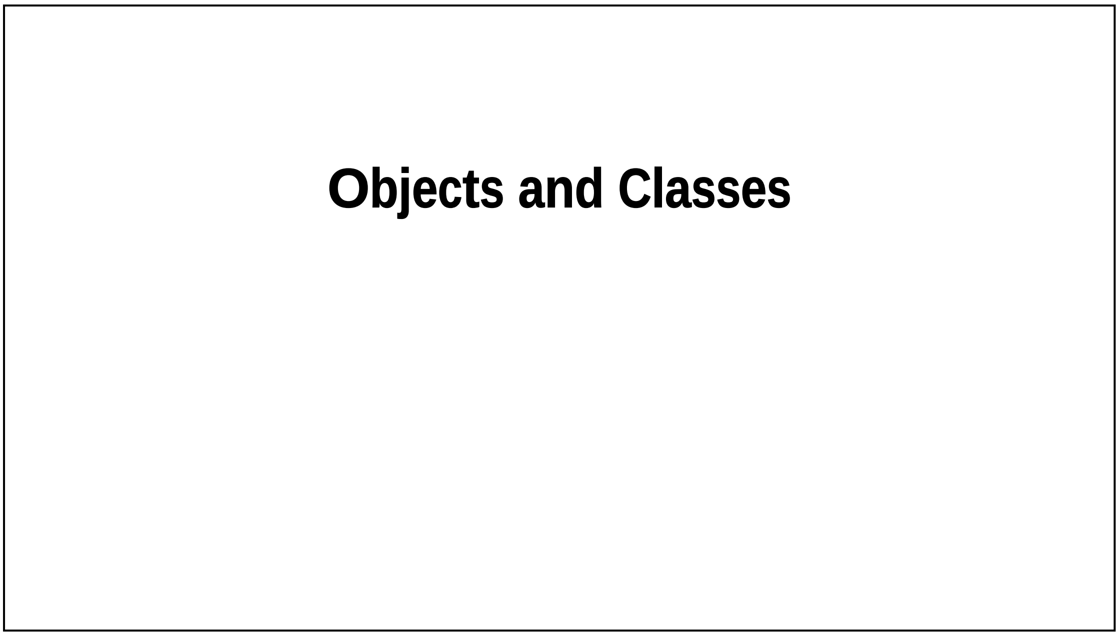
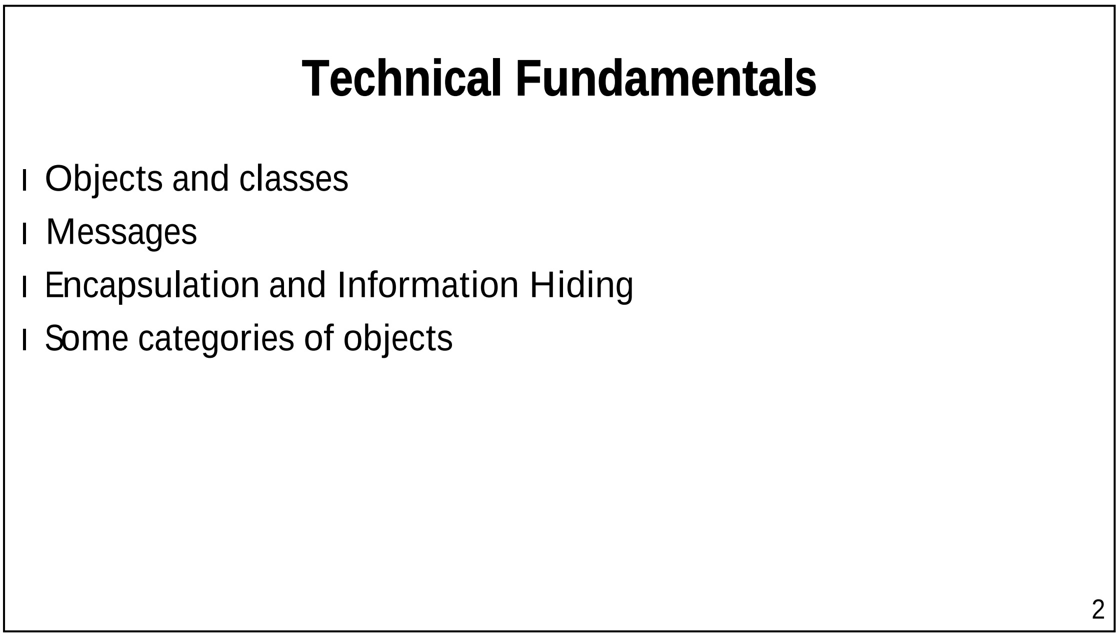


Objects and Classes



Technical Fundamentals

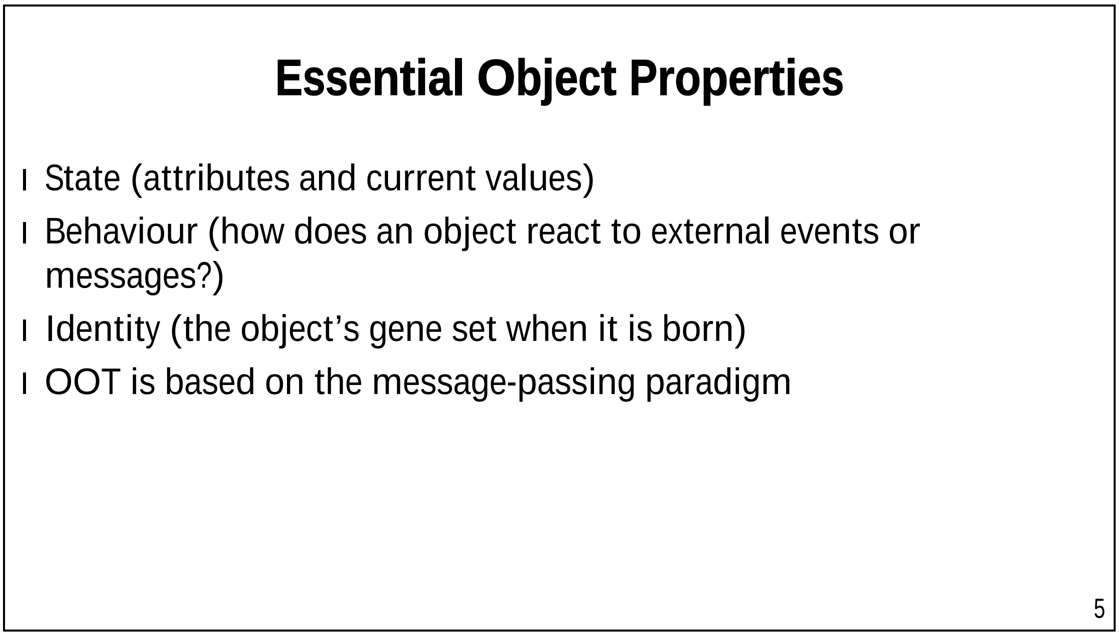
- Objects and classes
- Messages
- Encapsulation and Information Hiding
- Some categories of objects

Objects and Classes

- An object is usually tangible and has sharp boundaries
- Can be visible or their presence can be felt
- No two objects are the same (‘an object is born with its own unique identity’)
- It is important to define the context in which an object is to operate

Examples of Objects

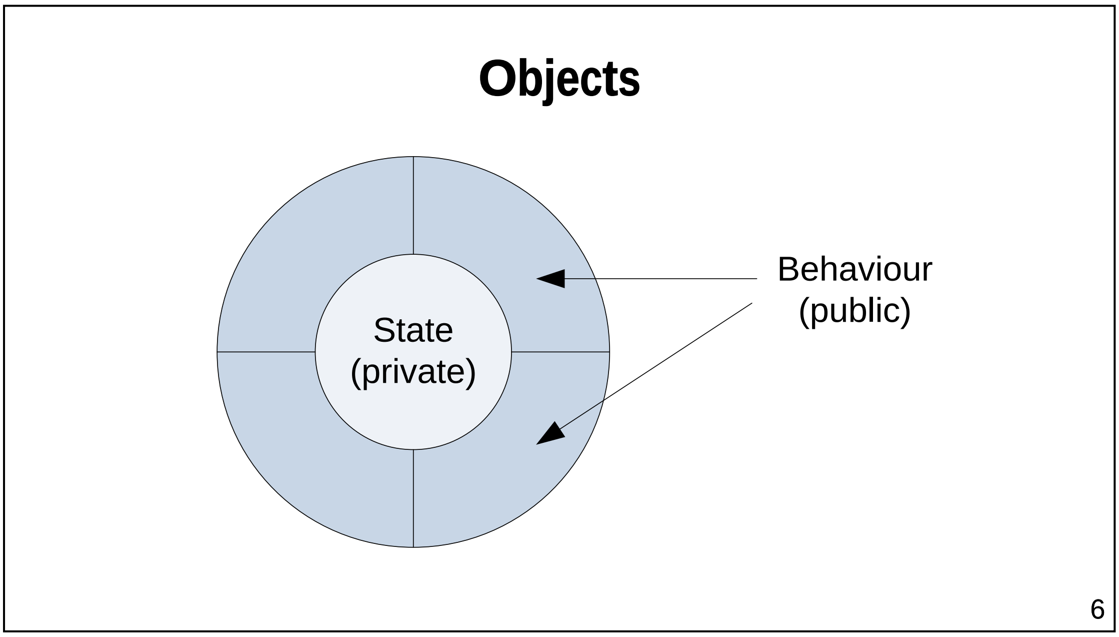
- Furnace number 1
- Account with account number 548274649
- The date 2000/1/1
- Weekend schedule



Essential Object Properties

- State (attributes and current values)
- Behaviour (how does an object react to external events or messages?)
- Identity (the object’s gene set when it is born)
- OOT is based on the message-passing paradigm

5



Objects

6

Classes

- A class is an abstraction (it corresponds to a set of objects)
- Synonyms: object factory, object template
- An object is called an instance of a class
- A class has structure (private) and an interface (public)
- All instances of a class have the same interface

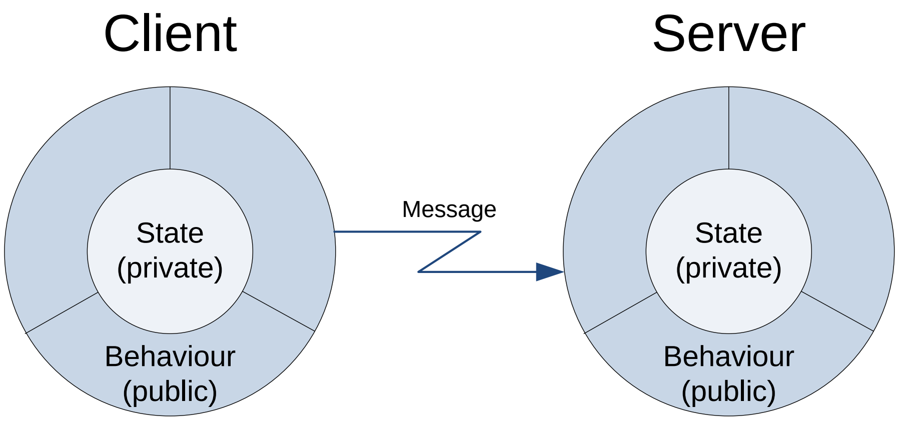
Messages (1/2)

- Objects communicate by sending messages (events)
- Possible to send messages to classes
- Constructor messages create objects
- Destructor messages destroy objects

Messages (2/2)

- Selector messages do not change object state (read operations)
- Modifier messages modify object state (write operations)

Messages and Objects



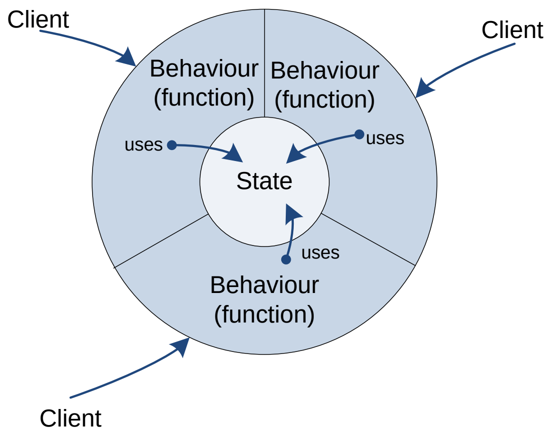
Encapsulation and Information Hiding (1/2)

- View each object as a black box
- Cannot access state directly; must use public interface
- Data and operations are tightly coupled
- Client requests cannot destroy integrity of object

Encapsulation and Information Hiding (2/2)

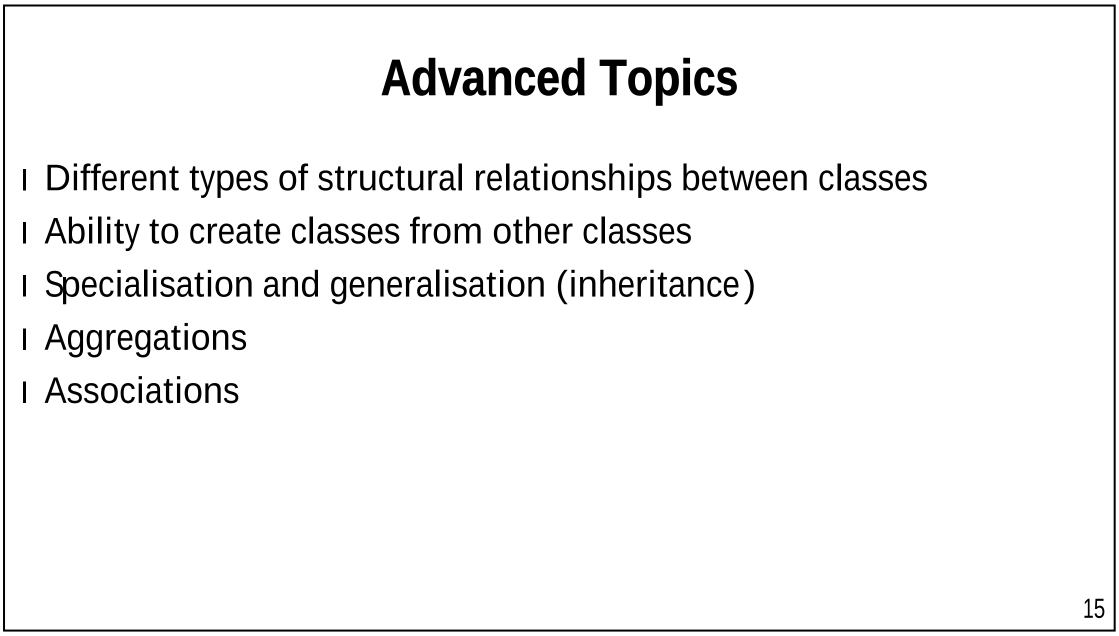
- Information hiding is a more general concept than encapsulation
- These concepts are the most important in OOT

Encapsulation



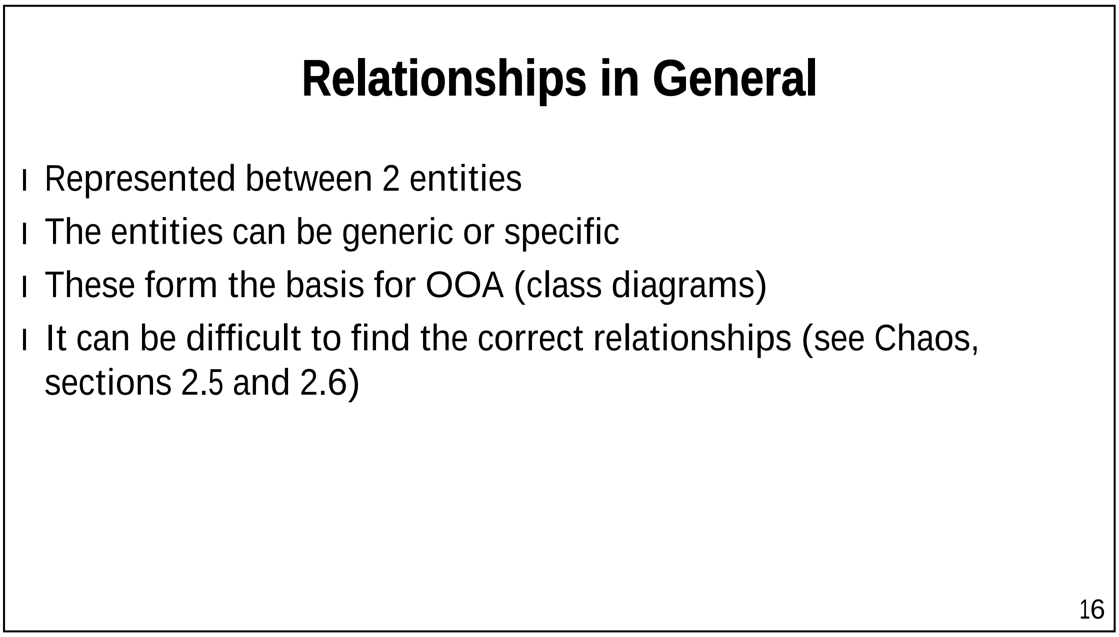
Advantages of using Information Hiding

- Modifications to internal structure occurs locally in a component
- No ‘ripple’ effects in other parts of code due to modifications
- Supported to a large extent by object-orient languages
- Results in flexible systems (interface specifications can easily be modified)



Advanced Topics

- Different types of structural relationships between classes
- Ability to create classes from other classes
- Specialisation and generalisation (inheritance)
- Aggregations
- Associations



Relationships in General

- Represented between 2 entities
- The entities can be generic or specific
- These form the basis for OOA (class diagrams)
- It can be difficult to find the correct relationships (see Chaos, sections 2.5 and 2.6)

Examples

- An employee is a person
- A file consists of records
- A person works for a company

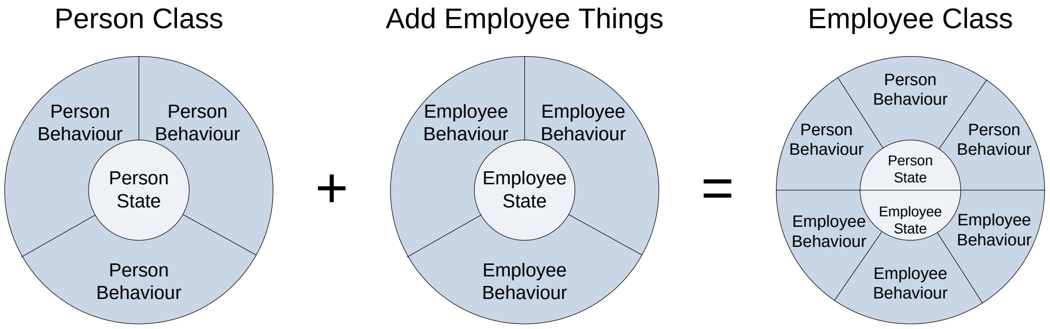
17

Specialisation and Generalisation

- Correspond to ISA and AKO relationships
- One type can be a specialisation or generalisation of another type
- These relationships map to inheritance mechanisms in OO languages

18

Specialisation



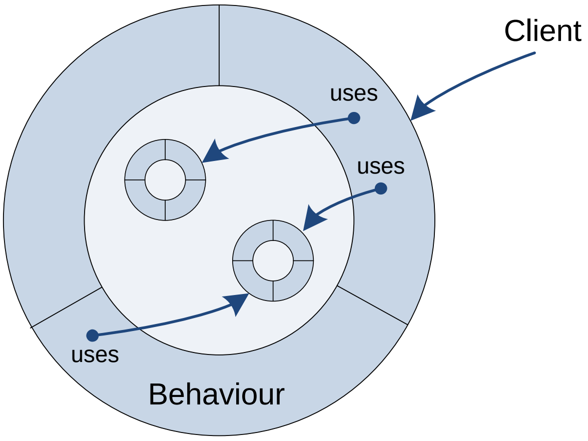
Aggregations (1/2)

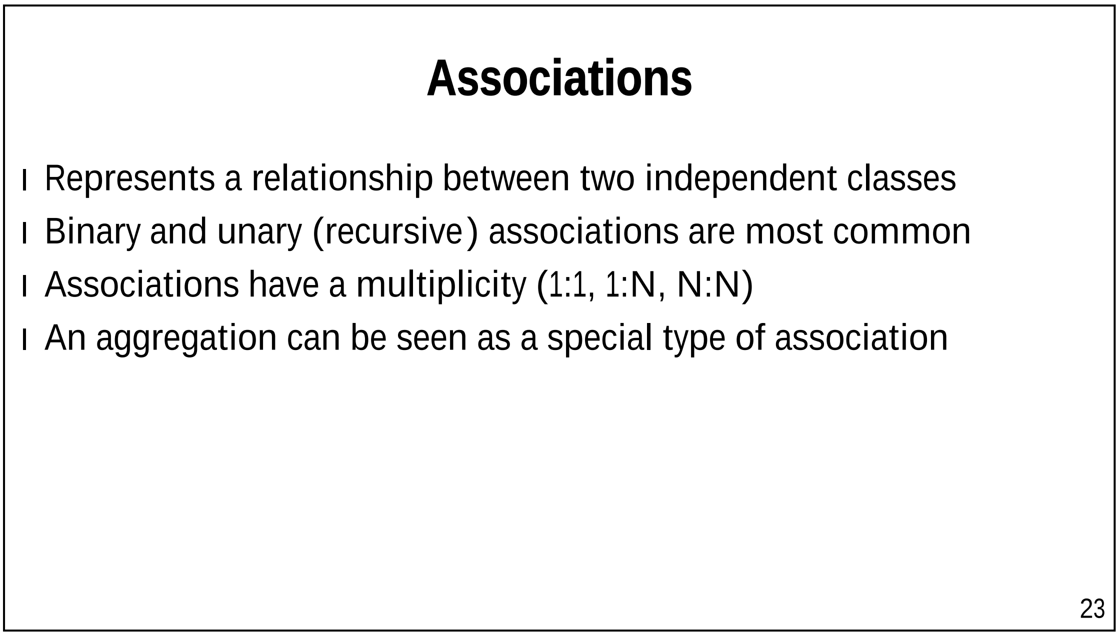
- Correspond to ‘whole-part’ relationship
- Such relationships occur in many types of applications
- We speak of a master object and its components
- The components may or may not be related

Aggregations (2/2)

- Master object delegates to its components
- Access to a given component must go via the master

Aggregation





Associations

- Represents a relationship between two independent classes
- Binary and unary (recursive) associations are most common
- Associations have a multiplicity (1:1, 1:N, N:N)
- An aggregation can be seen as a special type of association