperatures
groundTemperatures
windSpeeds
windDirections
pressures
rainfall
collect ()
summarise ()
Ground thermometer
temperature
test ()
calibrate ()
Anemometer
windSpeed
windDirection
test ()
Barometer
pressure
height
test ()
calibrate ()

设计模型

- 设计模型说明对象、对象类和对象间的关系
- 静态模型：用对象类和关联来描述系统静态结构
- 动态模型：描述对象间动态交互

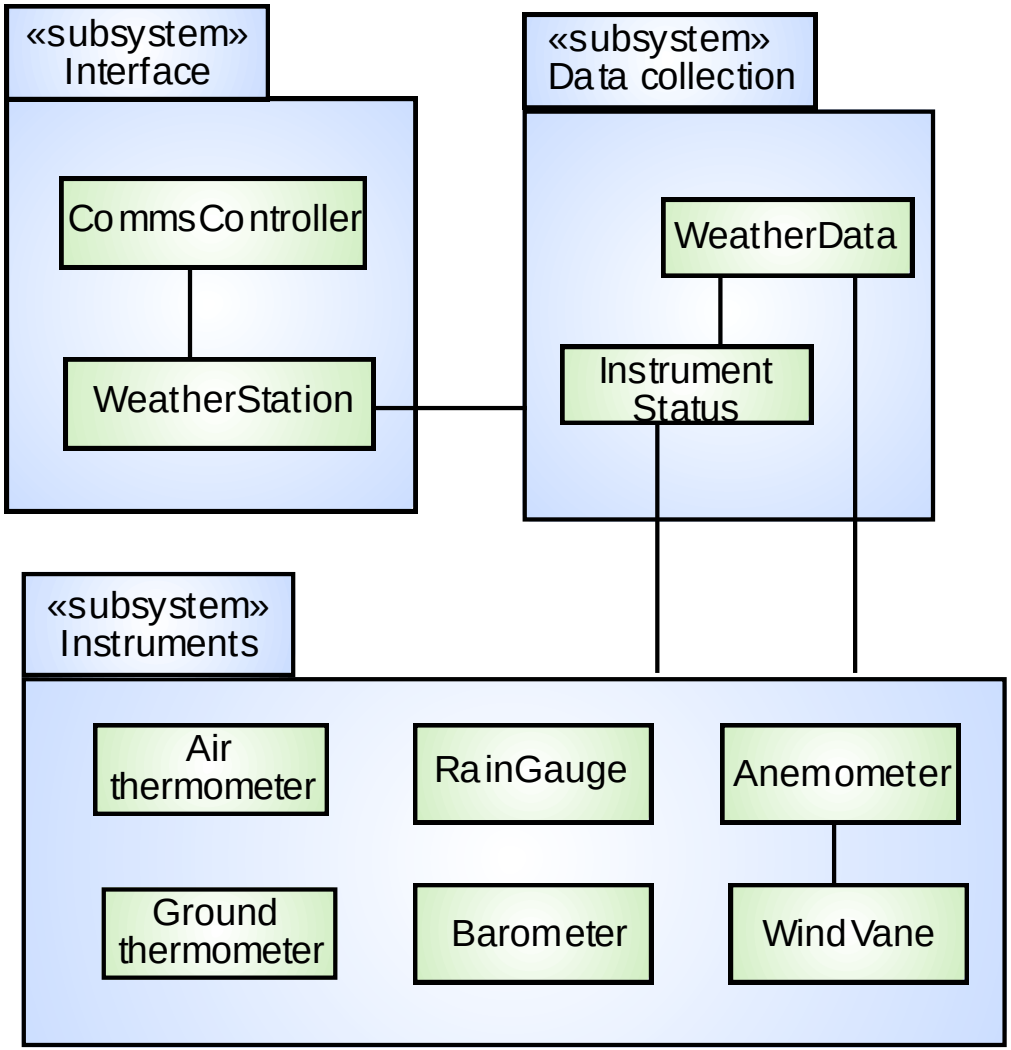
设计模型的例子

- 子系统模型：说明对象的逻辑分组，每个分组构成一个子系统
- 序列模型：说明对象交互的序列
- 状态机模型：说明单个对象如何响应事件来改变它们的状态
- 其他模型：用例模型、聚合模型、泛化模型等

子系统模型

- 说明设计是如何由逻辑上关联的一组对象构成的。
- 在UML中, 这些用“包”来说明。包是一种封装构造，这是一个逻辑模型，系统中实际的对象组织可以是不同的。

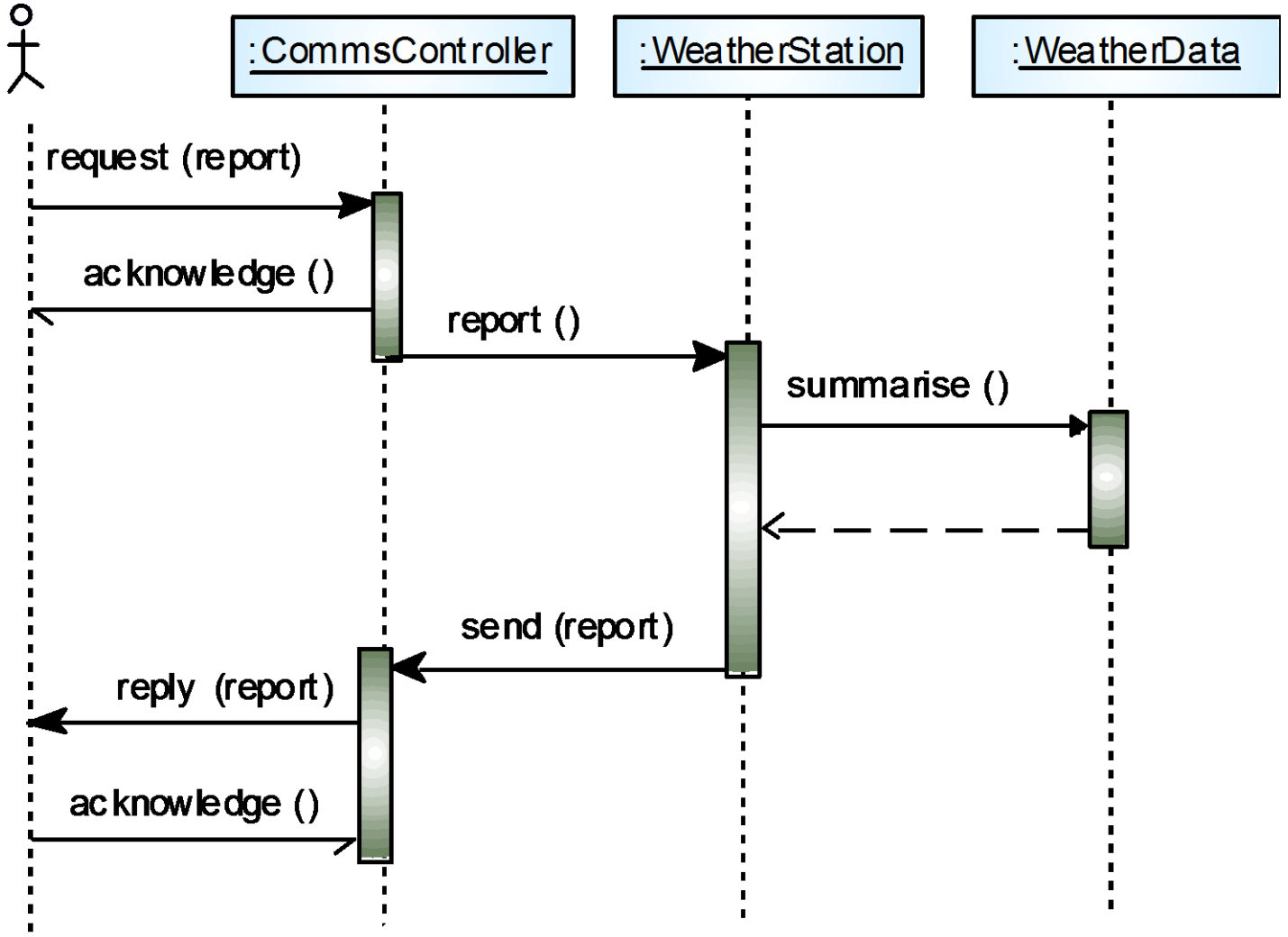
气象站子系统


«subsystem»
Interface
CommsController
WeatherStation
«subsystem»
Data collection
WeatherData
Instrument
Status
«subsystem»
Instruments
Air
thermometer
RainGauge
Anemometer
Ground
thermometer
Barometer
WindVane

序列模型

- 序列模型说明所发生的对象交互的序列
  - 参与交互的对象水平的排列，每个对象有一条垂直的线条与之连接。
  - 时间以垂直方向表示，时间的进展是沿着垂直的虚线向下
  - 对象之间的交互表示为带有标号的箭头，该箭头是与垂直线段相连的。这些不是数据流，而是表示交互中的基本消息或事件。
  - 对象生命线上的细长方形表示这个对象是系统中控制对象的时间。一个对象在这个长方形的顶端时刻接管控制，在底部放弃控制。

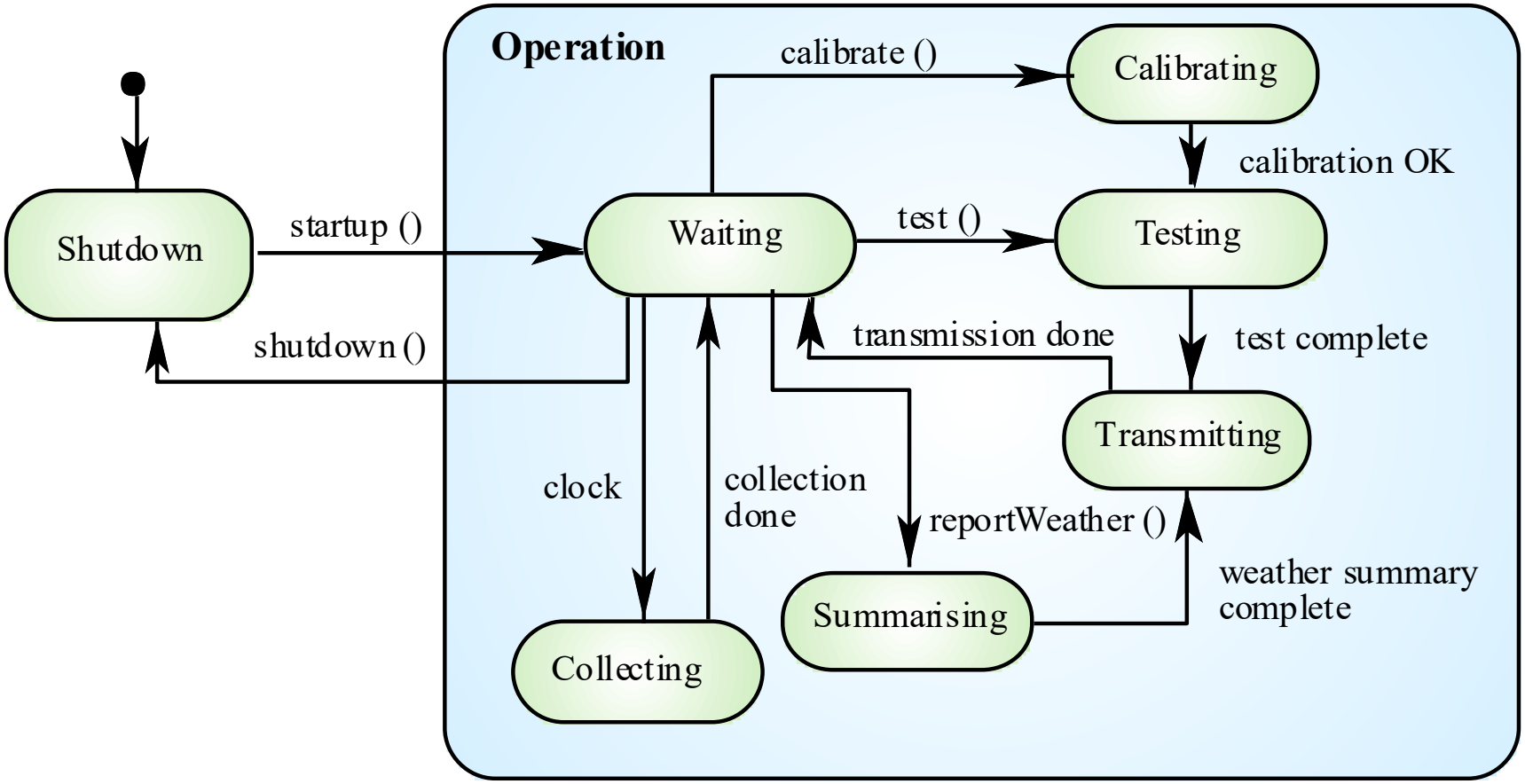
数据收集序列


:CommsController
:WeatherStation
:WeatherData
request (report)
acknowledge ()
report ()
summarise ()
send (report)
reply (report)
acknowledge ()

状态图

- 说明对象是如何对不同的服务请求作出响应，以及这些服务请求出发的状态迁移
  - 如果对象状态是Shutdown，则它只能响应 Startup()
  - 在等待状态中，对象等待进一步的消息。
  - 如果接收到一个reportWeather ()消息，系统就转移到整理数据状态
  - 如果收到一个calibrate ()消息，系统转移到一个校正状态
  - 如果收到一个时钟信号，系统就转移到采集状态

气象台的状态图


Operation
calibrate ()
Calibrating
calibration OK
Shutdown
startup ()
Waiting
test ()
Testing
shutdown ()
transmission done
test complete
Transmitting
clock
collection
done
reportWeather ()
weather summary
complete
Summarising
Collecting

对象接口描述

- 必须描述对象接口，以便对象和其他组件能够并行设计
- 设计者应避免设计接口的具体表示，应将其隐藏在对象自身中
- 同一个对象可有多个接口，从不同角度观察的方法可得到不同的接口
- UML用类图来描述接口，也使用Java

气象站接口

```
interface WeatherStation {

        public void WeatherStation () ;

        public void startup () ;
        public void startup (Instrument i) ;

        public void shutdown () ;
        public void shutdown (Instrument i) ;

        public void reportWeather ( ) ;

        public void test () ;
        public void test ( Instrument i ) ;

        public void calibrate ( Instrument i) ;

        public int getID () ;

} /WeatherStation
```

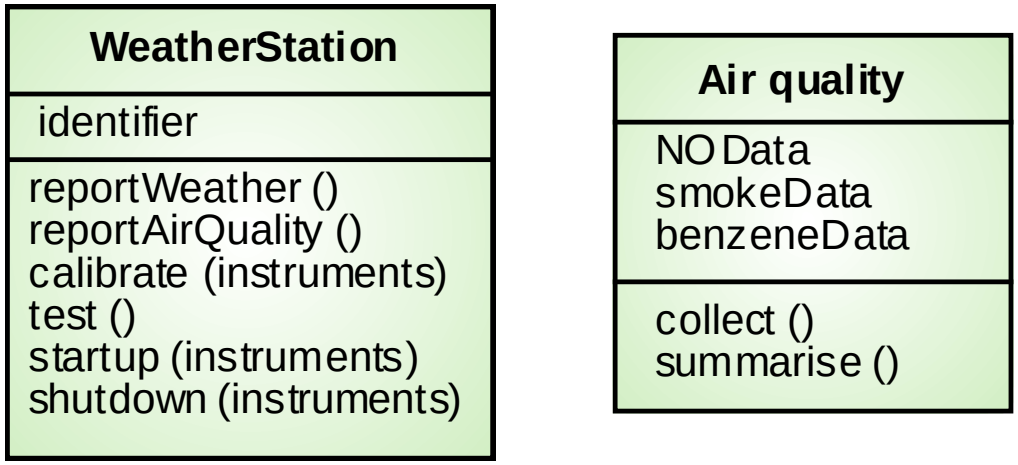
设计进化

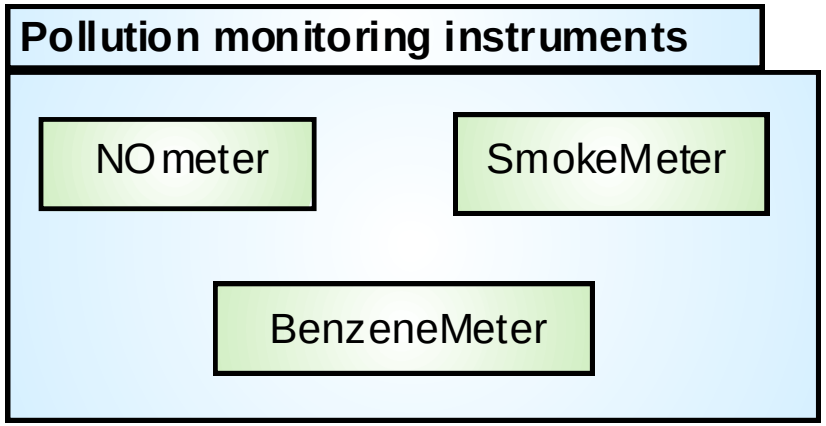
- 隐藏信息到对象中意味着改变对象不会以一种不可预知的方式影响其他对象
- 假设加入污染监控仪到气象站中。监测空气，算出空气中的不同污染物的量。
- 污染数据采集不会对气候数据采集产生任何影响

需要的变化

- 增加一个对象类 ‘Air quality’作为WeatherStation的一部分
- 增加一个操作reportAirQuality到类WeatherStation中。变更控制软件以采集污染数据。
- 增加表示污染监测仪器的对象

污染监测


WeatherStation
identifier
reportWeather ()
reportAirQuality ()
calibrate (instruments)
test ()
startup (instruments)
shutdown (instruments)
Air quality
NO Data
smokeData
benzeneData
collect ()
summarise ()


Pollution monitoring instruments
NO meter
SmokeMeter
BenzeneMeter

要点

- 面向对象设计是设计软件的一个重要手段。对象具有自己私有的状态和操作
- 对象应该有构造和检查的操作。提供服务给其他对象。
- 对象可以是顺序或并发地执行。
- 统一建模语言（UML）为定义不同的对象模型提供了不同的符号

要点

- 在面向对象设计过程中会产生很多不同的模型，包括静态模型和动态模型
- 对象接口应该使用诸如Java的编程语言来精确定义。
- 面向对象设计简化了系统进化