
Object-oriented Design

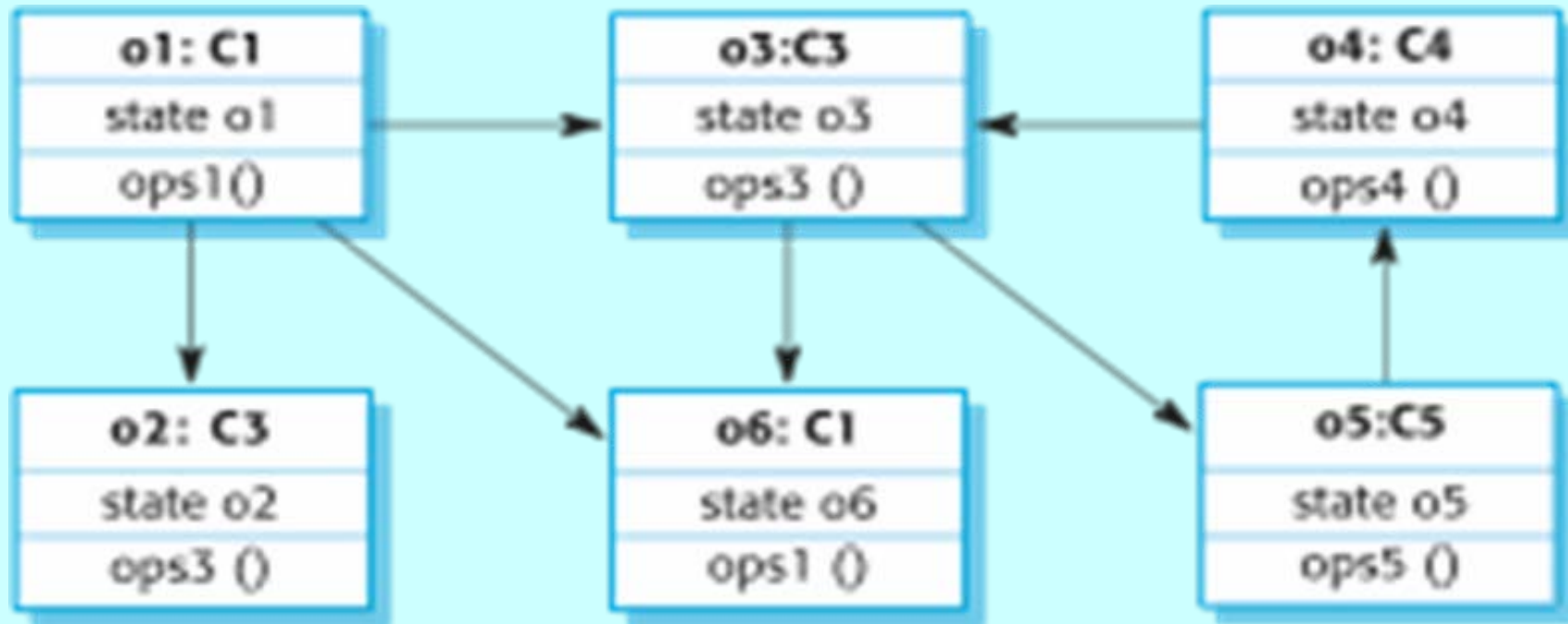
Objectives

- λ To explain how a software design may be represented as a set of interacting objects that manage their own state and operations
- λ To describe the activities in the object-oriented design process
- λ To introduce various models that can be used to describe an object-oriented design
- λ To show how the UML may be used to represent these models

Characteristics of OOD

- λ Objects are abstractions of real-world or system entities and manage themselves.
- λ Objects are independent and encapsulate state and representation information.
- λ System functionality is expressed in terms of object services.
- λ Shared data areas are eliminated. Objects communicate by message passing.
- λ Objects may be distributed and may execute sequentially or in parallel.

Interacting objects



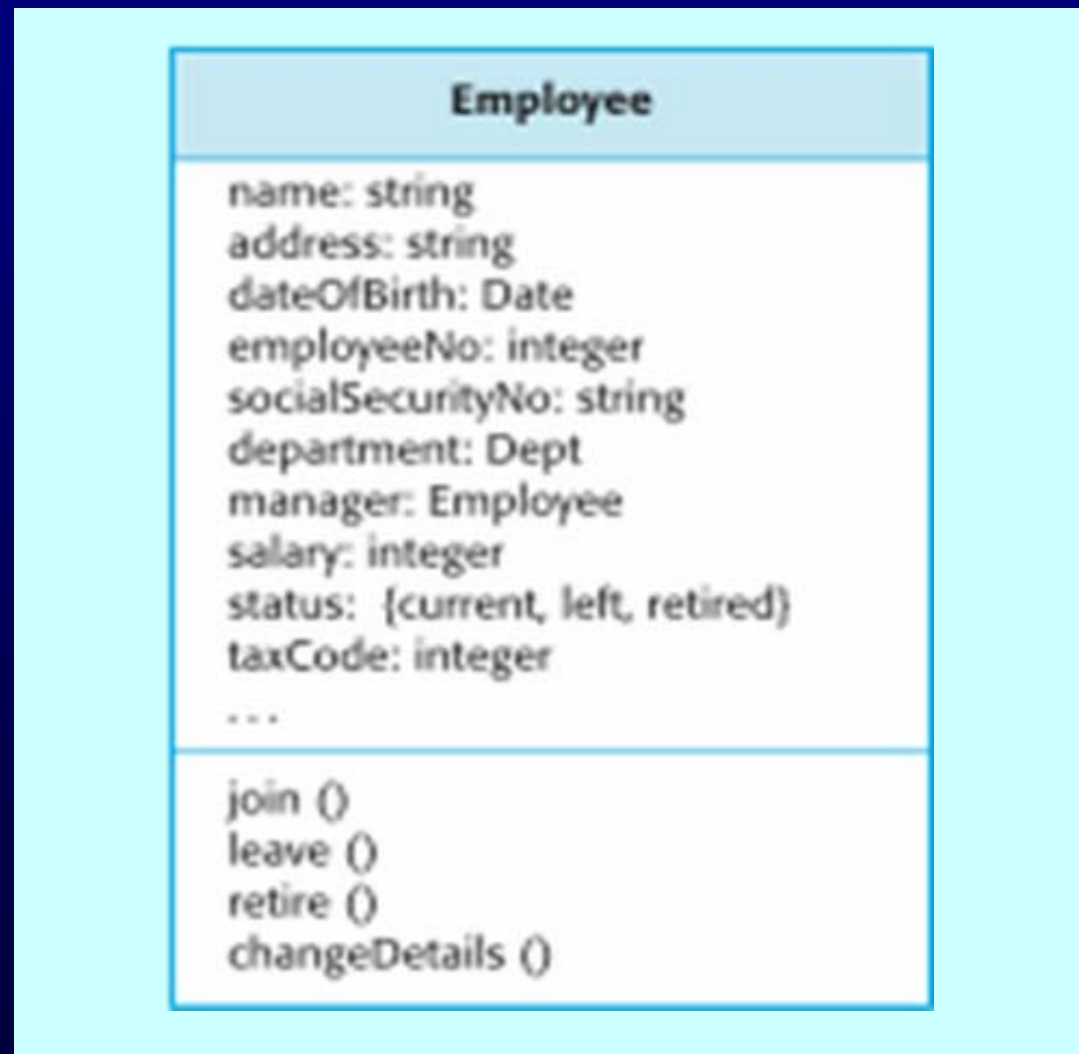
Advantages of OOD

- λ Easier maintenance. Objects may be understood as stand-alone entities.
- λ Objects are potentially reusable components.
- λ For some systems, there may be an obvious mapping from real world entities to system objects.

The Unified Modeling Language

- λ Several different notations for describing object-oriented designs were proposed in the 1980s and 1990s.
- λ The Unified Modeling Language is an integration of these notations.
- λ It describes notations for a number of different models that may be produced during OO analysis and design.
- λ It is now a *de facto* standard for OO modelling.

Employee object class (UML)



Object communication

- λ Conceptually, objects communicate by message passing.
- λ Messages
 - The name of the service requested by the calling object;
 - Copies of the information required to execute the service and the name of a holder for the result of the service.
- λ In practice, messages are often implemented by procedure calls
 - Name = procedure name;
 - Information = parameter list.

Message examples

```
// Call a method associated with a buffer  
// object that returns the next value  
// in the buffer
```

```
v = circularBuffer.Get ();
```

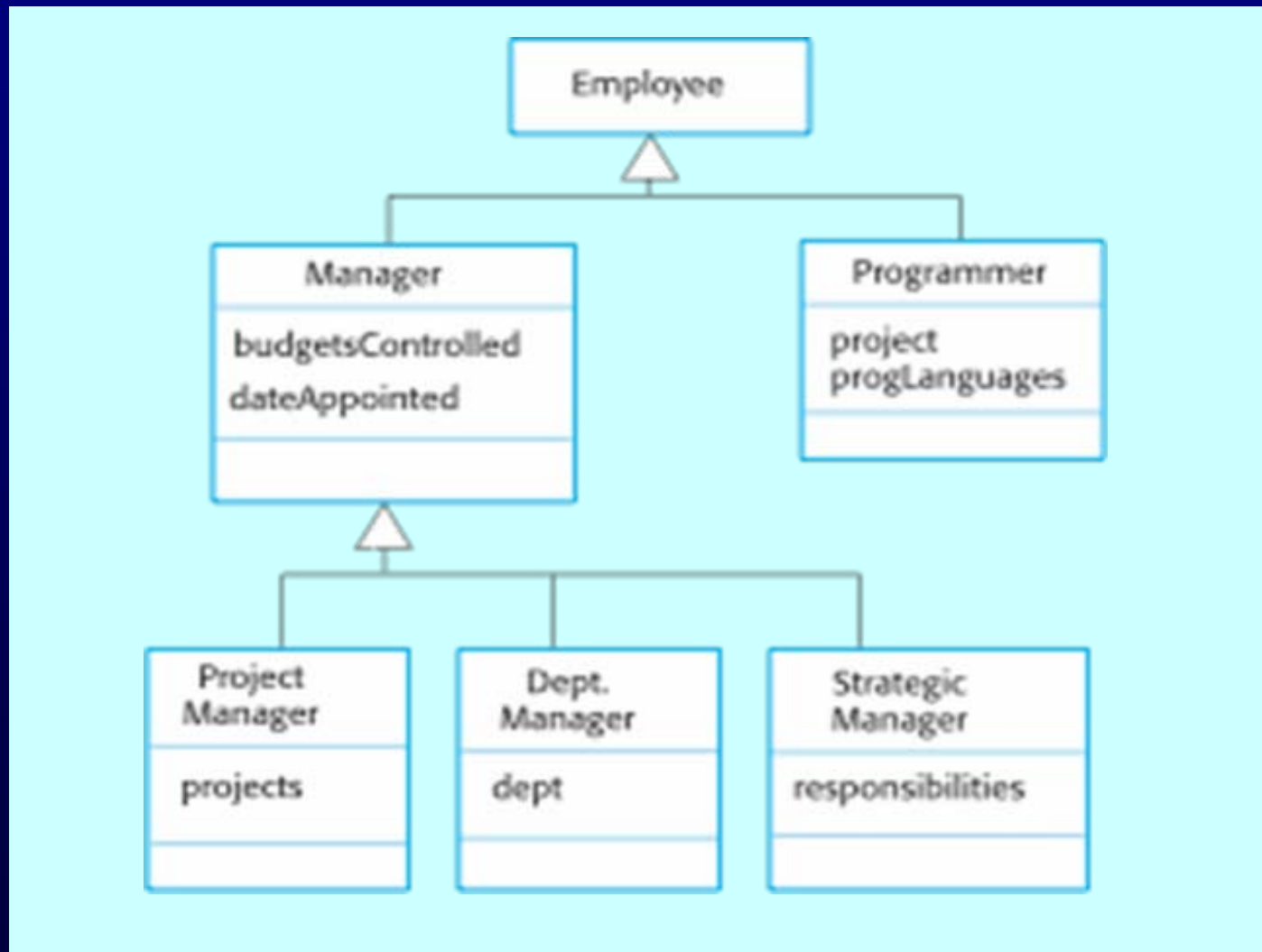
```
// Call the method associated with a  
// thermostat object that sets the  
// temperature to be maintained
```

```
thermostat.setTemp (20) ;
```

Generalisation and inheritance

- λ Objects are members of classes that define attribute types and operations.
- λ Classes may be arranged in a class hierarchy where one class (a super-class) is a generalisation of one or more other classes (sub-classes).
- λ A sub-class inherits the attributes and operations from its super class and may add new methods or attributes of its own.
- λ Generalisation in the UML is implemented as inheritance in OO programming languages.

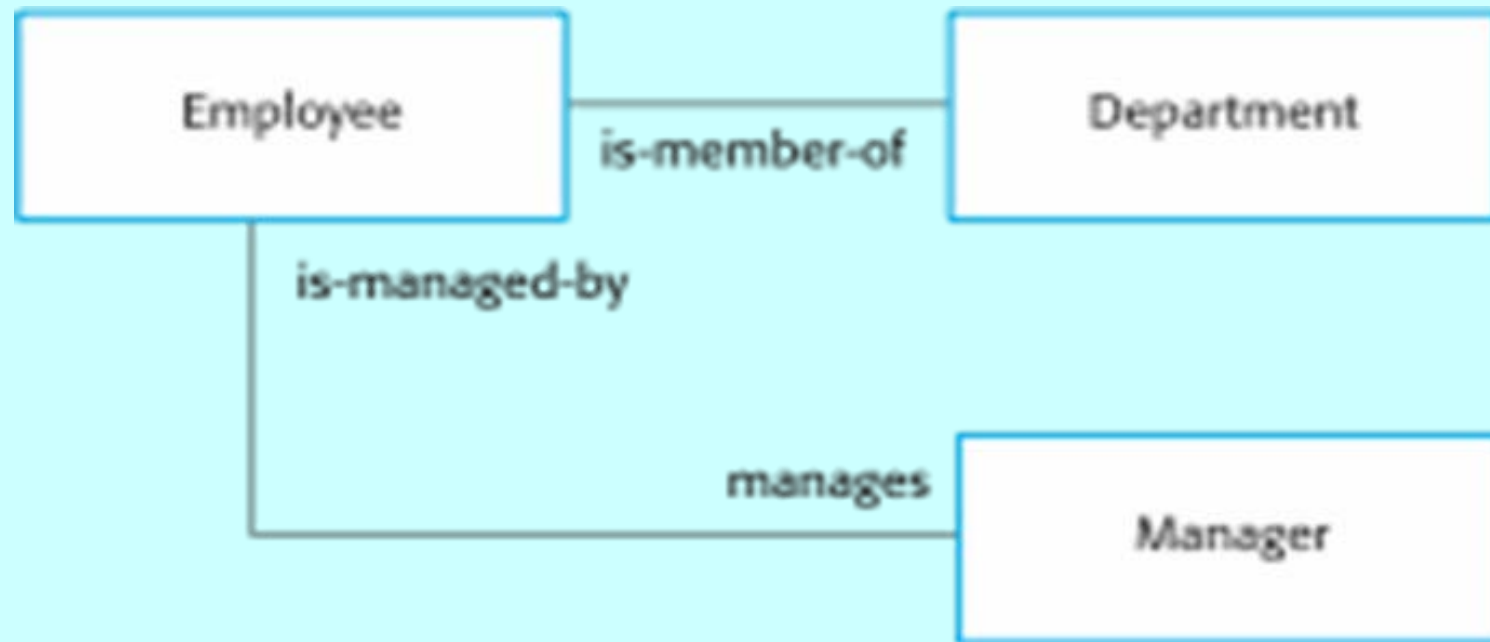
A generalisation hierarchy



UML associations

- λ Objects and object classes participate in relationships with other objects and object classes.
- λ In the UML, a generalised relationship is indicated by an association.
- λ Associations may be annotated with information that describes the association.
- λ Associations are general but may indicate that an attribute of an object is an associated object or that a method relies on an associated object.

An association model



An object-oriented design process

- λ Structured design processes involve developing a number of different system models.
- λ They require a lot of effort for development and maintenance of these models and, for small systems, this may not be cost-effective.
- λ However, for large systems developed by different groups design models are an essential communication mechanism.

Weather station description

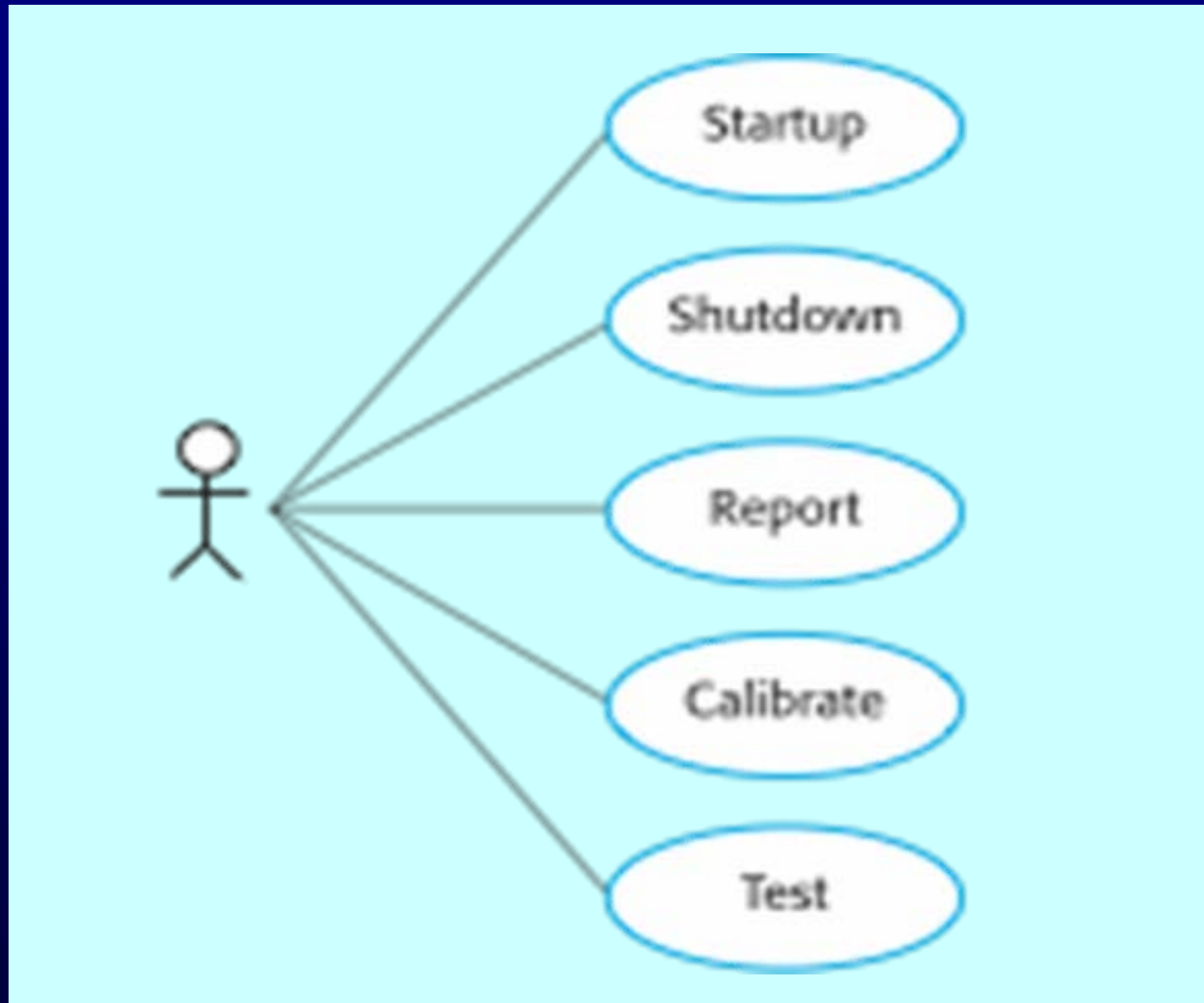
A **weather station** is a package of software controlled instruments which collects data, performs some data processing and transmits this data for further processing. The instruments include air and ground thermometers, an anemometer, a wind vane, a barometer and a rain gauge. Data is collected periodically.

When a command is issued to transmit the weather data, the weather station processes and summarises the collected data. The summarised data is transmitted to the mapping computer when a request is received.

Use-case models

- λ Use-case models are used to represent each interaction with the system.
- λ A use-case model shows the system features as ellipses and the interacting entity as a stick figure.

Use-cases for the weather station



Use-case description

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.

Object identification

- λ Identifying objects (or object classes) is the most difficult part of object oriented design.
- λ There is no 'magic formula' for object identification. It relies on the skill, experience and domain knowledge of system designers.
- λ Object identification is an iterative process. You are unlikely to get it right first time.

Approaches to identification

- λ Use a grammatical approach based on a natural language description of the system (used in Hood OOD method).
- λ Base the identification on tangible things in the application domain.
- λ Use a behavioural approach and identify objects based on what participates in what behaviour.
- λ Use a scenario-based analysis. The objects, attributes and methods in each scenario are identified.

Weather station description

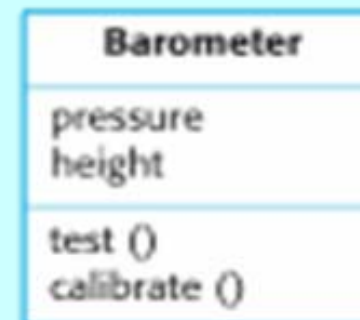
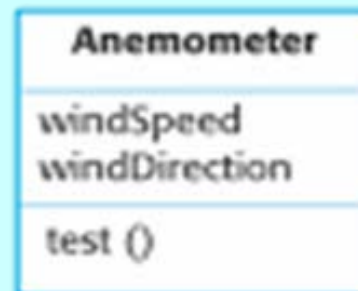
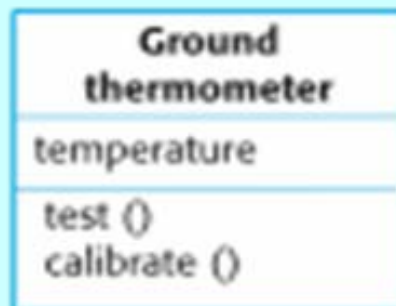
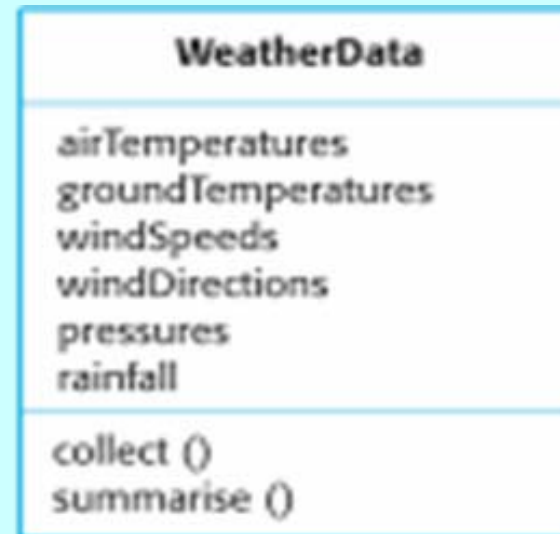
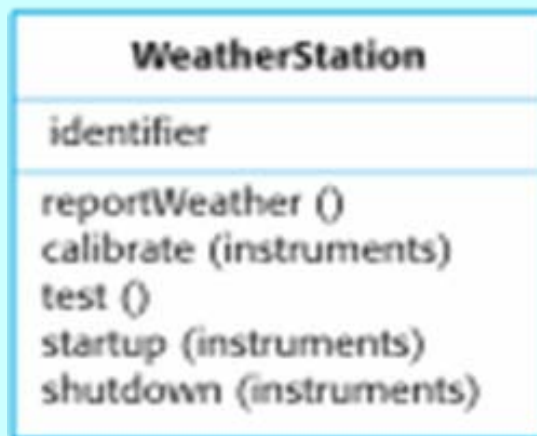
A **weather station** is a package of software controlled instruments which collects data, performs some data processing and transmits this data for further processing. The instruments include air and ground thermometers, an anemometer, a wind vane, a barometer and a rain gauge. Data is collected periodically.

When a command is issued to transmit the weather data, the weather station processes and summarises the collected data. The summarised data is transmitted to the mapping computer when a request is received.

Weather station object classes

- λ Ground thermometer, Anemometer, Barometer
 - Application domain objects that are 'hardware' objects related to the instruments in the system.
- λ Weather station
 - The basic interface of the weather station to its environment. It therefore reflects the interactions identified in the use-case model.
- λ Weather data
 - Encapsulates the summarised data from the instruments.

Weather station object classes



Further objects and object refinement

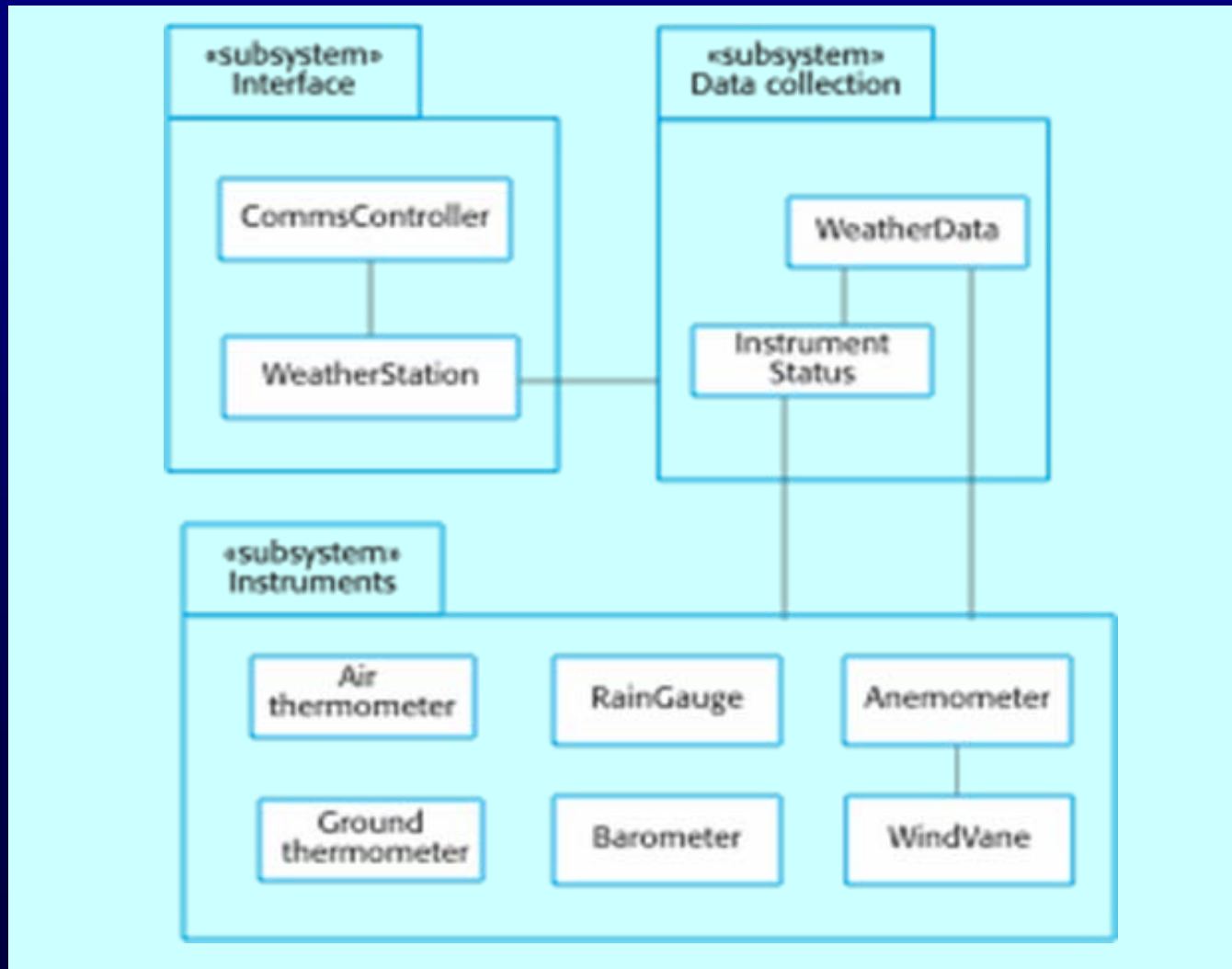
- λ Use domain knowledge to identify more objects and operations
 - Weather stations should have a unique identifier;
 - Weather stations are remotely situated so instrument failures have to be reported automatically. Therefore attributes and operations for self-checking are required.

- λ Active or passive objects
 - In this case, objects are passive and collect data on request rather than autonomously. This introduces flexibility at the expense of controller processing time.

Subsystem models

- λ Shows how the design is organised into logically related groups of objects.
- λ In the UML, these are shown using packages - an encapsulation construct. This is a logical model. The actual organisation of objects in the system may be different.
- λ A subsystem model should have low coupling (interactions between classes in different subsystems) and high cohesion (consistency between classes in the same subsystem).

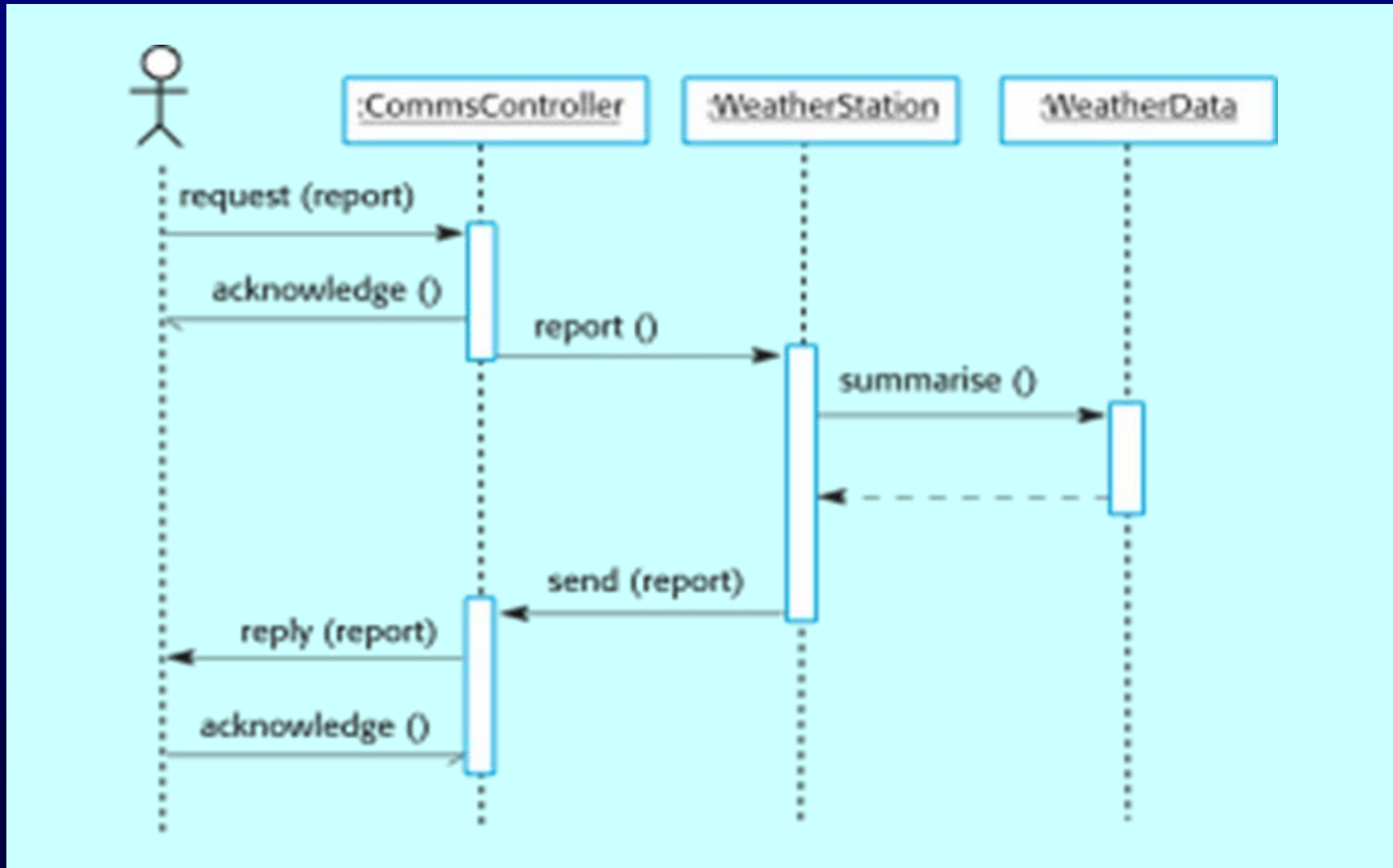
Weather station subsystems



Sequence models

- λ Sequence models show the sequence of object interactions that take place
 - Objects are arranged horizontally across the top;
 - Time is represented vertically so models are read top to bottom;
 - Interactions are represented by labelled arrows, Different styles of arrow represent different types of interaction;
 - A thin rectangle in an object lifeline represents the time when the object is the controlling object in the system.

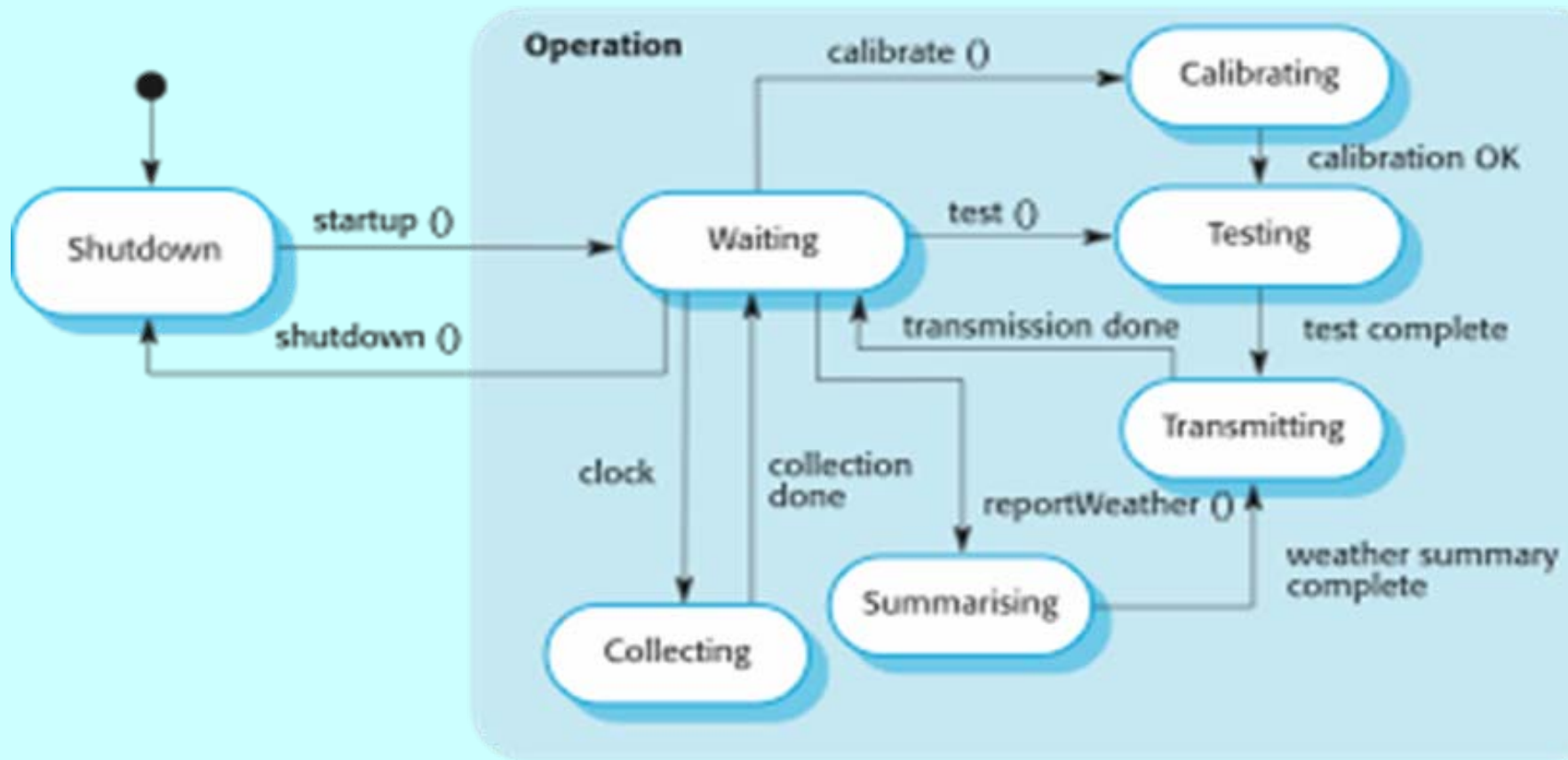
Data collection sequence



Statecharts

- λ Show how objects respond to different service requests and the state transitions triggered by these requests
 - If object state is Shutdown then it responds to a Startup() message;
 - In the waiting state the object is waiting for further messages;
 - If reportWeather () then system moves to summarising state;
 - If calibrate () the system moves to a calibrating state;
 - A collecting state is entered when a clock signal is received.

Weather station state diagram



Object interface specification

- λ Object interfaces have to be specified so that the objects and other components can be designed in parallel.
- λ Designers should avoid designing the interface representation but should hide this in the object itself.
- λ Objects may have several interfaces which are viewpoints on the methods provided.
- λ The UML uses class diagrams for interface specification but Java may also be used.

Weather station interface

```
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
```

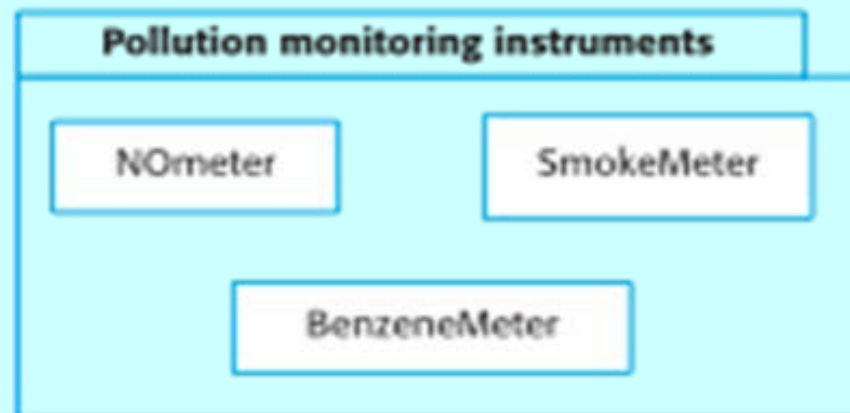
Design evolution

- λ Hiding information inside objects means that changes made to an object do not affect other objects in an unpredictable way.
- λ Assume pollution monitoring facilities are to be added to weather stations. These sample the air and compute the amount of different pollutants in the atmosphere.
- λ Pollution readings are transmitted with weather data.

Changes required

- λ Add an object class called **Air quality** as part of **WeatherStation**.
- λ Add an operation **reportAirQuality** to **WeatherStation**. Modify the control software to collect pollution readings.
- λ Add objects representing pollution monitoring instruments.

Pollution monitoring



Key points

- λ OOD is an approach to design so that design components have their own private state and operations.
- λ Objects should have constructor and inspection operations. They provide services to other objects.
- λ Objects may be implemented sequentially or concurrently.
- λ The Unified Modeling Language provides different notations for defining different object models.

Key points

- λ A range of different models may be produced during an object-oriented design process. These include static and dynamic system models.
- λ Object interfaces should be defined precisely using e.g. a programming language like Java.
- λ Object-oriented design potentially simplifies system evolution.