

Object-oriented Design

Objectives

- To explain how a software design may be represented as a set of interacting objects that manage their own state and operations
- To describe the activities in the 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

Topics covered

- Objects and object classes
- An object-oriented design process
- Design evolution

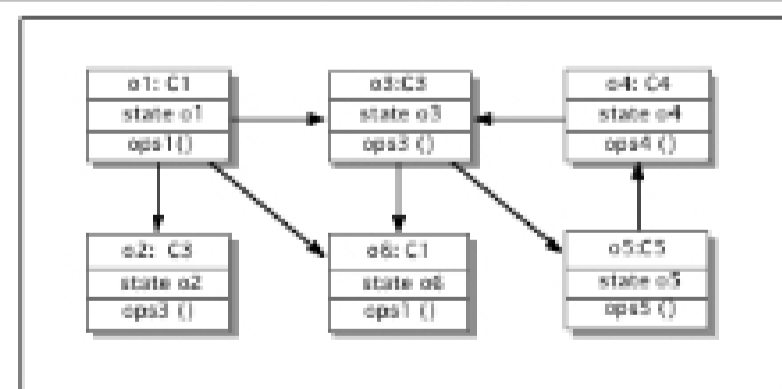
Object-oriented development

- Object-oriented analysis, design and programming are related but distinct.
- OOA is concerned with developing an object model of the application domain.
- OOD is concerned with developing an object-oriented system model to implement requirements.
- OOP is 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 and manage themselves.
- 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.

Interacting objects



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

Objects and 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
 - 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.
- In practice, messages are often implemented by procedure calls
 - Name = procedure name;
 - Information = parameter list.

Message examples

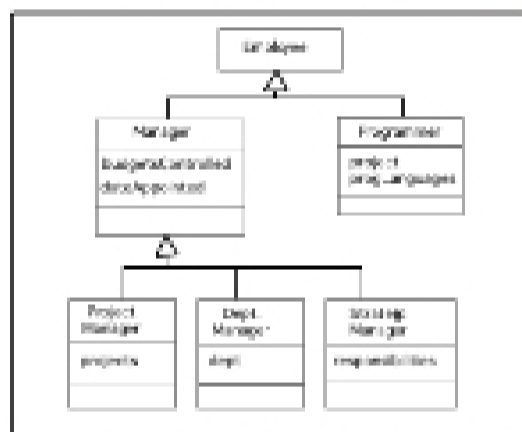
```
// Call a method associated with a buffer
// object that returns the next value
// in the buffer
    v = circularBuffer.Get () ;

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

Generalisation and inheritance

- Objects are members of classes that define attribute types and operations.
- 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 inheritance in OO programming languages.

A generalisation hierarchy



Advantages of inheritance

- 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.
- The inheritance graph is a source of organisational knowledge about domains and systems.

Problems with inheritance

- 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.
- The inheritance graphs of analysis, design and implementation have different functions and should be separately maintained.

UML associations

- 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 are general but may indicate that an attribute of an object is an associated object or that a method relies on an associated object.