2008 Fall

Software Modeling & Analysis

Part 3. Design

- Application Architectures
- Object-Oriented Design

Lecturer: JUNBEOM YOO
jbyoo@konkuk.ac.kr

※ This lecture note is based on materials from Ian Sommerville 2006.

Chapter 13.
Application Architectures

Objectives

- To explain two fundamental models of business systems - batch processing and transaction processing systems
- To describe abstract architecture of resource management systems
- To explain how generic editors are event processing systems
- To describe structure of language processing systems

Generic Application Architectures

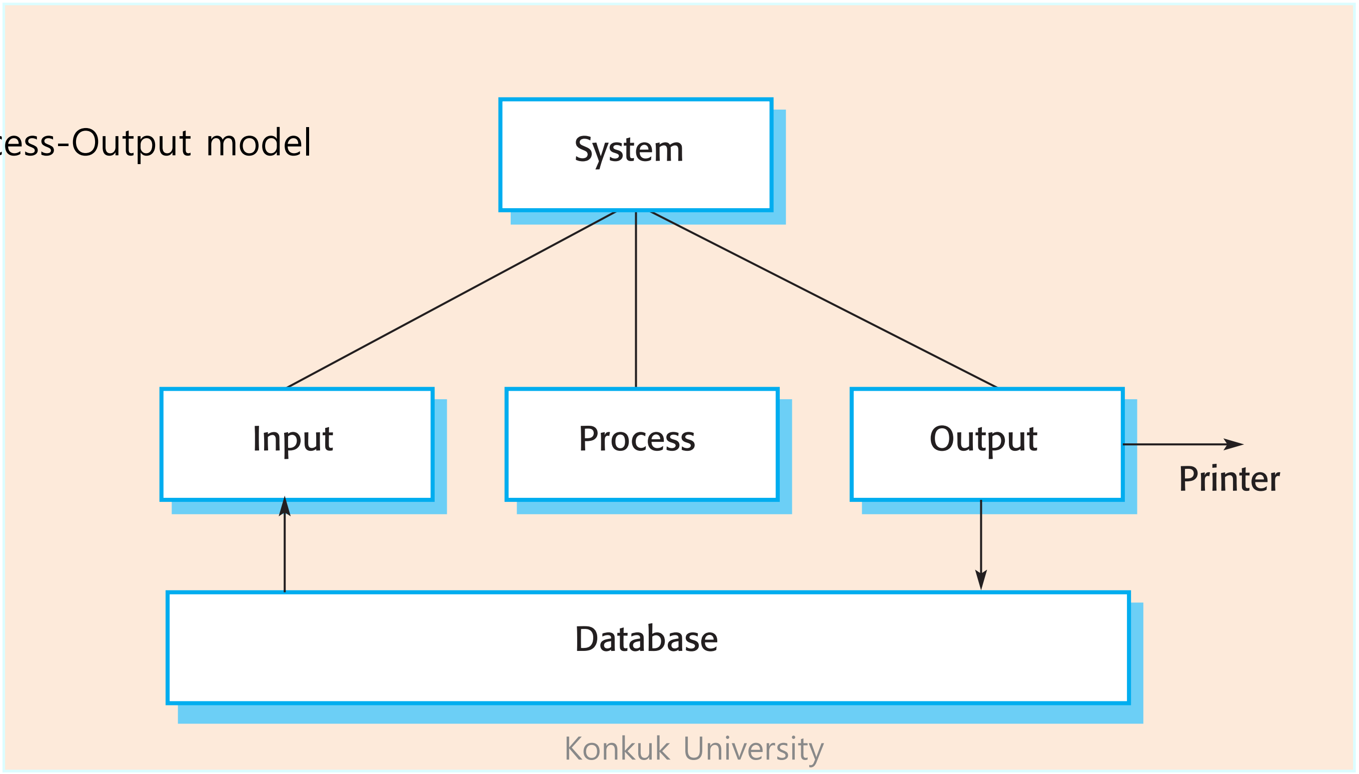
- Application systems are designed to meet an organizational need.
- As businesses have much in common, their application systems also tend to have a common architecture that reflects the application requirements.

- Application types:
  1. <u>Data processing applications</u>
     - Data driven applications that process data in batches without explicit user intervention during the processing. Ex) Billing system, Payroll system
  2. <u>Transaction processing applications</u>
     - Data-centered applications that process user requests and update information in a system database. Ex) E-commerce system, Reservation system
  3. <u>Event processing systems</u>
     - System actions depend on interpreting events from the system's environment. Ex) Word processor, Real-time system
  4. <u>Language processing systems</u>
     - Users' intentions are specified in a formal language that is processed and interpreted by the system. Ex) Compiler, Command interpreter

1. Data Processing System

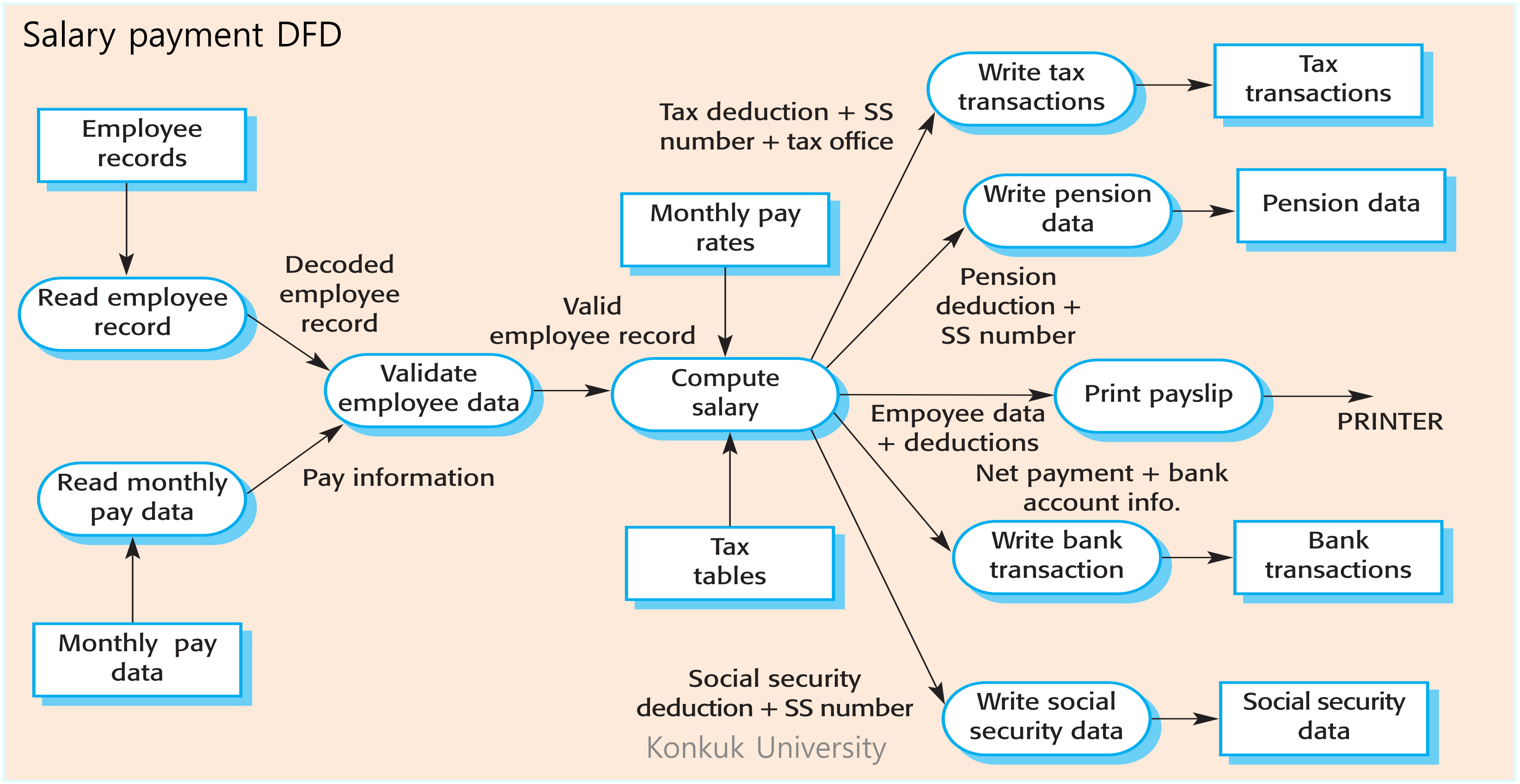
- Systems that are data-centered where the databases used are usually orders of magnitude larger than the software itself.
- Data is input and output in batches.
- Data processing systems usually have an input-process-output structure.

Input-Process-Output model



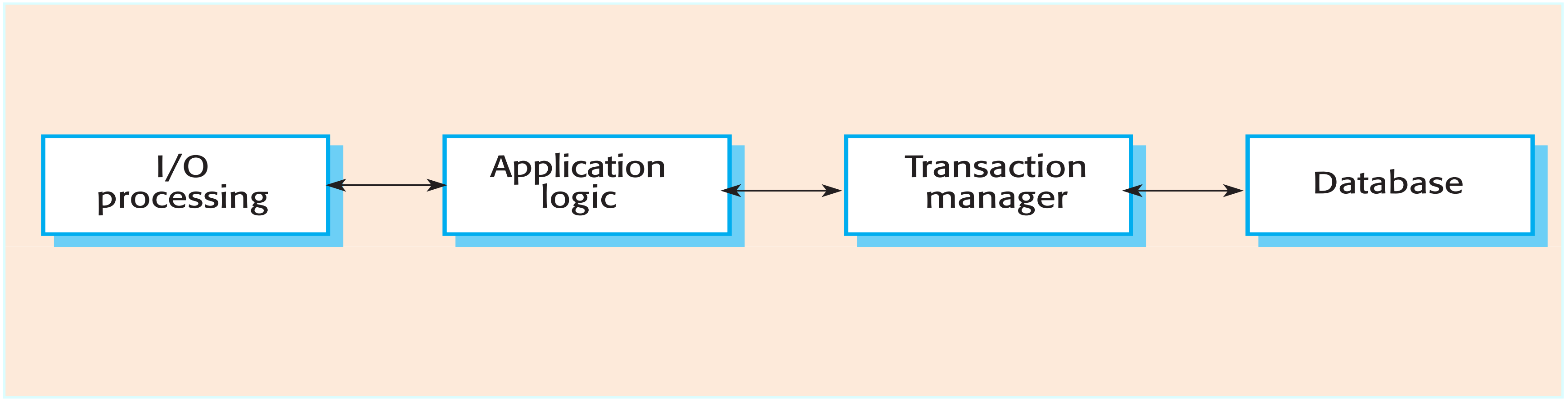
Data-Flow Diagram

- Show how data is processed as it moves through a system.
- Transformations are represented as round-edged rectangles, data-flows as arrows between them and files/data stores as rectangles.



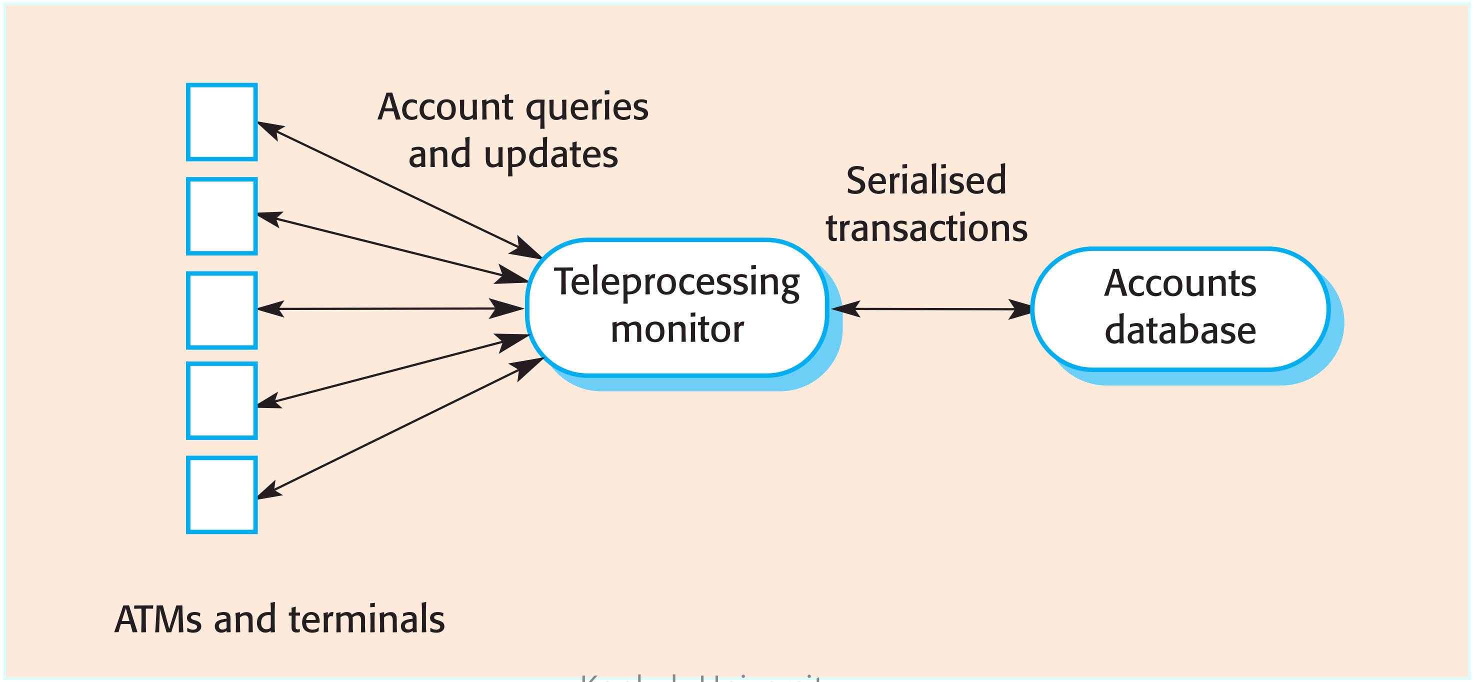
2. Transaction Processing System

- Processes user requests for information from a database or requests to update the database.
- From a user perspective a transaction is:
  - Any coherent sequence of operations that satisfies a goal
- Users make asynchronous requests for service which are then processed by a transaction manager.



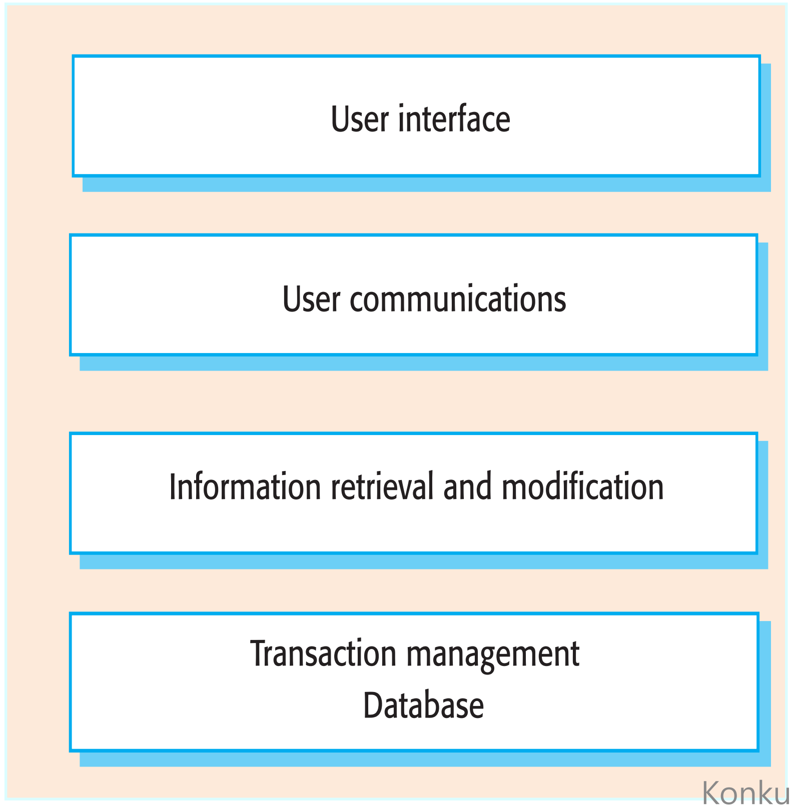
Transaction Processing Middleware

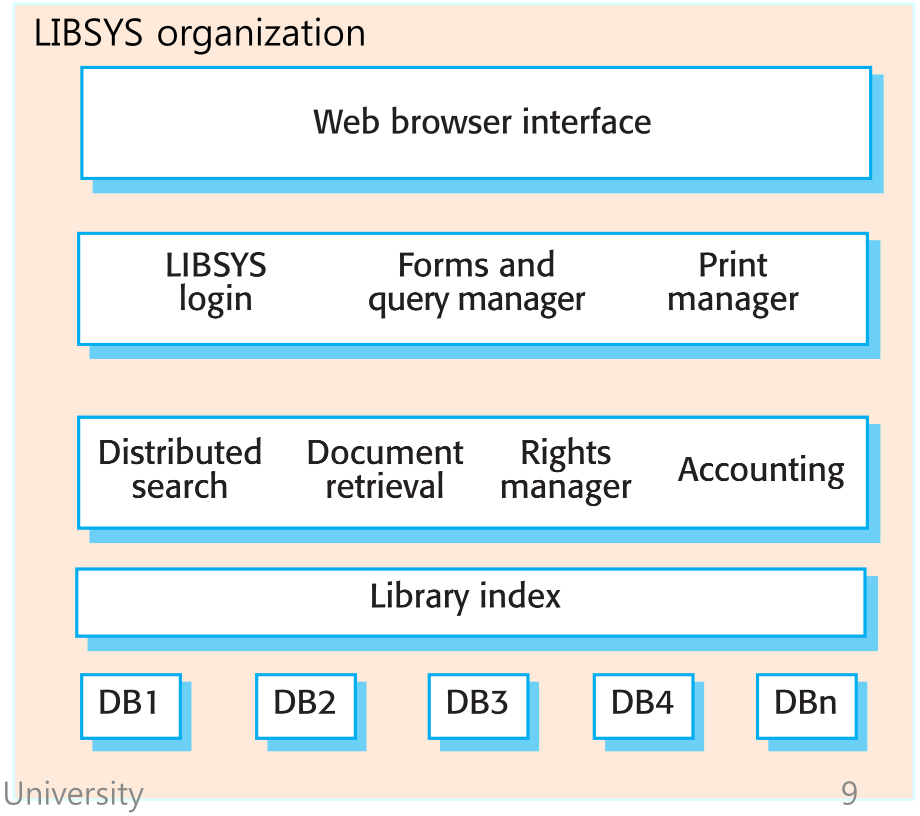
- Transaction management middleware or teleprocessing monitors handle communications with different terminal types (e.g. ATMs and counter terminals), serializes data and sends it for processing.
- Query processing takes place in the system database and results are sent back through the transaction manager to the user's terminal.



Information System Architecture

- Information systems have a generic architecture that can be organized as a layered architecture.
- Layers include:

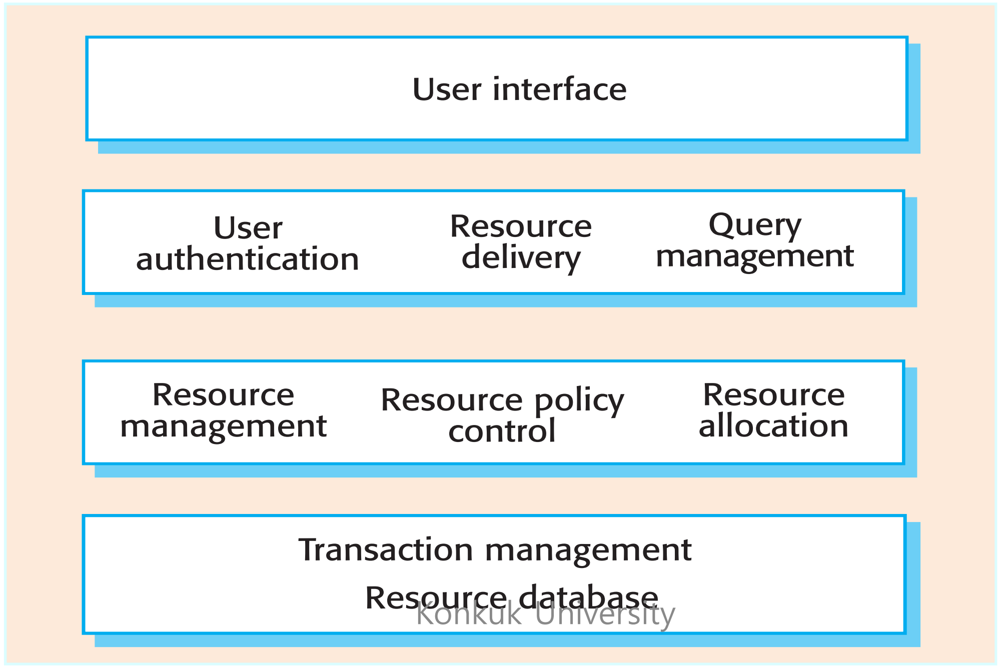




Resource Allocation System

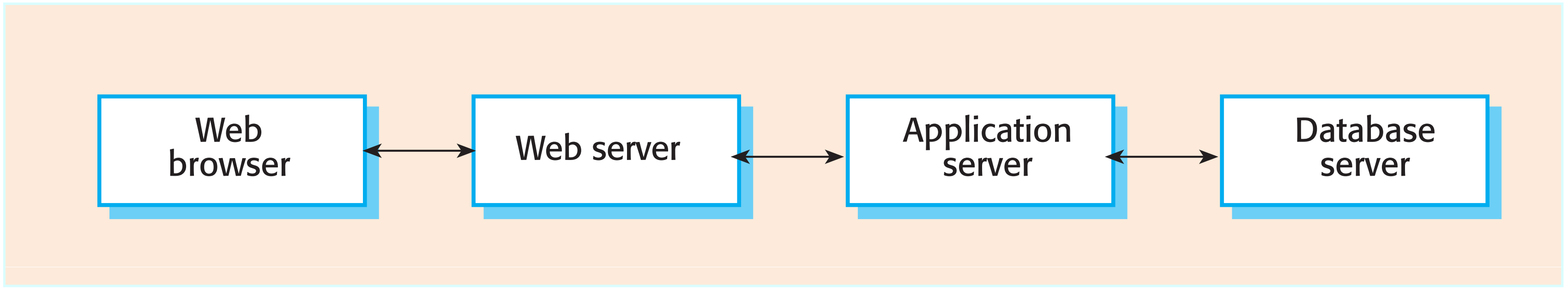
- Manages fixed amount of some resource and allocate this to users.
- Examples of resource allocation systems:
  - Timetabling systems where the resource being allocated is a time period
  - Library systems where the resource being managed is books for loan
  - Air traffic control systems where the resource being managed is the airspace

- Layer resource allocation architecture



E-commerce System Architecture

- E-commerce systems are Internet-based resource management systems that accept electronic orders for goods or services.
- They are usually organized using a multi-tier architecture with application layers associated with each tier.

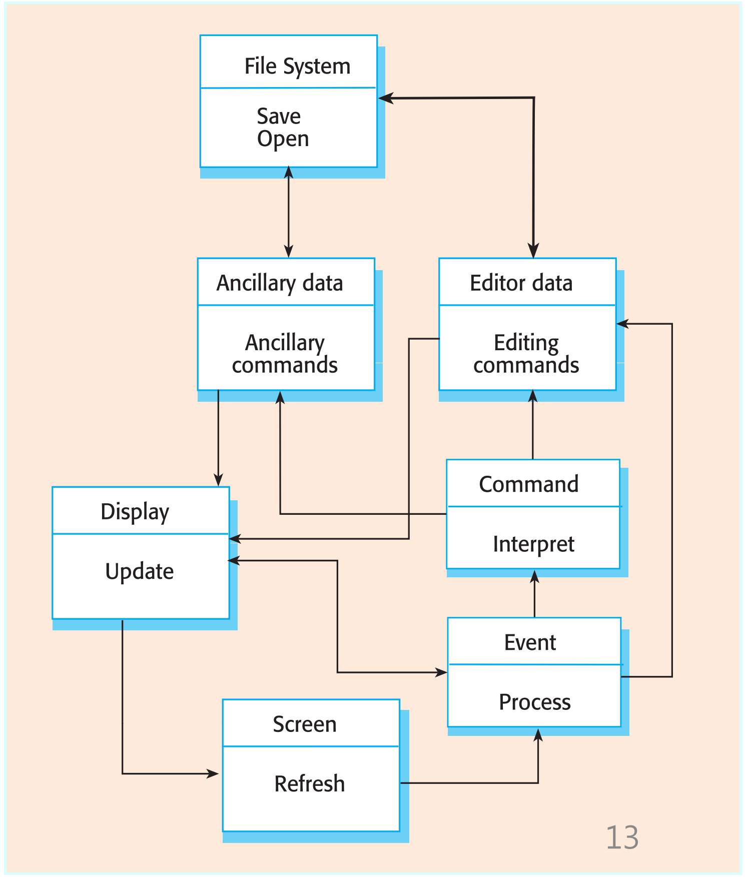


3. Event Processing Systems

- These systems respond to events in the system’s environment.
- Their key characteristic is that event timing is unpredictable so the architecture has to be organized to handle this.
- Many common systems: word processors, games, etc.

Editing System

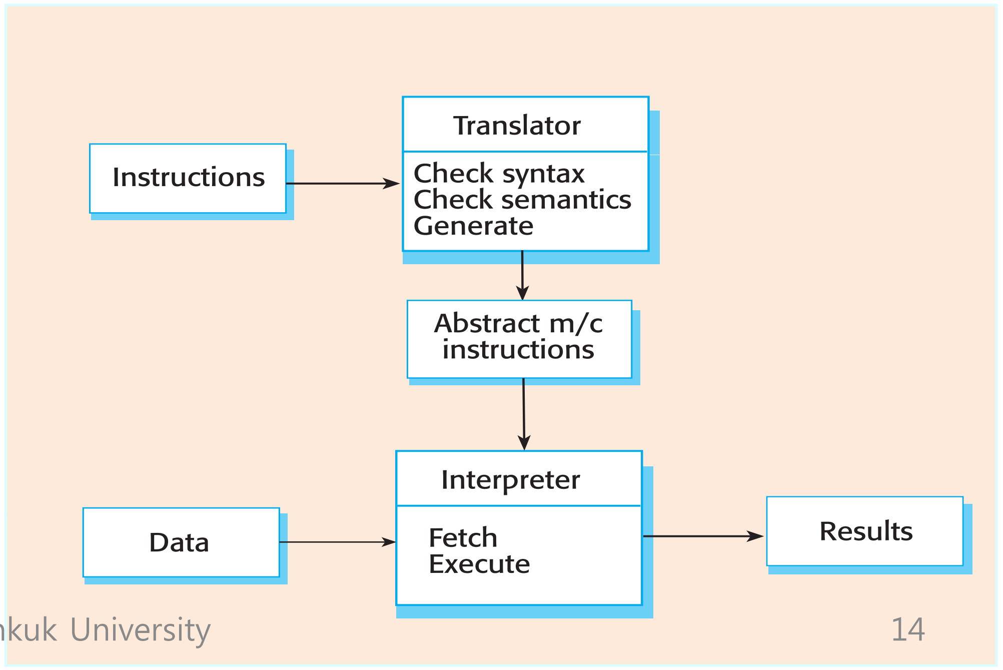
- Real-time systems and editing systems are the most common types of event processing system.
- Editing system characteristics:
  - Single user systems
  - Must provide rapid feedback to user actions
  - Organized around long transactions so may include recovery facilities



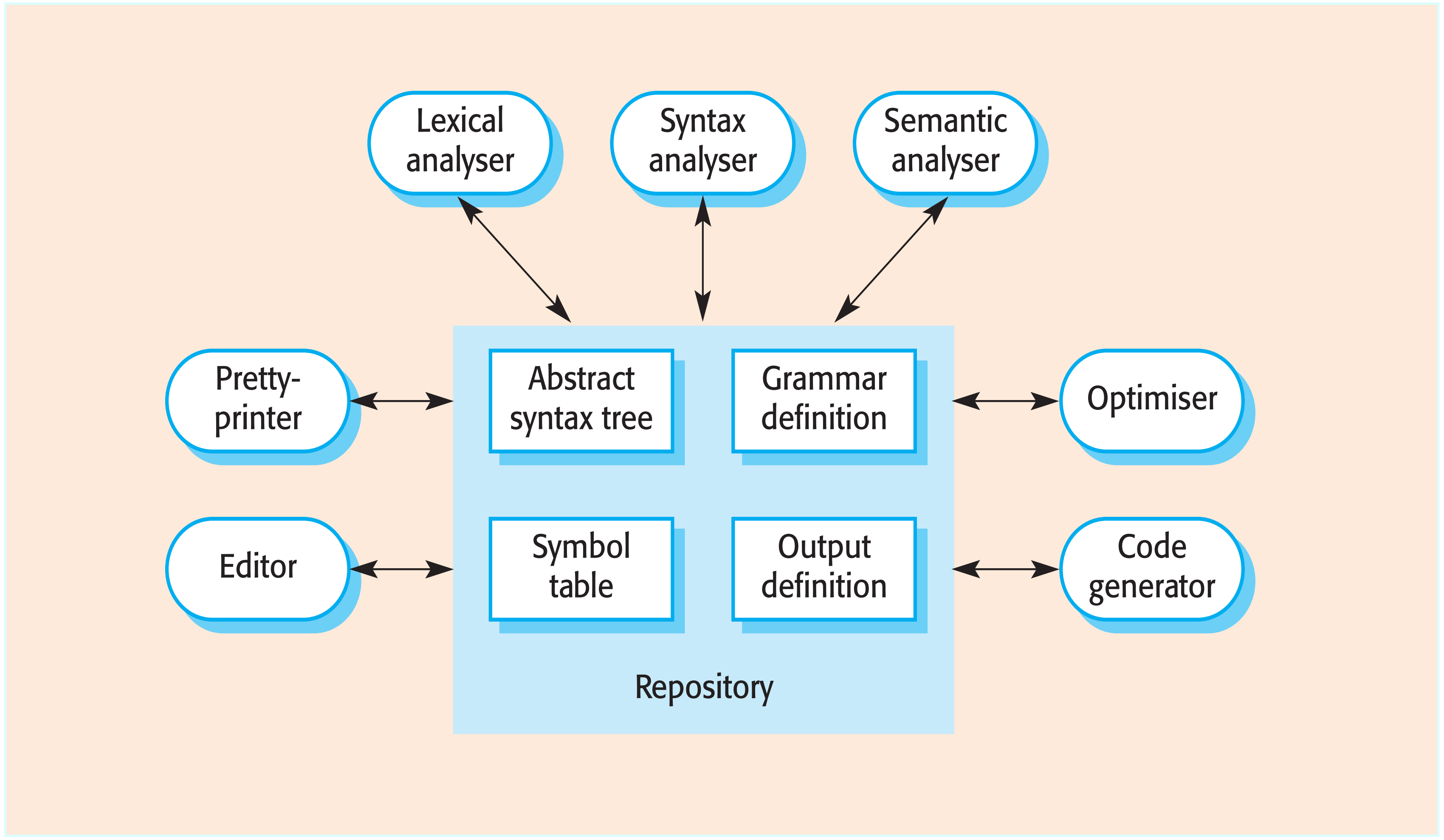
4. Language Processing System

- Accept a natural or artificial language as input and generate some other representation of that language.
- May include an interpreter to act on the instructions in the language that is being processed.

- Components of language processing systems
  - Lexical analyser
  - Symbol table
  - Syntax analyser
  - Syntax tree
  - Semantic analyser
  - Code generator



Repository Model of a Compiler



Summary

- Generic models of application architectures help us understand and compare applications.
- Important classes of application are data processing systems, transaction processing systems, event processing systems and language processing system.
- Data processing systems operate in batch mode and have an input-process-output structure.
- Transaction processing systems allow information in a database to be remotely accessed and modified by multiple users.
- Event processing systems include editors and real-time systems.
- In an editor, user interface events are detected and an in-store data structure is modified.
- Language processing systems translate texts from one language to another and may interpret the specified instructions.

Chapter 14.
Object-Oriented Design

Objectives

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

Object-Oriented Development

- Object-oriented analysis, design and programming are related but distinct.
  - OOA : concerned with developing an object model of the application domain.
  - OOD : concerned with developing an object-oriented system model to implement requirements.
  - OOP : concerned with realising an OOD using an OO programming language such as Java or C++.
- Characteristics of OOD
  - Objects are abstractions of real-world or system entities.
  - Objects are independent and encapsulate state and representation information.
  - System functionality is expressed in terms of object services.
  - Shared data areas are eliminated. Objects communicate by message passing.
  - Objects may be distributed and may execute sequentially or in parallel.

Advantages of OOD

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

Objects and Object Classes

- Objects are entities in software system, which represent instances of real-world and system entities.
- Object classes are templates for objects, which used to create objects.
- Object classes may inherit attributes and services from other object classes.

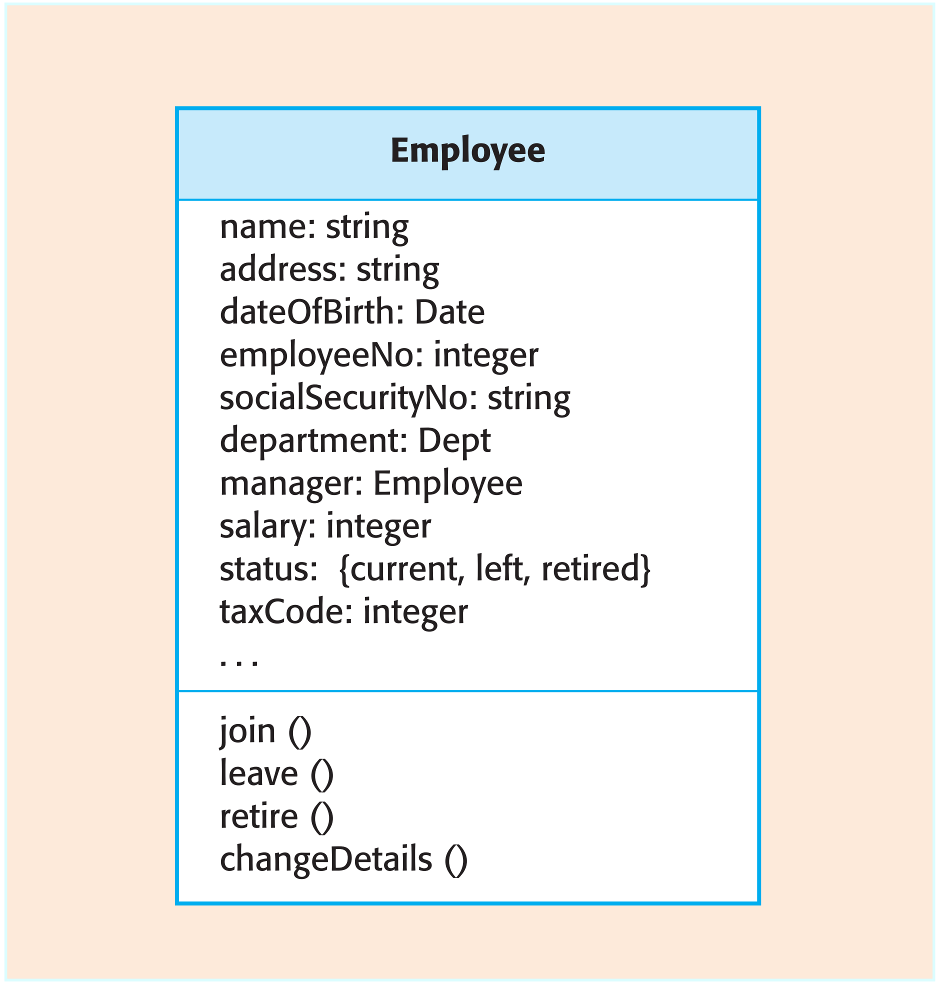
*An **object** is an entity that has a state and a defined set of operations which operate on that state. The state is represented as a set of object attributes. The operations associated with the object provide services to other objects (clients) which request these services when some computation is required.*

*Objects are created according to some **object class** definition. An object class definition serves as a template for objects. It includes declarations of all the attributes and services which should be associated with an object of that class.*

The Unified Modeling Language

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

Employee Object Class (UML)



Object Communication

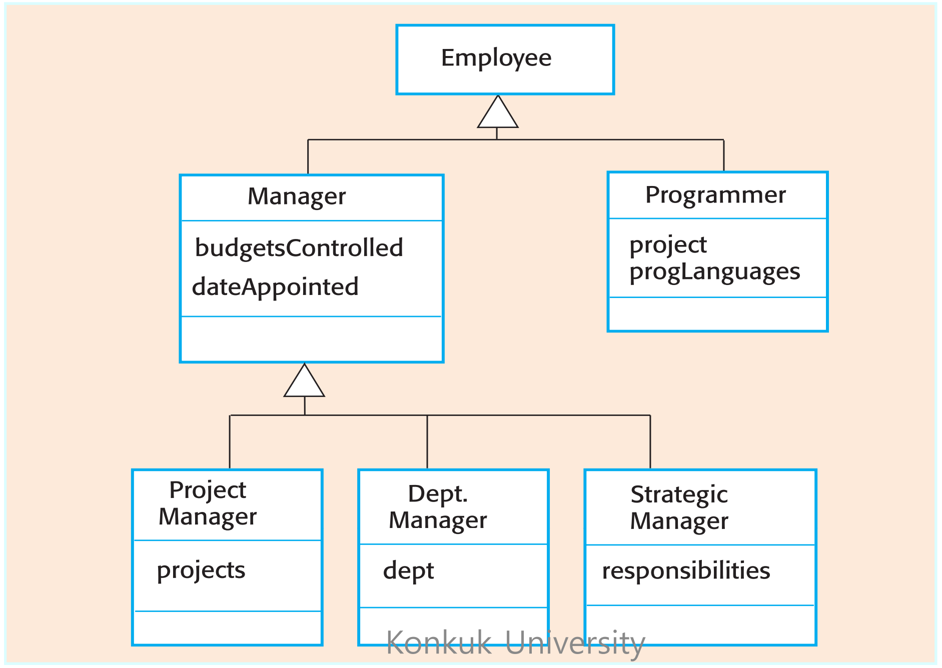
- Conceptually, objects communicate by message passing.
- Messages
  - Name of service requested by calling object
  - Copies of information required to execute the service
- In practice, messages are often implemented by procedure calls
  - Name = procedure name
  - Information = parameter list

```
// 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) ;
```

Generalization and Inheritance

- 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).
- A sub-class inherits the attributes and operations from its super class and may add new methods or attributes of its own.
- Generalisation in the UML is implemented as an inheritance in OO programming languages.



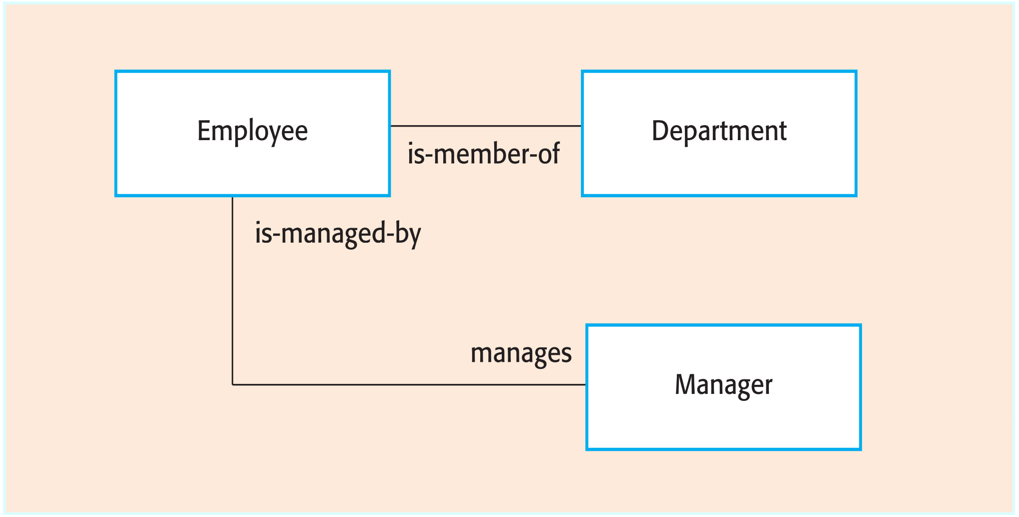
Features of Inheritance

- Advantages:
  - It is an abstraction mechanism which may be used to classify entities.
  - It is a reuse mechanism at both the design and the programming level.
  - Inheritance graph is a source of organisational knowledge about domains and systems.

- Problems:
  - Object classes are not self-contained. They cannot be understood without reference to their super-classes.
  - Designers have a tendency to reuse the inheritance graph created during analysis. Can lead to significant inefficiency.
  - Inheritance graphs of analysis, design and implementation have different functions and should be separately maintained.

UML Association

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



Concurrent Object

- The nature of objects as self-contained entities make them suitable for concurrent implementation.
- The message-passing model of object communication can be implemented directly if objects are running on separate processors in a distributed system.

- Servers
  - The object is implemented as a parallel process (server) with entry points corresponding to object operations.
  - If no calls are made to it, the object suspends itself and waits for further requests for service.
- Active objects
  - Objects are implemented as parallel processes and the internal object state may be changed by the object itself and not simply by external calls.
  - Thread in Java is a simple construct for implementing concurrent objects.

Java Thread

- Thread in Java is a simple construct for implementing concurrent objects.
- Threads must include a method called run() and this is started up by the Java run-time system.
- Active objects typically include an infinite loop so that they are always carrying out the computation.

Object-Oriented Design Process

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

- <u>Common key activities</u> for OOD processes
  1. Define the context and modes of use of the system
  2. Design the system architecture
  3. Identify the principal system objects
  4. Develop design models
  5. Specify object interfaces

Example: Weather system description

A weather mapping system is required to generate weather maps on a regular basis using data collected from remote, unattended weather stations and other data sources such as weather observers, balloons and satellites. Weather stations transmit their data to the area computer in response to a request from that machine.

The area computer system validates the collected data and integrates it with the data from different sources. The integrated data is archived and, using data from this archive and a digitised map database a set of local weather maps is created. Maps may be printed for distribution on a special-purpose map printer or may be displayed in a number of different formats.

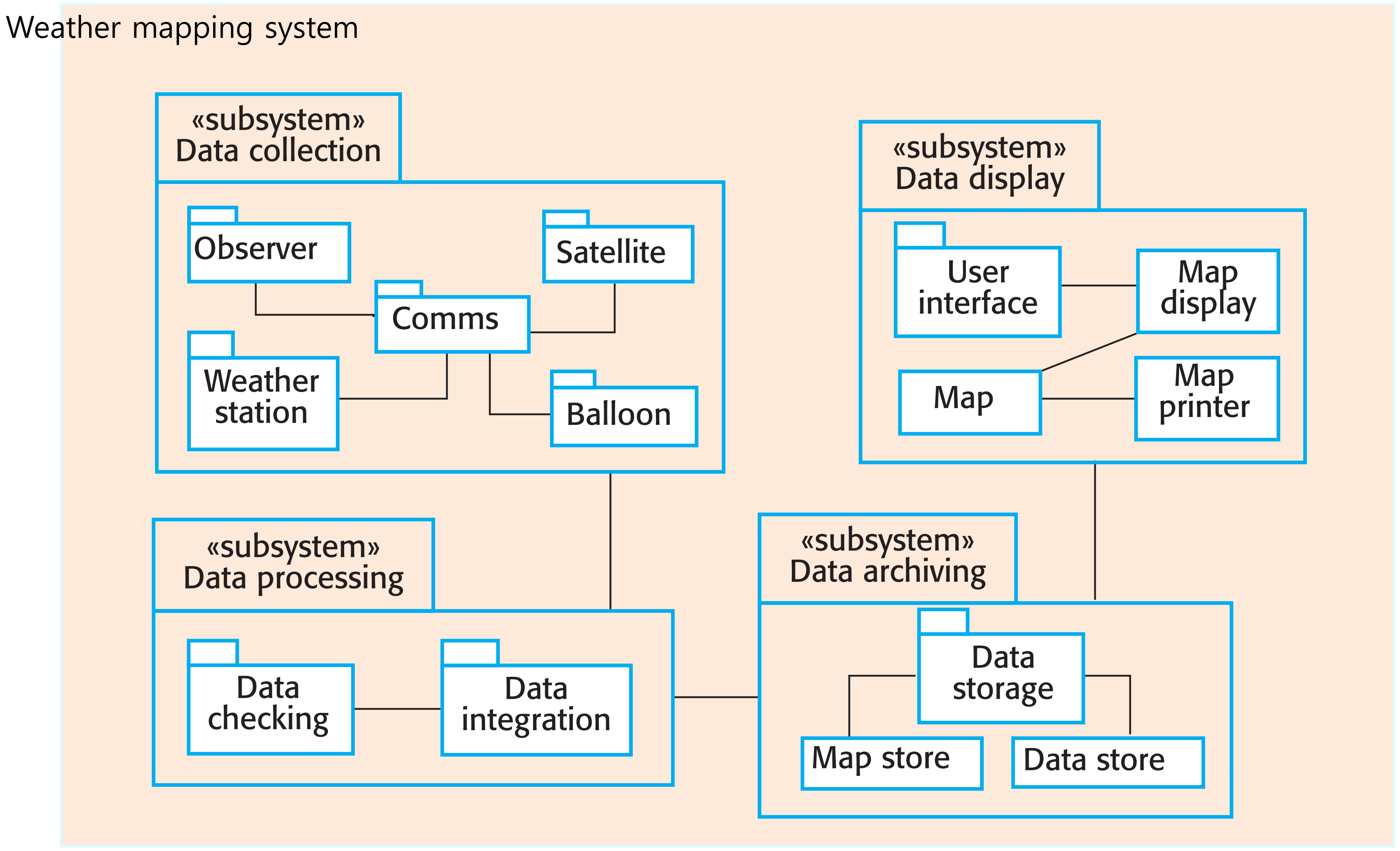
1. System Context and Models of System Use

- Develop an understanding of the relationships between the software being designed and its external environment

- System context
  - A static model that describes other systems in the environment.
  - Use a <u>subsystem model</u> to show other systems.

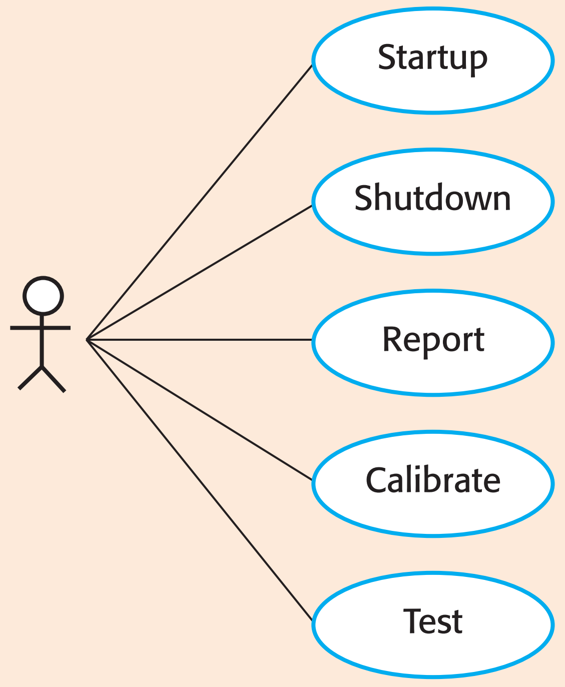
- Model of system use
  - A dynamic model that describes how the system interacts with its environment.
  - Use <u>use-cases</u> to show interactions

Subsystem Model



Use-Case model

Weather station use-case

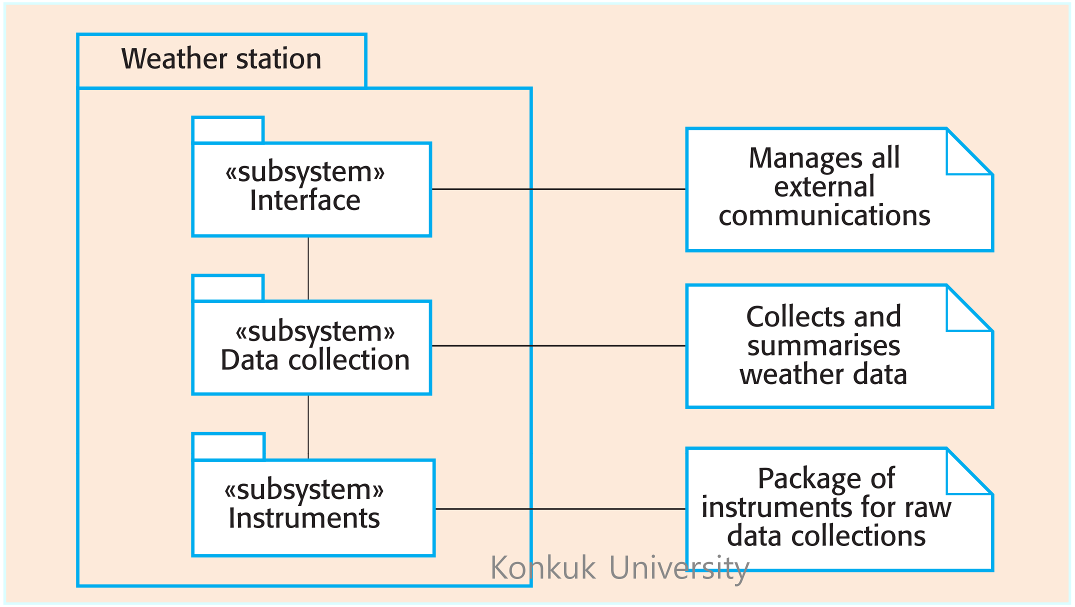


Use-case description

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.

2. Architectural Design

- Once interactions between the system and its environment have been understood, you use this information for designing the system architecture.
- A layered architecture is appropriate for the weather station
  - Interface layer for handling communications
  - Data collection layer for managing instruments
  - Instruments layer for collecting data



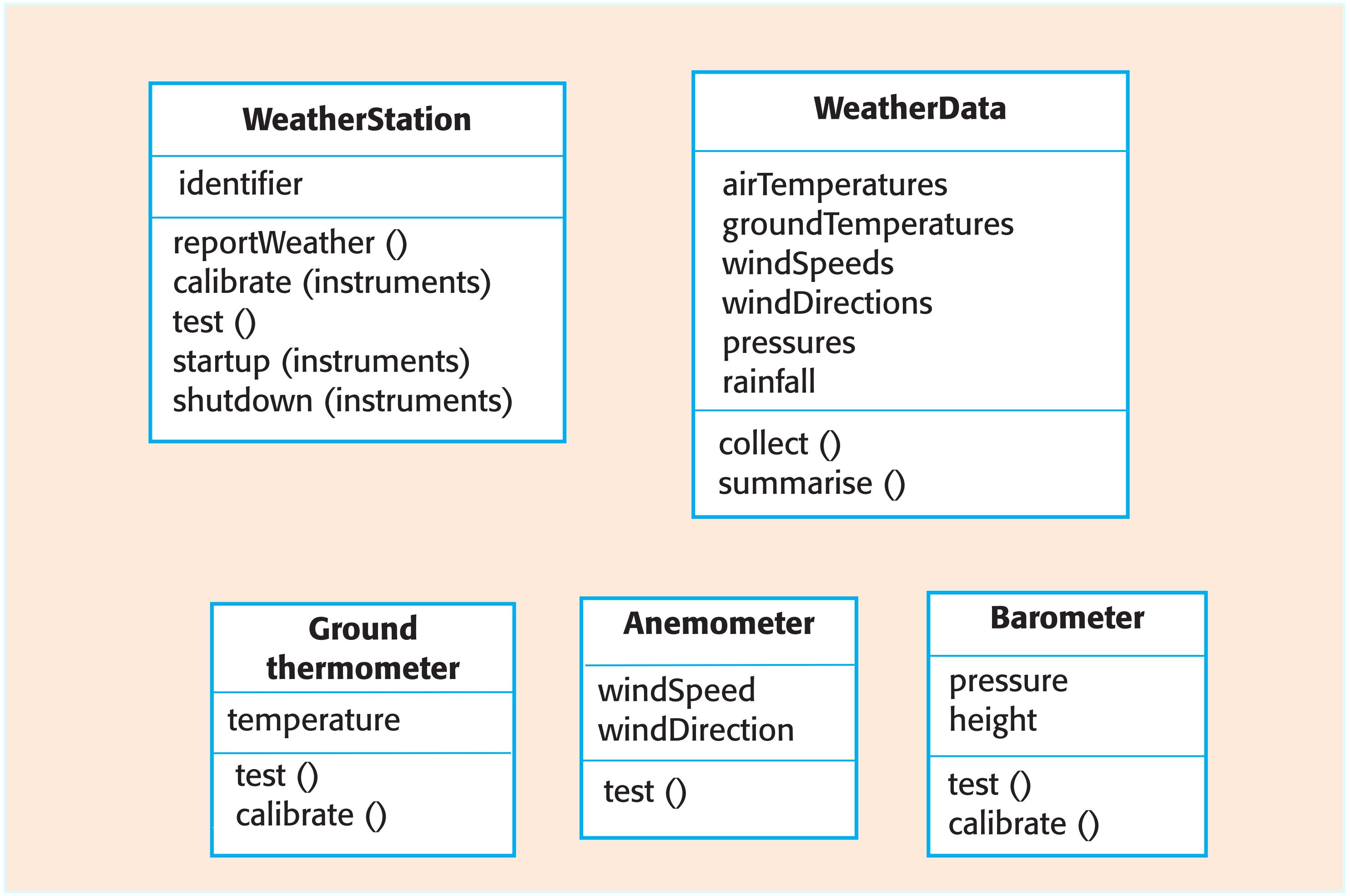
Weather station
«subsystem» Interface
«subsystem» Data collection
«subsystem» Instruments
Manages all external communications
Collects and summarises weather data
Package of instruments for raw data collections

3. Object Identification

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

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

Weather Station Object Classes



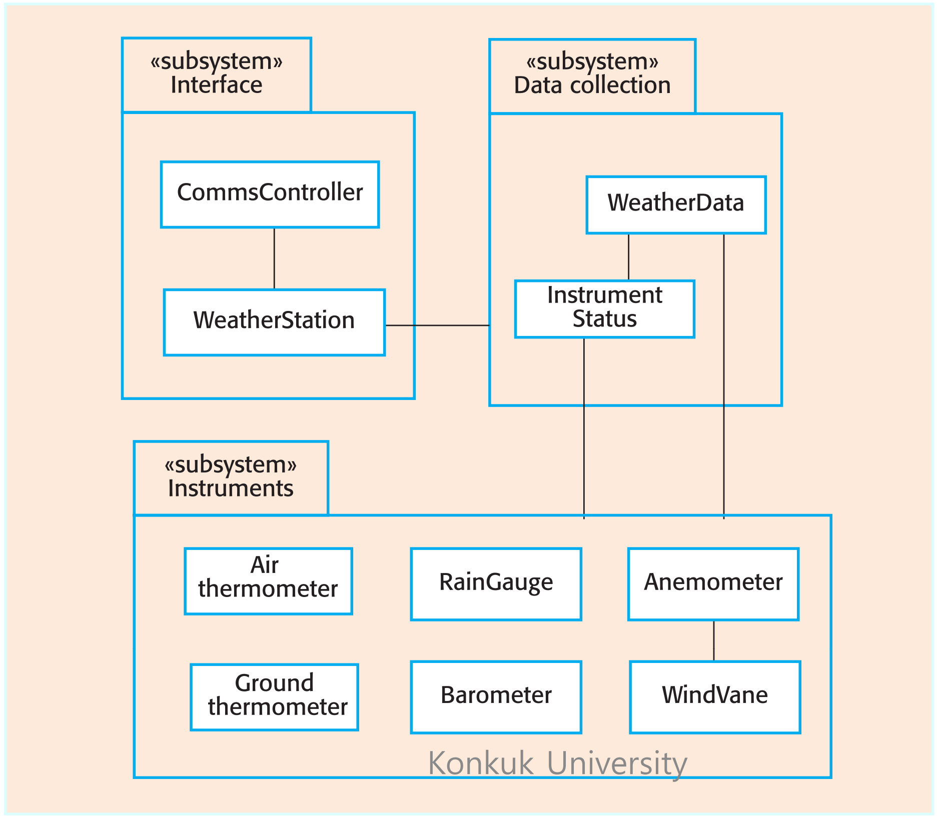
4. Developing Design Model

- Design models show the objects and object classes and relationships between these entities.
  - Static models describe the static structure of the system in terms of object classes and relationships.
  - Dynamic models describe the dynamic interactions between objects.

- Examples of design models:
  - Sub-system model : shows logical groupings of objects into coherent subsystems.
  - Sequence model : shows the sequence of object interactions.
  - State machine model : show how individual objects change their state in response to events.
  - Other models include use-case models, aggregation models, generalisation models, etc.

Subsystem Model

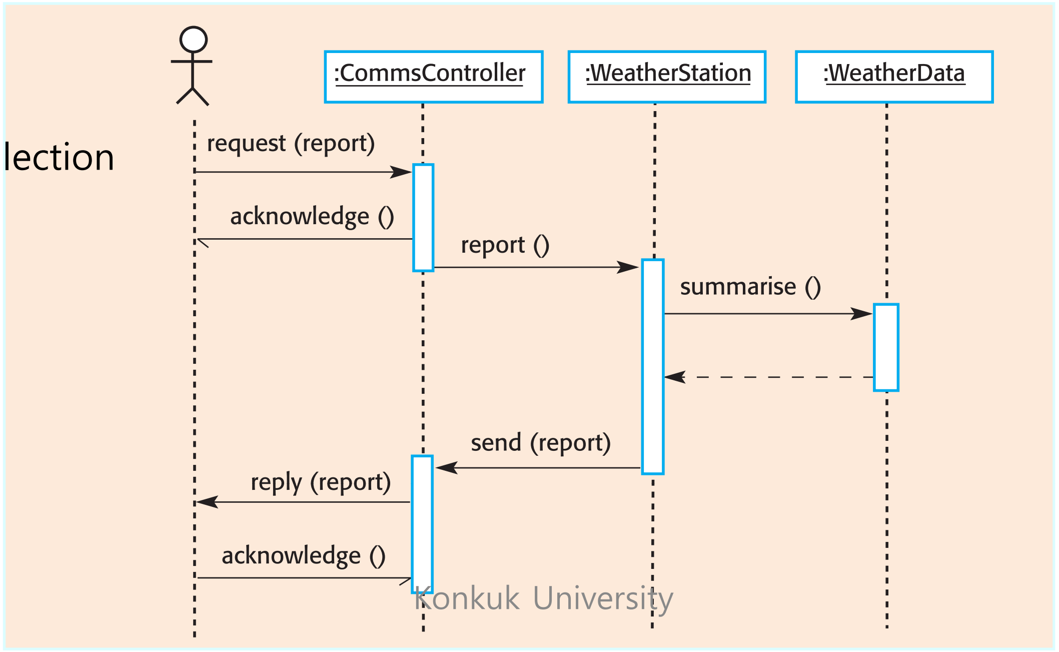
- Shows how the design is organised into logically related groups of objects. In the UML, these are shown using packages.
- This is a logical model. The actual organisation of objects in the system may be different.



Sequence Model

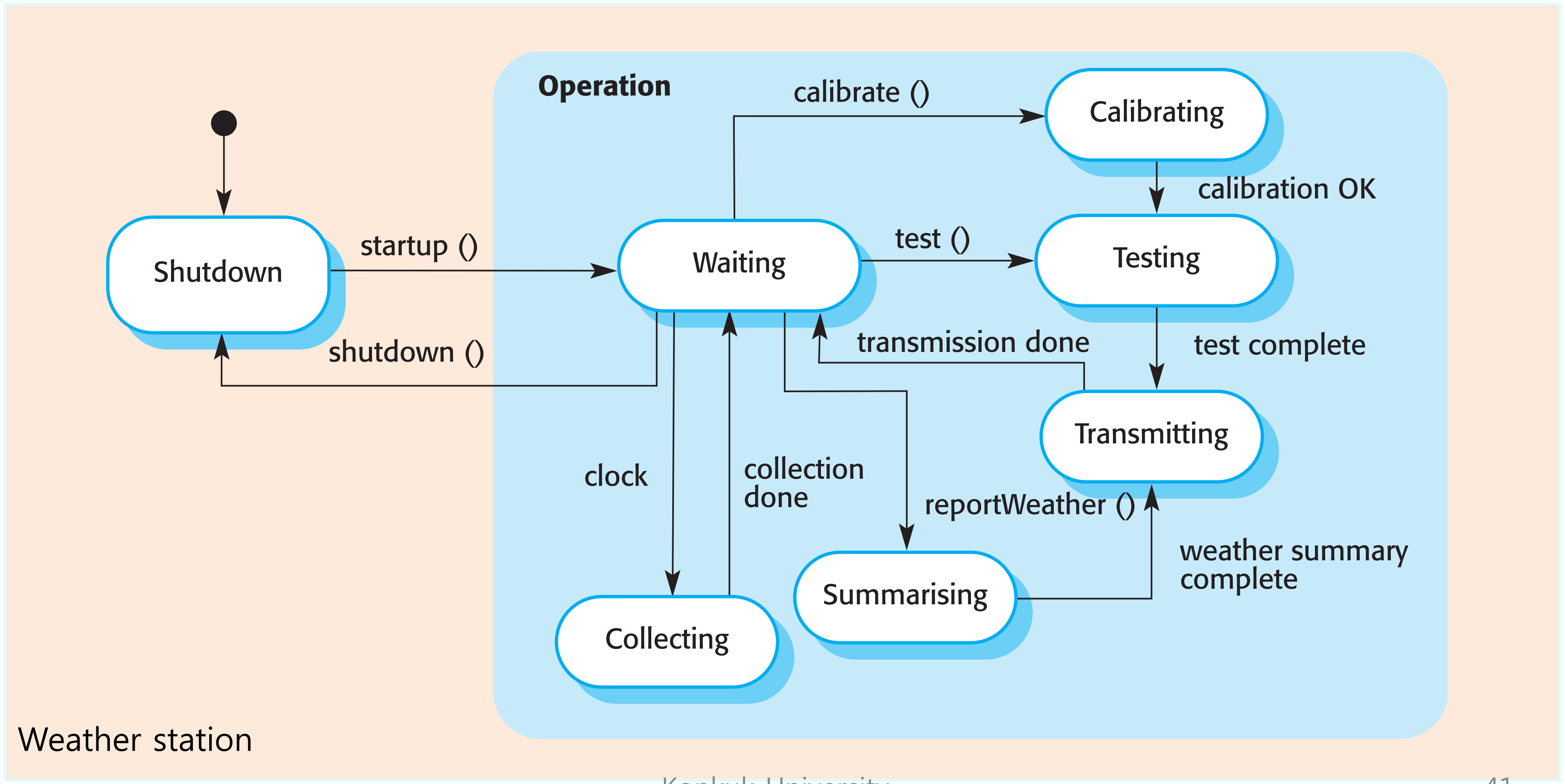
- Sequence models show the sequence of object interactions that take place
  - Objects are arranged horizontally across the top.
  - Time is represented vertically so models are read top to bottom.
  - Interactions are represented by labelled arrows, Different styles of arrow represent different types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

Data collection



State Machine Model : Statecharts

- Show how objects respond to different service requests and the state transitions triggered by these requests.



5. Object Interface Specification

- Object interfaces have to be specified so that the objects and other components can be designed in parallel.
- Objects may have several interfaces which are viewpoints on the methods provided.
- The UML uses class diagrams for interface specification.

```
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
```

Summary

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

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