オブジェクト指向設計
Object-oriented Design

参考書: Ian Sommerville, “Software Engineering”, 8th edition, Pearson Education

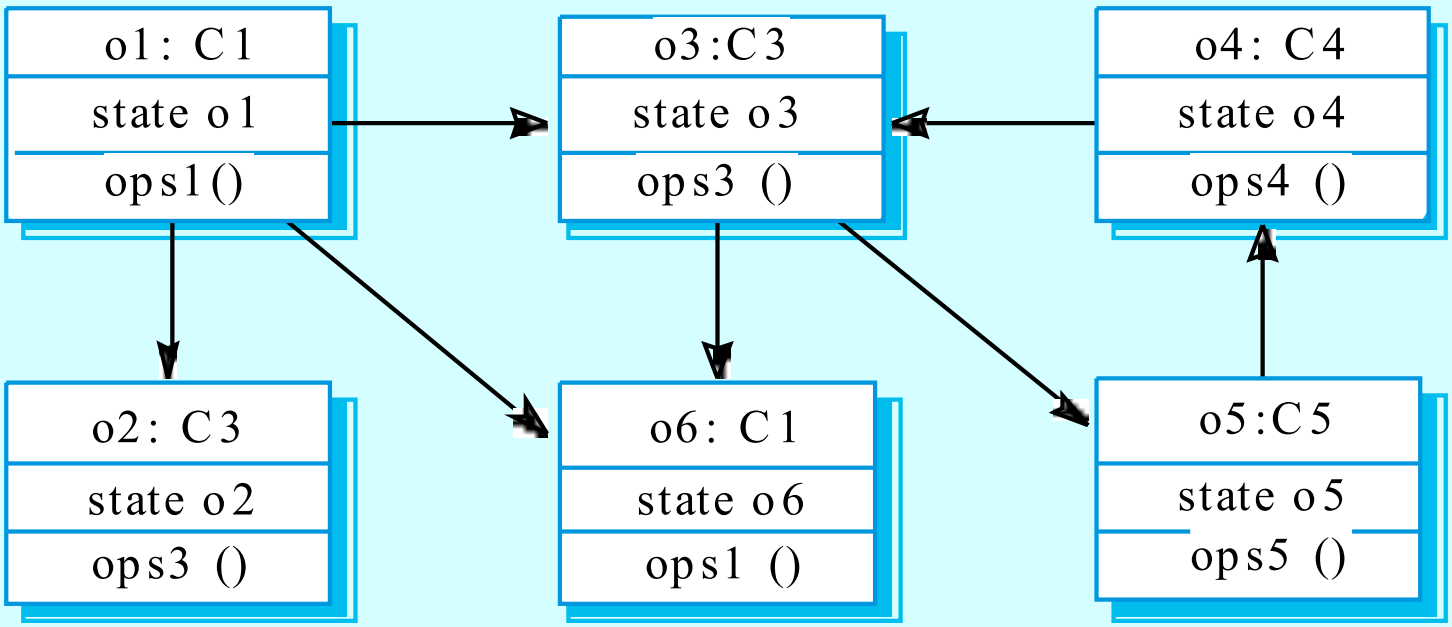
オブジェクト指向開発

- オブジェクト指向分析（OOA），オブジェクト指向設計（OOD），オブジェクト指向プログラミング（OOP）は関係してはいるものの別のプロセス
- OOAでは応用領域のモデルをオブジェクトモデルで構築
- OOD は要求を実装するためのシステムモデルをオブジェクトモデルで構築
- OOPはJavaやC++などのオブジェクト指向プログラミング言語でOODの結果を実装

OODの特徴

- 自身を管理する「オブジェクト」によって実世界やシステム上の対象を抽象的に表現
- オブジェクトは独立性を持ち，自身の状態と表現情報をカプセル化
- システムの機能をオブジェクトサービスとして表現
- 共有データ領域は無い，もしくは限られ，オブジェクト間の通信はメッセージ伝達による
- オブジェクトは分散配置でき，逐次的に実行されても並行に実行されても良い

やりとりするオブジェクト



OODの利点

- 保守が容易．オブジェクトを自立した対象と捉えることができる.
- オブジェクトは再利用の単位（コンポーネント）となりうる.
- ある種のシステムでは，現実世界の対象とオブジェクトとの対応が明らかで，モデル化が容易でモデルの理解も容易となる.

オブジェクトとオブジェクトクラス

- オブジェクト：実世界あるいはシステム内の実体（インスタンス）をソフトウェアシステム上で表現するもの.
- オブジェクトクラスはオブジェクトのテンプレートであって、オブジェクトを生成する時に利用される.
- オブジェクトクラスが他のクラスの属性やサービスを継承することもある.

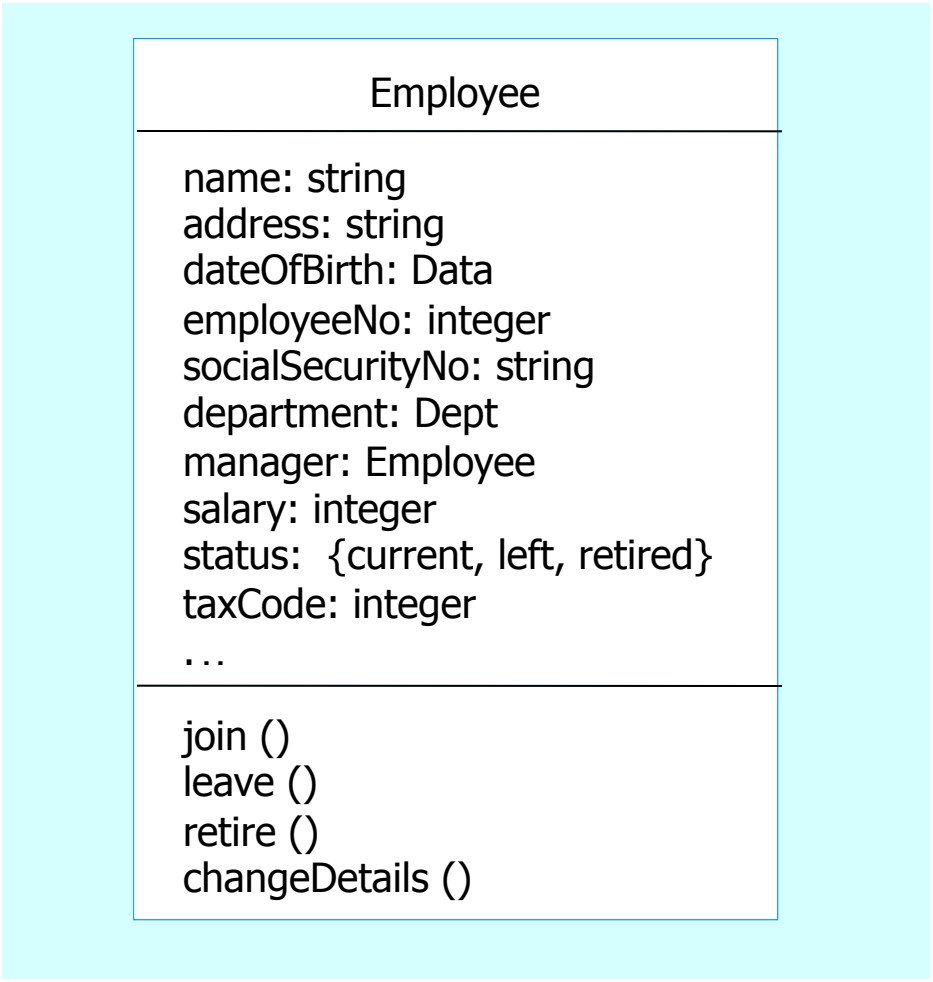
オブジェクトとオブジェクトクラス

- オブジェクトは一つの状態と状態に対するオペレーション群を持つ.
  - 状態はオブジェクト属性の集合として表現される.
  - オペレーションは，何らかの計算を必要とする他のオブジェクト（クライアント）からの要求に応じてサービスを提供する.
- オブジェクトは，オブジェクトクラスの定義にしたがって生成される．オブジェクトクラスの定義はオブジェクトのテンプレートとなる．そのクラスのオブジェクトが持つすべての属性とサービスの宣言が行われる.

UML: Unified Modeling Language

- 1980-90年代にかけて色々なオブジェクト指向分析設計のためのモデル記法が提案.
- Unified Modeling Language (UML) はそれらを統合したもの.
- OOAおよびOODで使われるいくつかのモデルのための記法を規定したもの.
- オブジェクト指向モデリングにとって，既成事実化された標準であるといえる.

オブジェクトクラスEmployee (UML)



オブジェクト間通信

- 概念的には，オブジェクト間のやりとりはメッセージ通信で行われる.
- メッセージ
  - 呼び出し側のオブジェクトが要求するサービスの名前
  - サービスを実行するのに必要な情報のコピーとサービスの結果を保持する入れもの
- 実際にはメッセージを手続き呼び出しで実現
  - サービスの名前 = 手続き名
  - 必要な情報 = 引数リスト

メッセージの例

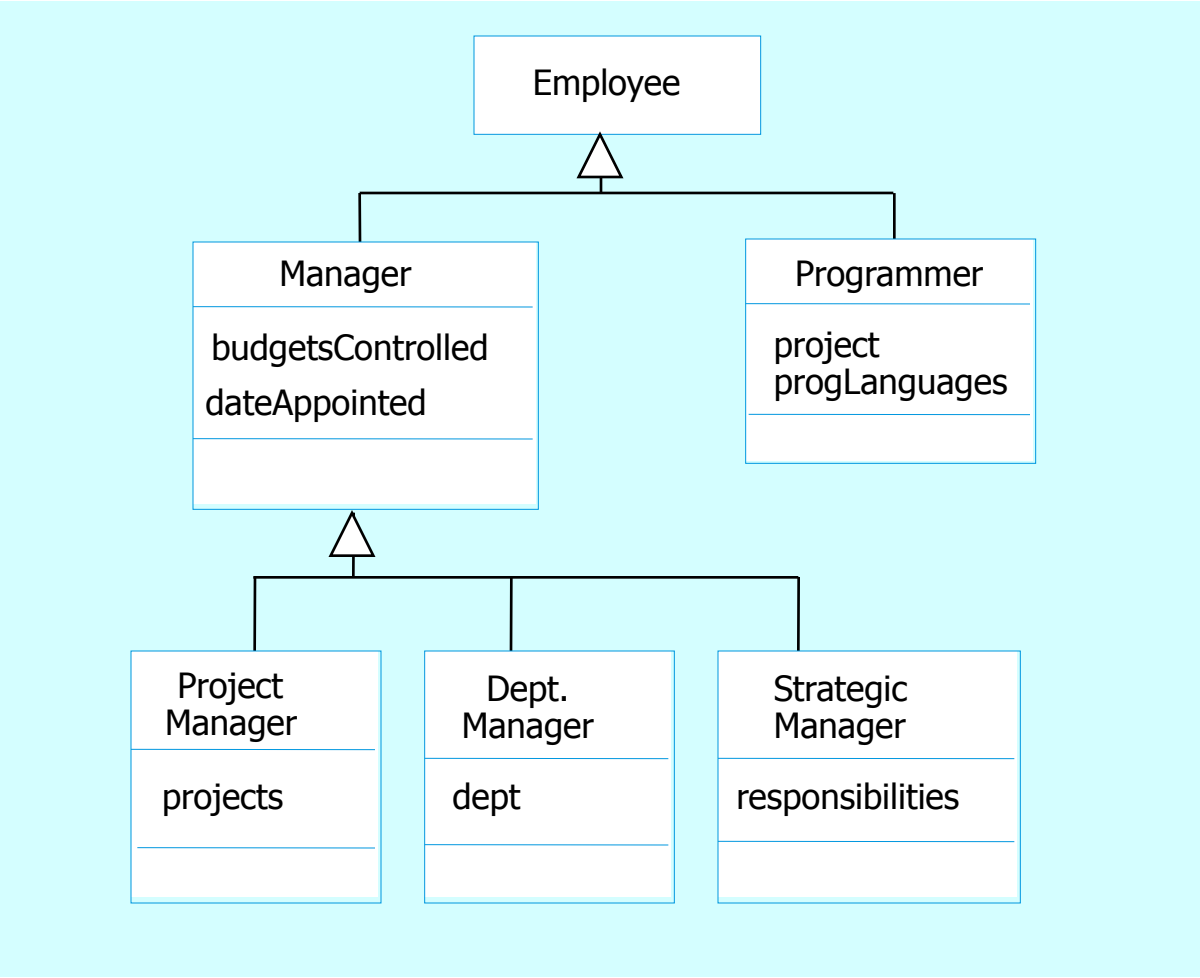
```
// バッファオブジェクトに関連づけられた
// メソッドの呼び出し。
// バッファ内の次の値を返す。
    v = circularBuffer.Get () ;

// サーモスタットオブジェクトに関連づけられた
// メソッドの呼び出し。
// 保持するべき温度を設定する。
    thermostat.setTemp (20) ;
```

汎化関係と継承

- オブジェクトは属性の型とオペレーションを定義したクラスのメンバ.
- クラスは，あるクラス（スーパークラス）が他の複数のクラス（サブクラス）を汎化したものであるという関係の階層状に整理できる.
- サブクラスは，スーパークラスから属性とオペレーションを継承し，それに独自の属性やオペレーションを付け加えることができる.
- UMLでの汎化関係はOOプログラミング言語の継承として実現される.

汎化階層


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

継承の利点

- 抽象化のメカニズムであり，対象を分類する目的で用いることができる.
- 設計およびプログラミングの両レベルでの再利用のメカニズムを提供する.
- 継承グラフは応用領域やシステムに対する組織的な知識体系に相当する.

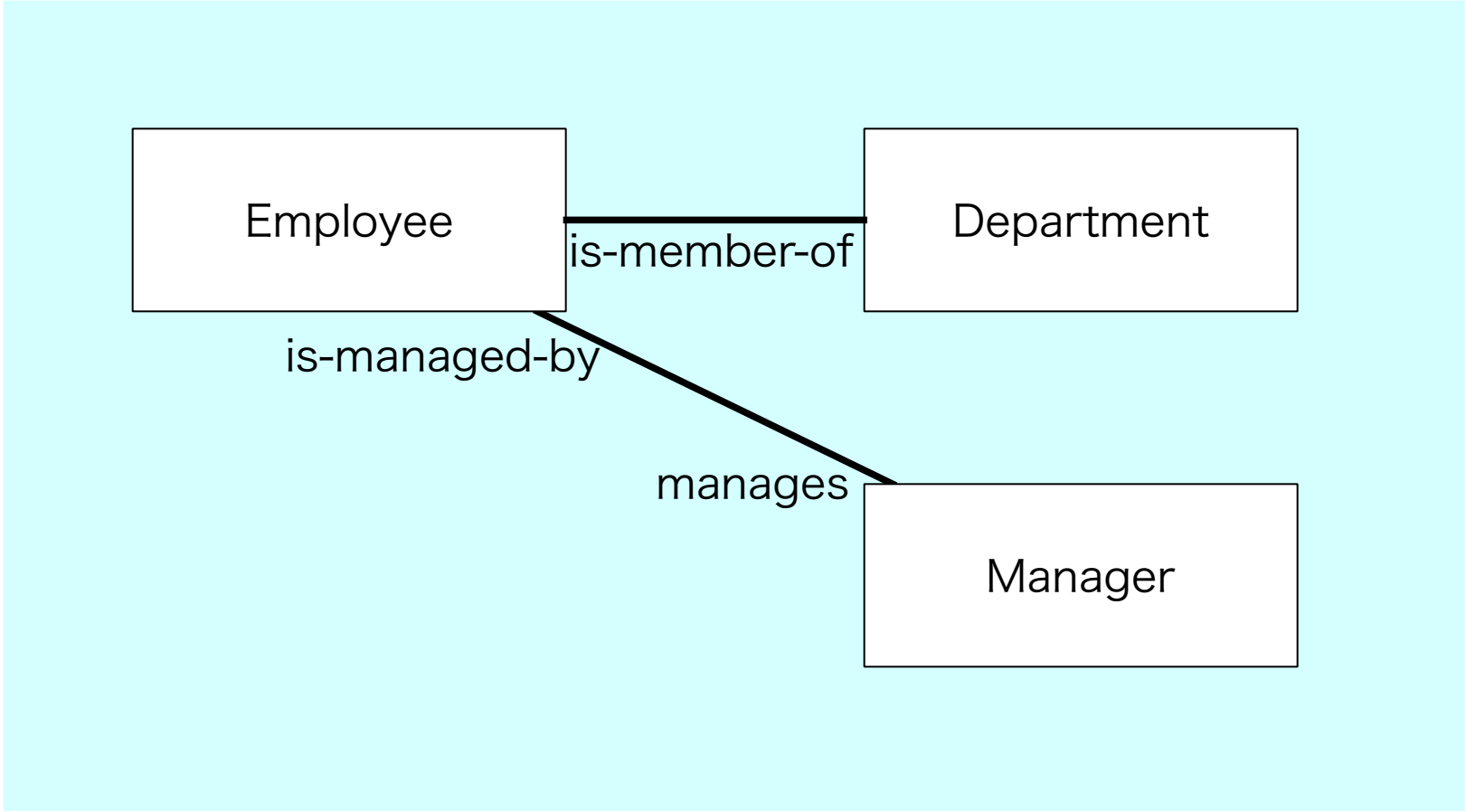
継承の問題点

- オブジェクトクラスが自己完結しなくなる．そのクラスを理解するためにはスーパークラスの理解が不可欠になる．
- 要求分析の時に得られた継承グラフを設計のときにそのまま利用しがちだが，性能劣化の原因となる場合がある．
- 分析，設計，実装結果としての継承グラフの目的／役割はそれぞれ異なっており，別々に保守すべきものである．

UMLアソシエーション（関連）

- オブジェクトおよびクラスと，他のオブジェクトおよびクラスとの間の関連
- UMLでは，一般的な関連はアソシエーションとして表現される
- アソシエーションにはそのアソシエーションを表現する情報を付与できる
- アソシエーションは一般的な関連／結び付き．あるオブジェクトの属性が関連先のオブジェクトに依存したり，あるオブジェクトのメソッドが関連先のオブジェクトに依存したりする場合に用いる

アソシエーションモデル



並行オブジェクト

- オブジェクトは本来自立的な対象であり，並行実装に向く.
- オブジェクトが分散システム内の異なるプロセッサ上で動作している場合，メッセージ通信モデルを素直に実装できる.

オブジェクト指向設計プロセス

- 構造化された開発プロセスでは多種類のモデルが数多く作られる
- それらを作成し，保守するには多くの労力が必要．したがって小さなシステムでは対費用効果が小さい
- 大規模なシステムを多くのグループにより開発する場合，モデルは重要な情報交換の手段となる．

プロセスの詳細

- 一般的に次のような段階で作業が行われる.
  - システムのコンテキストと利用モデルを定義する（コンテキストモデルの構築）
  - システムアーキテクチャの決定（アーキテクチャモデルの構築）
  - 基本的なシステムオブジェクトの同定
  - 設計モデルの構築
  - オブジェクトインタフェースの定義

お天気システムの記述

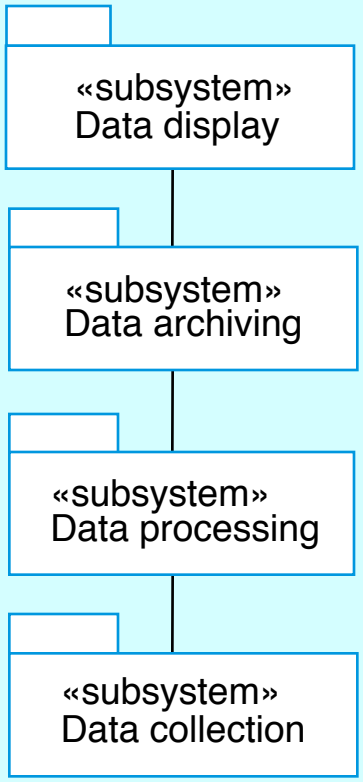
天気マップシステムは，測候基地で天気マップを作成するために用いられる．マップは遠隔の無人天気ステーション，および，天気観測者，気球，衛星などその他のデータソースから集められたデータを用いて作成される．天気ステーションはエリアコンピュータからの要求に応じてデータを送付する．

エリアコンピュータシステムは，集めたデータの検証を行い，他のデータソースから集められたデータと統合する．統合されたデータは蓄積される．蓄積されたデータとデジタル地図データベースを用いて一連の天気マップが生成される．作成された天気マップは専用のプリンタで配布用に出力したり，様々なフォーマットでディスプレイすることができる．

システムのコンテキストと利用のモデル

- 設計しようとするソフトウェアと外部環境との関連を理解する
- システムコンテキスト
  - 環境内の他のシステムを記述する静的なモデル．サブシステムモデルを利用して他のシステムを表現.
  - 次のスライドに天気ステーションシステムの回りにある他のシステムを示す.
- システム利用のモデル
  - システムが外部環境とどのようにやりとりするかを表現する動的なモデル．ユースケースを利用してやりとりを記述.

レイヤードアーキテクチャ



Data display layer

人間が読める形式でデータを準備し、提供するオブジェクト群

Data archiving layer

今後の処理のためにデータを格納するオブジェクト群

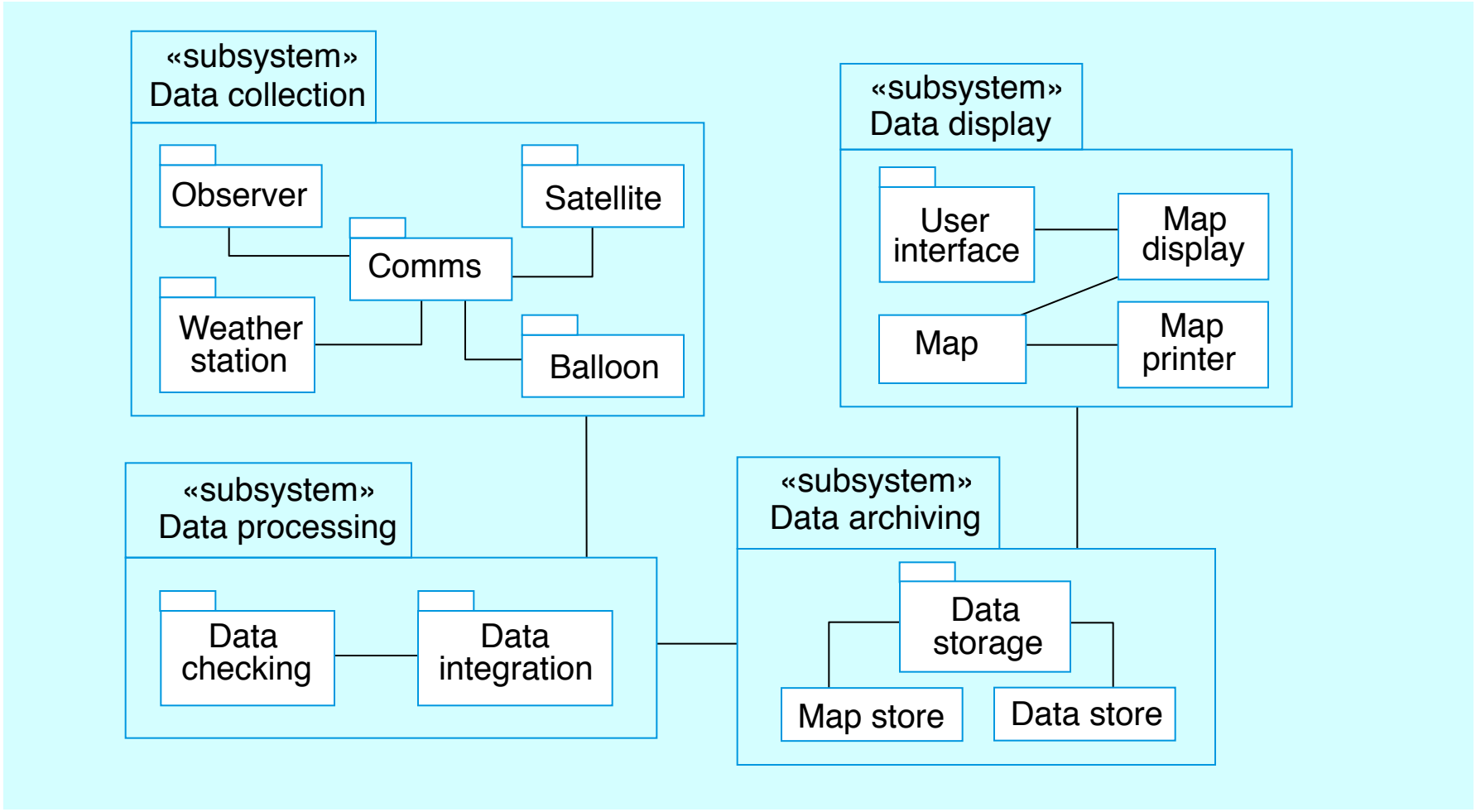
Data processing layer

収集されたデータをチェックし、統合するオブジェクト群

Data collection layer

遠隔のデータ源からデータを得るためのオブジェクト群

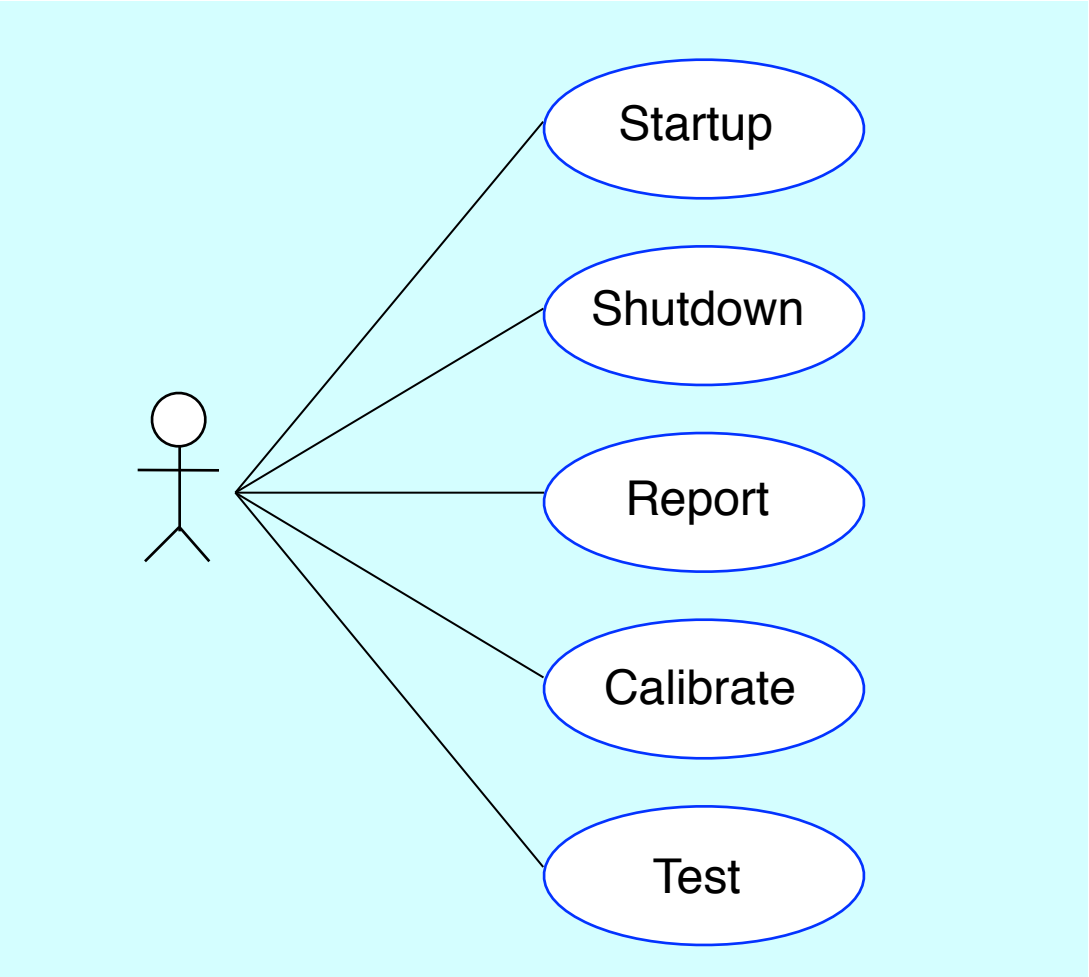
天気マップシステムのサブシステムモデル



ユースケースモデル

- ユースケースモデルにより，システムと外部とのやりとりを記述
- システム機能を楕円で，システムとやりとりする対象を人型の図で表す.

天気ステーションのユースケース



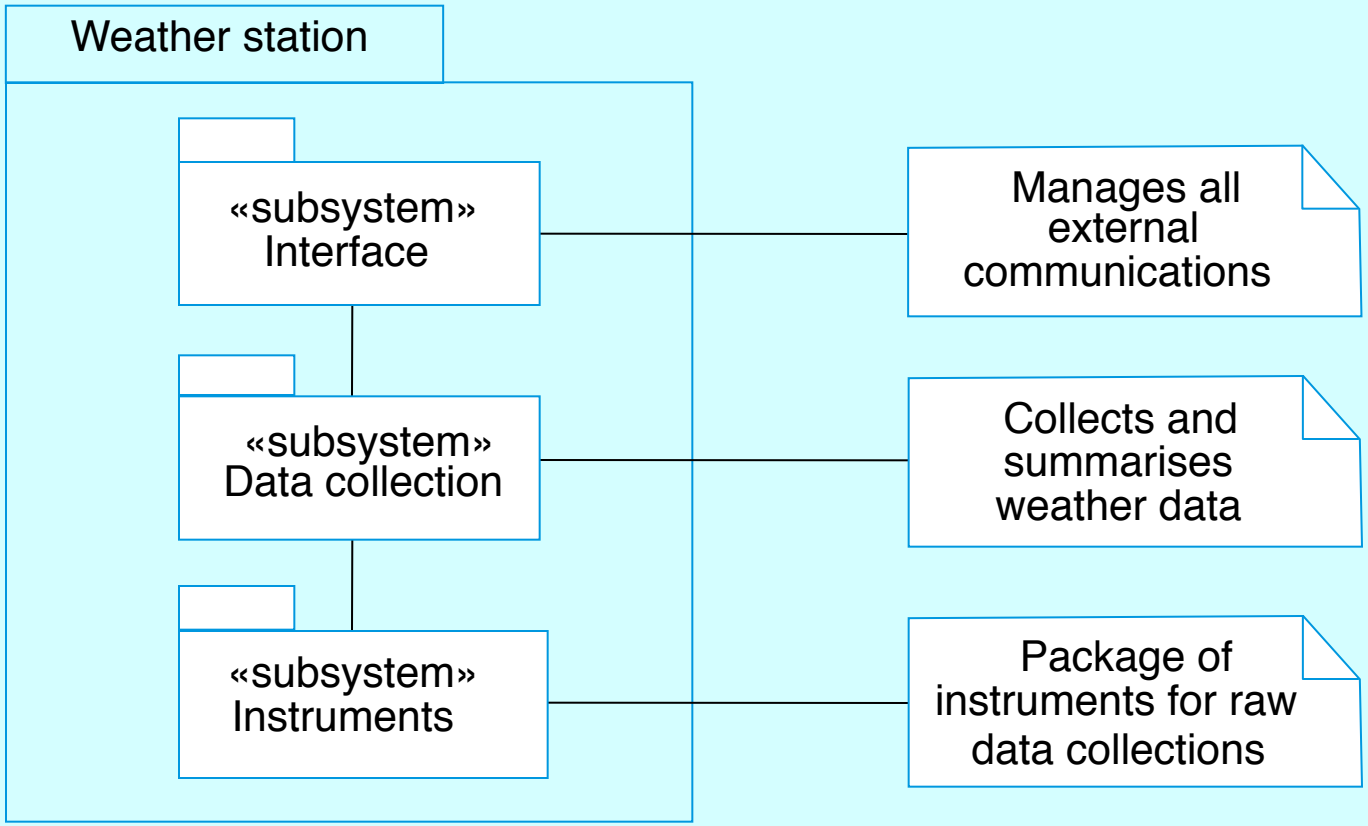
ユースケース記述

システム	天気ステーション
ユースケース	レポート
アクター	天気データ収集システム、天気ステーション
データ	天気ステーションは、データ収集周期に計器から集めた天気データの概要を、天気データ収集システムに送る。送られるデータは、地温と気温の最高・最低・平均、気圧の最高・最低・平均、風速の最高・最低・平均、５分間隔で計測した総雨量と風向である。
刺激 (stimulus)	天気データ収集システムは天気ステーションとモデムのリンクを確立し、データの送信を要求する。
反応	まとめられたデータは天気データ収集システムに送られる。
コメント	天気ステーションは通常、１時間に１度、レポートを要求するが、この頻度は天気ステーションごとに異なり、また、将来は変更されるかもしれない。

アーキテクチャ設計

- システムと環境とのやりとりを明示した後，その情報をシステムアーキテクチャの設計に利用.
- レイヤードアーキテクチャ（Chapter 11）が天気ステーションには適切
  - 通信を処理するインタフェース層
  - 計測機器を管理するデータ収集層
  - 天気データを集める計測機器層
- アーキテクチャモデルには一般的に７個以上の対象を含めないのが良いとされる.

天気ステーションのアーキテクチャ



オブジェクトの同定

- オブジェクト（オブジェクトクラス）を決めることがオブジェクト指向設計で最も難しいとされる.
- 魔法の公式はなく，システム設計者の能力，経験，領域に対する知識の深さに依存.
- 繰り返されるプロセスで，最初から良い結果を得ることはまれ.

オブジェクト同定の方針

- システムに対する自然言語記述を手がかりに，文法的な処理を行う．(HOOD方法論).
- 応用領域の中にある，目に見える・触ることのできる「もの」を基準にする.
- どの振舞いに何が参加するかということを手がかりにオブジェクトを見つける.
- シナリオに基づく分析を利用する．それぞれのシナリオにおいてオブジェクト，属性，メソッドを見つける.

天気ステーションの記述

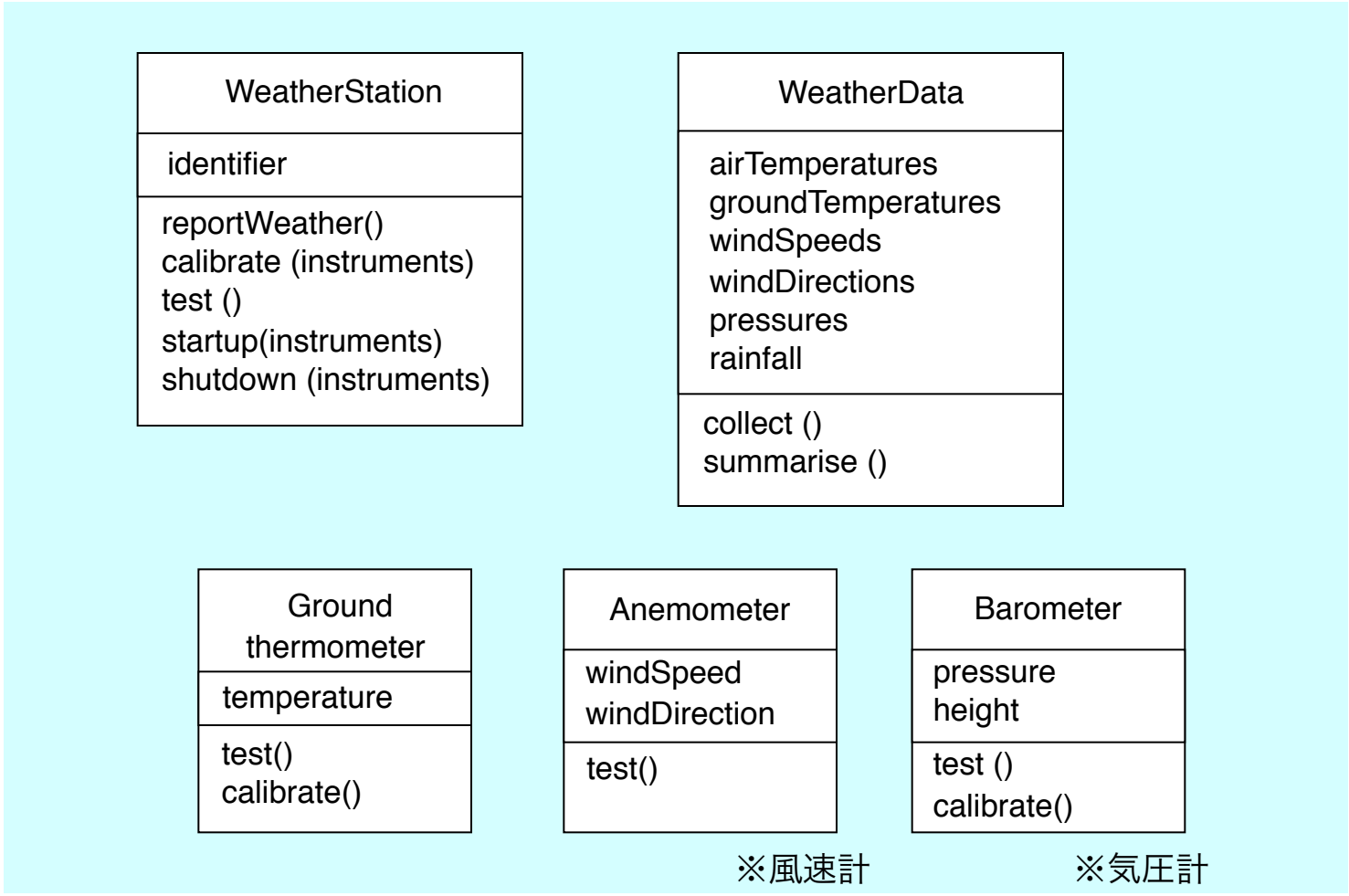
天気ステーションは，データを収集し，何らかのデータ処理を行い，基地でのさらなる処理のためにデータを送付するための，ソフトウェア制御による計測機器群のことである．計測機器には，気温計，地温計，風速計，風向計，気圧計，雨量計があり，データは定期的に収集される．

天気データの送信コマンドが発せられると，天気ステーションは収集したデータの処理と取りまとめを行う．まとめられたデータはエリアコンピュータからの要求に応じて転送される．

天気ステーションのオブジェクトクラス

- 地温計，風速計，気圧計
  - 計測機器のハードウェアに対応する応用領域のオブジェクト
- 天気ステーション
  - 天気ステーションと周囲の環境とのやりとりの基本的インタフェースを提供するオブジェクト．ユースケースに示されたやりとりを反映する．
- 天気データ
  - 各機器のデータを取りまとめた天気データのオブジェクト

天気ステーションのオブジェクトクラス



その他のオブジェクトとオブジェクトの洗練

- 応用領域に関する知識を利用してその他のオブジェクトやオペレーションを見つける
  - 天気ステーションには一意識別子が必要
  - 天気ステーションは遠隔地にあるので，機器の障害を自動的に報告しなければならない．つまり自己チェックのオペレーションが必要.
- アクティブオブジェクトかパッシブオブジェクトか
  - この場合，パッシブオブジェクトを採用し，自律的にではなく要求された時にデータを集めるようにする.
  - コントローラの処理時間が多くなる代わりにシステムの柔軟性が期待できる.

設計モデル

- オブジェクト・クラス・それらの関係を表現する.
- 静的モデルはオブジェクトクラスとそれらの関係によってシステムの静的構造を表現.
- 動的モデルはオブジェクト間のオブジェクト間のやり取りを表現.

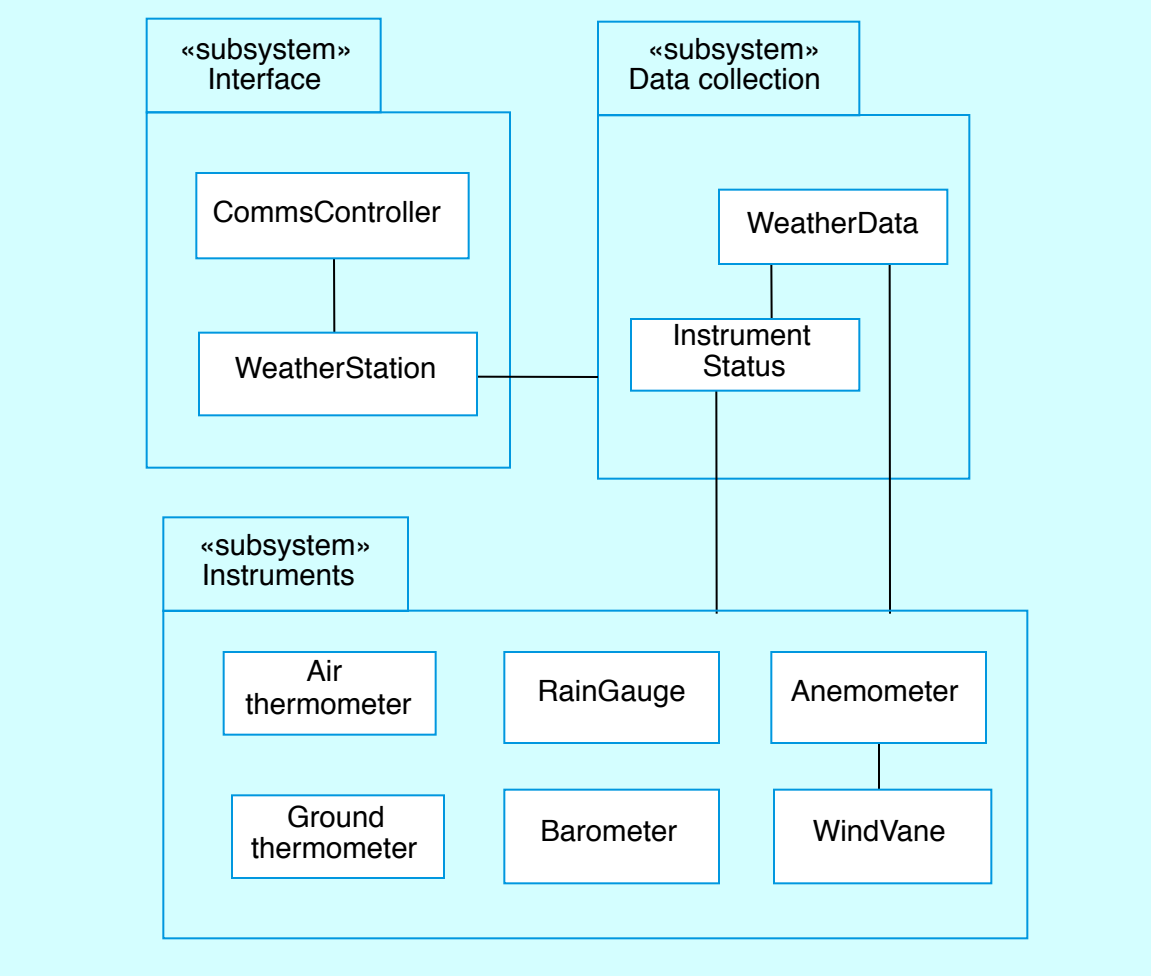
設計モデルの例

- サブシステムモデルにより，オブジェクトの論理的なまとまりをサブシステムとして表現
- シーケンスモデルによりオブジェクトのやりとりの系列を表現
- 状態機械モデルにより，個々のオブジェクトがイベントに対し状態を変化させる様子を表現
- 他のモデルには，ユースケースモデル，集約モデル，汎化関係モデルなどがある.

サブシステムモデル

- 設計結果のオブジェクトがどのように論理的にまとめられるかを示す.
- UMLでは パッケージ（カプセル化のための要素）図を用いる. これは論理的なモデルであり, オブジェクトの実際の構成はまた異なる可能性がある.

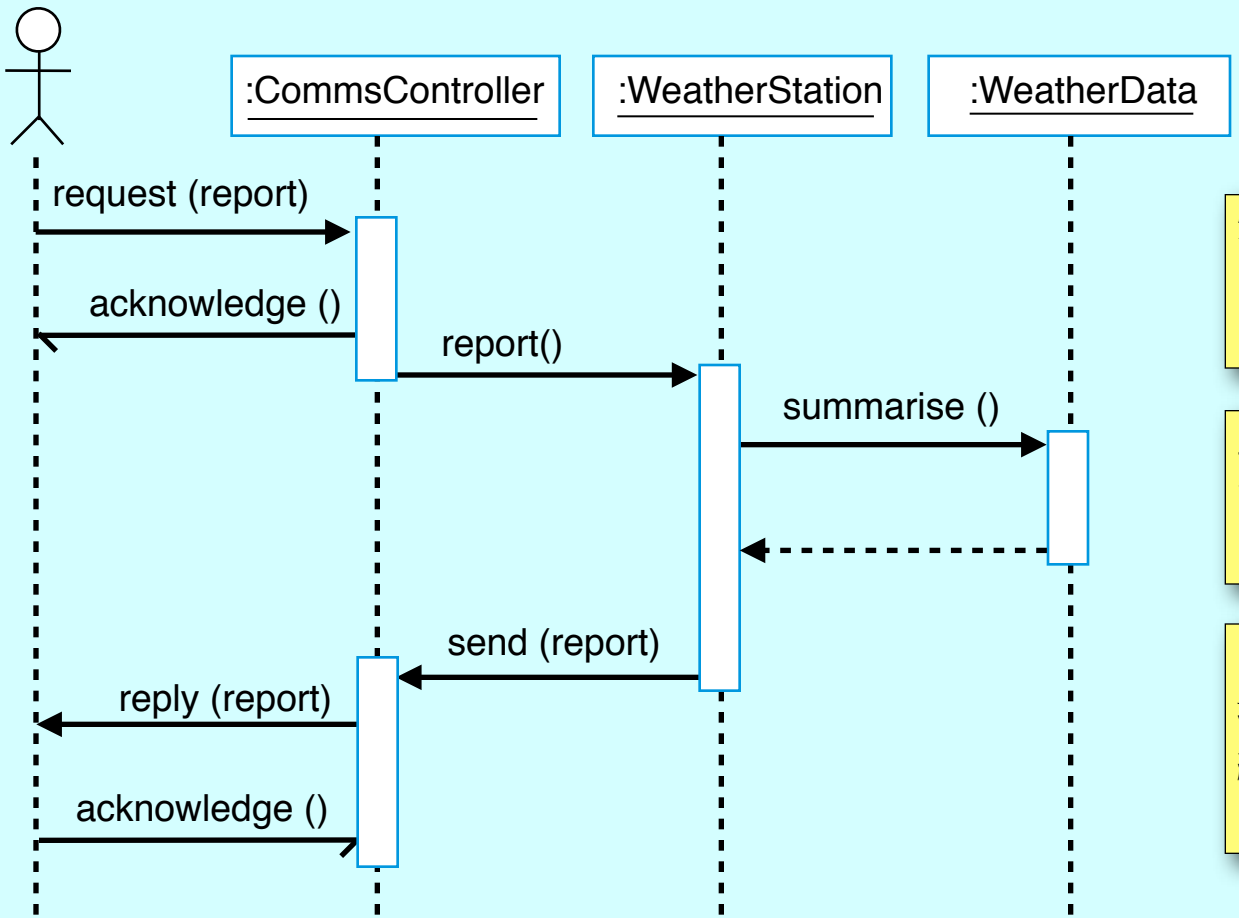
天気ステーションのサブシステム



シーケンスモデル

- オブジェクト間のやりとりを出現順に系列として表現
  - オブジェクトが縦に配置
  - 上から下へ時間に沿って読む
  - やりとりは名前つきの矢印で表現．矢印の種類によってやりとりの種類を表す.
  - 縦長の四角は，オブジェクトがシステム内で活性化し，何らかの制御を行っていることを示すライフライン.

データ収集のシーケンス



矢じりが黒いのは同期メッセージ、そうでないのは非同期メッセージ

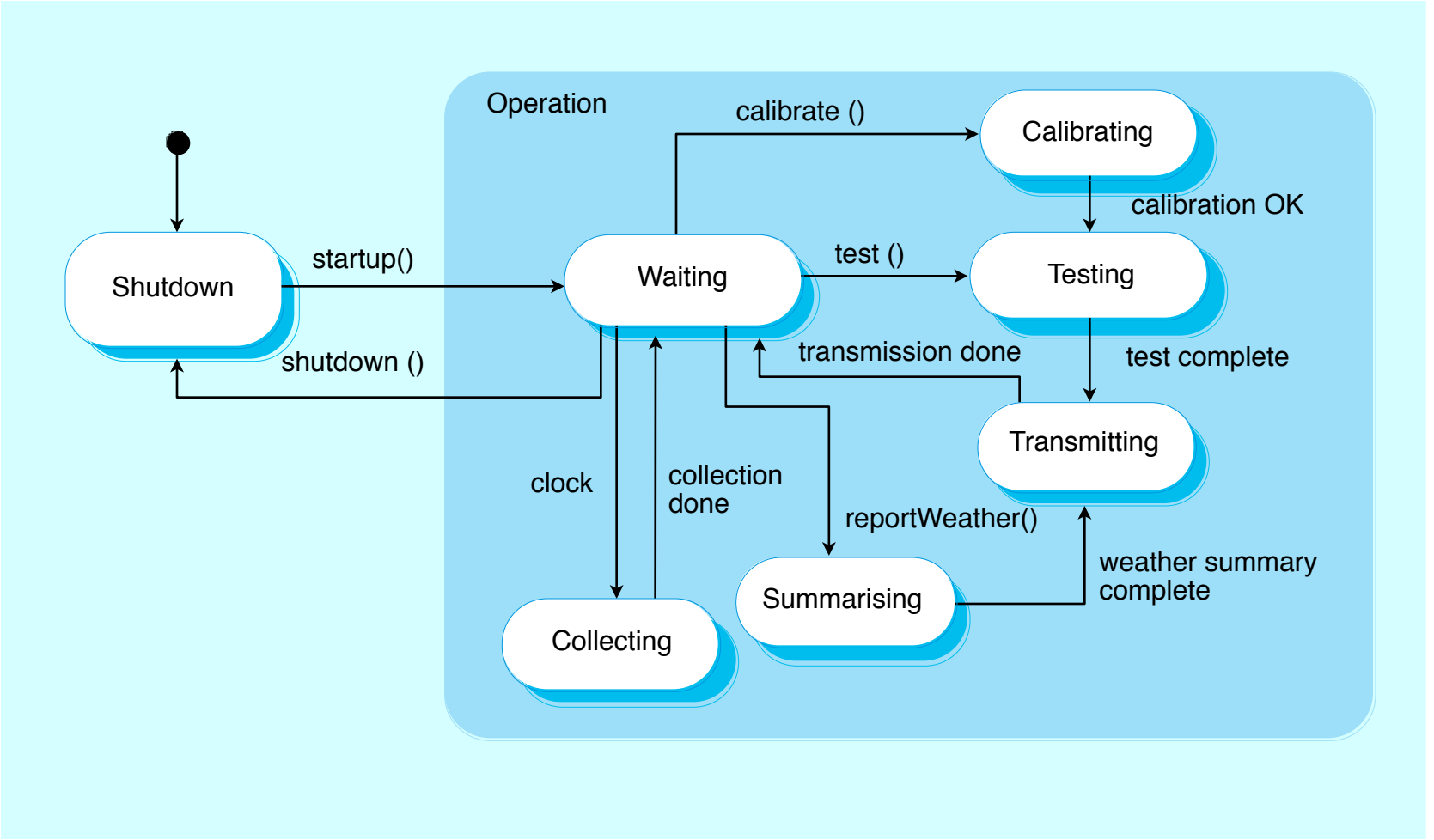
点線の矢印は、その時点で呼び出しが終了したことを表す。省略可。

オブジェクトが消滅する場合、縦の点線をそこで終える。さらに×印を書き加えてもよい。

ステートチャート

- 様々なサービス要求に対してオブジェクトがどう反応し，状態をどのように変化させるかを表現する.
  - オブジェクトの状態がShutdownならStartup() メッセージに反応する
  - waiting状態なら，オブジェクトは次のメッセージを待つ
  - reportWeather()が要求されるとシステムはsummarising状態に変化する
  - calibrate() が要求されるとシステムはcalibrating状態に変化する
  - collecting状態へは，クロックイベントが到着した時に遷移する

天気ステーションのステートチャート



オブジェクトインタフェースの定義

- オブジェクトや他のコンポーネントを並行して設計できるようにするためには，オブジェクトインタフェースを定義しなければならない.
- インタフェースの中身（実現）はオブジェクト内部に隠すように設計する.
- オブジェクトは複数のインタフェースをもつことができる．オブジェクトが提供するメソッドの視点に応じてインタフェースが定義される.
- UMLではクラス図を用いてインタフェースを記述する．Javaのインタフェース定義も用いられる.

天気ステーションのインタフェース

```
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
```

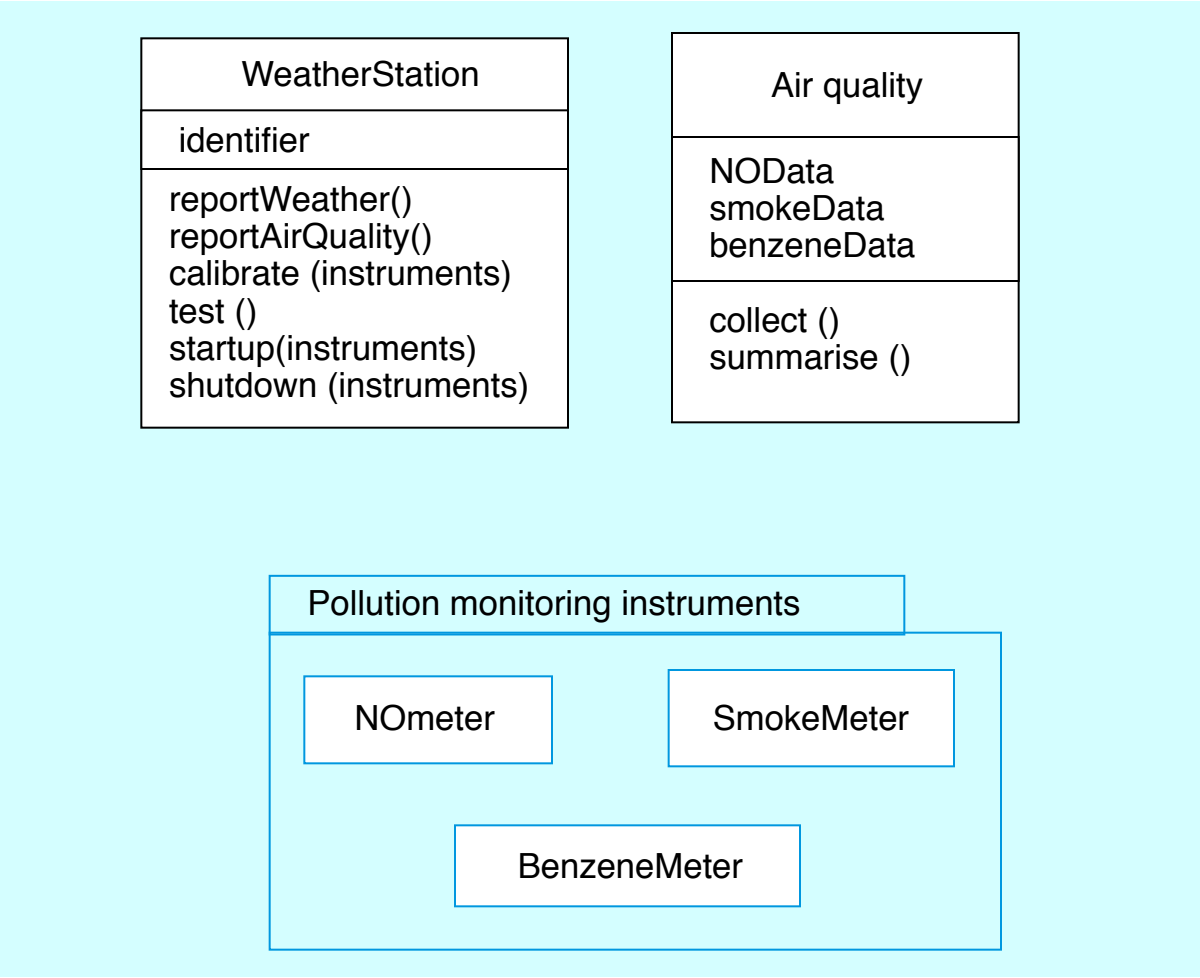
設計の発展

- 表現に関する情報をオブジェクト内部に隠すということは，そのオブジェクトの中身に関する変更が他のオブジェクトに予期しない形で波及することがないということ.
- 例えば，汚染監視の機能を天気ステーションに加えたいとする．つまり，空気のサンプルから，大気中の様々な汚染物質の量を計算する機器を加える.
- 汚染物質量は天気データとともに送付するものとする.

必要となる変更

- Air qualityというオブジェクトクラスを WeatherStation クラスの一部として追加
- reportAirQuality というオペレーションを WeatherStationに追加し，制御ソフトウェアで汚染物質量を収集するように変更.
- 汚染物質用の計測器に対応するオブジェクトを追加.

汚染監視


WeatherStation
identifier
reportWeather()
reportAirQuality()
calibrate (instruments)
test ()
startup(instruments)
shutdown (instruments)
Air quality
NOData
smokeData
benzeneData
collect ()
summarise ()
Pollution monitoring instruments
NOmeter
SmokeMeter
BenzeneMeter

要点

- OOD は，自身の状態とそれに対するオペレーションをもつオブジェクトという要素でシステム設計を行うアプローチ.
- オブジェクトは他のオブジェクトに対してサービスを提供する.
- オブジェクトの実装は逐次的であっても並行であっても構わない.
- Unified Modeling Languageはオブジェクトモデルに対する様々な記法を提供する.

要点

- オブジェクト指向設計では，システムに対する様々なモデルが作られる．それらのうちのいくつかは静的なモデルであり，いくつかは動的なモデルである．
- オブジェクトインタフェースはJavaのようなプログラミング言語を用いるなどして明確に定義しなければならない．
- オブジェクト指向設計によりシステム発展を容易にできる．

要点の要点

- オブジェクト指向の考え方
- オブジェクト指向設計のプロセス
- クラスの継承／集約／関連
- オブジェクト（クラス）モデル、ユースケースモデル、シーケンスモデル、状態遷移モデルの役割を説明できるか／記述できるか
- ソフトウェア発展に関するオブジェクト指向設計の利点について説明できるか