

OBJECT-ORIENTED MODELLING (OOM) - 32536

MODULE 2

**Fundamentals of Object Orientation;
The Unified Modeling Language
(UML); Organizing your project
using Packages**

*(POOA – Chapter 1 & 2; POOD – Chapter 1) -
(POOA-Chapter 3)*

"Copyright Practical OO Analysis & Practical OO Design, Thomson Publishing, 2005-8" 1

Module Outline

- Understanding object-orientation
- Objects and classes
- Differences between procedural and object-oriented approaches
- Fundamentals of object-orientation: Classification, Abstraction, Inheritance, Encapsulation and Polymorphism
- Purpose of Modelling
- The three modelling spaces: Model Of Problem Space, Model Of Solution Space and Model Of Background Space.
- Advantages of modelling with object-orientation: Unified Modeling Language
- Mapping the UML to the Modelling Spaces

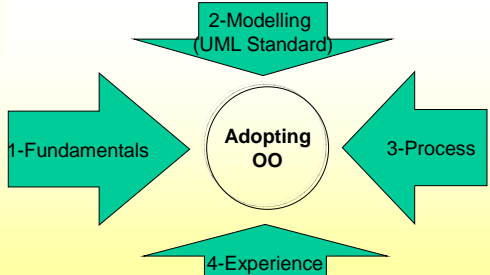
"Copyright Practical OO Analysis & Practical OO Design, Thomson Publishing, 2005-8" 2

Why Object-oriented?

- OO Software Engineering results in easier maintainability and extensibility
 - Due to Localized and controlled changes
- OO Software Engineering results in more Opportunities for Reuse
 - Code, Design (including Patterns and Frameworks), Requirements (Use cases), Components (Link time and Run time)

"Copyright Practical OO Analysis & Practical OO Design, Thomson Publishing, 2005-8" 3

Learning and Adopting Object Orientation



"Copyright Practical OO Analysis & Practical OO Design, Thomson Publishing, 2005-8" 4

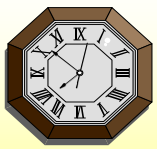
Sub-Module

Fundamentals of Object Orientation

Basic Concepts

"Copyright Practical OO Analysis & Practical OO Design, Thomson Publishing, 2005-8" 5

Classes and Objects



A "CLOCK" in general is a CLASS.
Your CLOCK and My CLOCK are specific Objects

"Copyright Practical OO Analysis & Practical OO Design, Thomson Publishing, 2005-8" 6

What is a Class?

- A Class is a *DEFINITION*, a *TEMPLATE*, for the Objects
- Objects are *INSTANCES* of a Class
 - NOTE: A Class is *NOT* a Collection of Objects;
 - Example: A 'class' of OOA students sitting together in a lecture room are *NOT* a Class in the object-oriented sense

The Star of OO Fundamentals



Classification



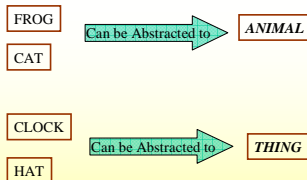
Abstraction: 1st Level is from Objects to Class

Examples

a Frog → FROG
 a Hat → HAT
 a Clock → CLOCK
 a Cat → CAT

Note that these are *ABSTRACTIONS* and not real Objects. Objects are Classified and good Classification leads to creation of good Abstractions. Classes that represent a collection of Objects are Abstract.

Abstraction: 2nd Level is from Classes to Classes



Classes are abstracted to higher level class

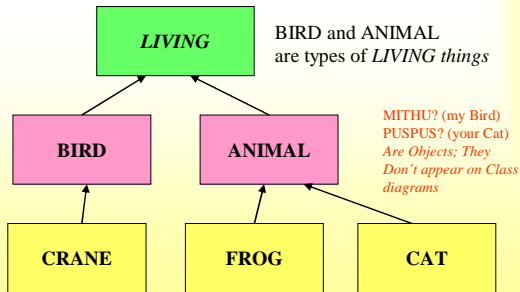
Classification & Abstraction



Requirements from Problem Space can be Classified & Abstracted to arrive at Good Classes

MAN
 BOOK
 CHEST
 SHOE
 HAT

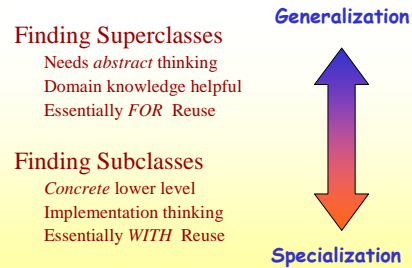
Identifying more general groupings to create Inheritance Hierarchy



Copyright Practical OO Analysis & Practical OO Design, Thomson

13

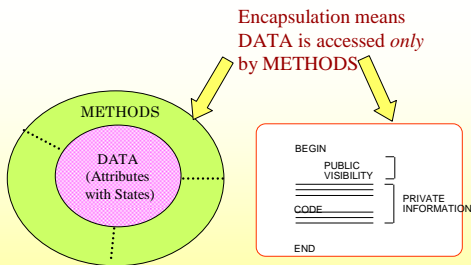
Moving on the Inheritance Tree



Copyright Practical OO Analysis & Practical OO Design, Thomson

14

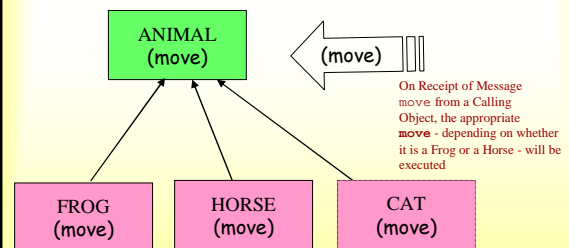
Encapsulation



Copyright Practical OO Analysis & Practical OO Design, Thomson

15

Polymorphism



Advantage? CALLING object need not know what is Moved, so, if a *new* CAT object is added, the CALLING class doesn't change

Copyright Practical OO Analysis & Practical OO Design, Thomson

16

Polymorphism..

- Essentially the ability of an Entity to behave differently - even after receiving the same message
- Translates to the ability of the software to substitute different Objects at runtime
- Implemented via Inheritance in Object-orientation

Copyright Practical OO Analysis & Practical OO Design, Thomson

17

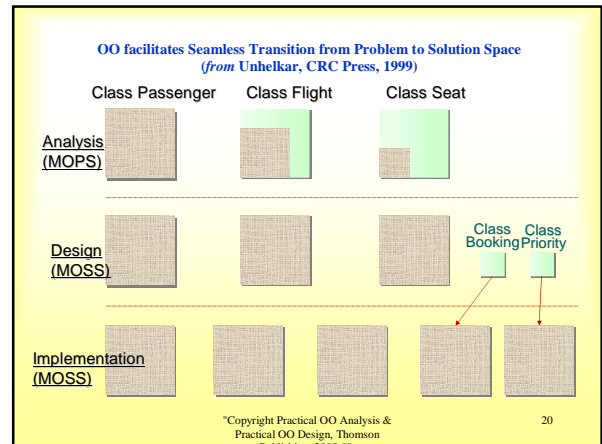
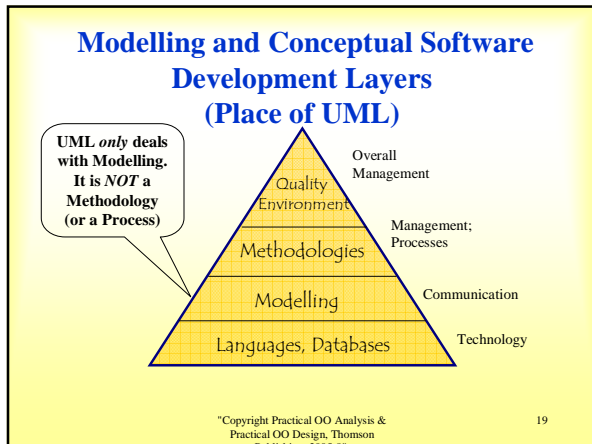
Sub-Module

The Need for Modelling

Software Development Layers

Copyright Practical OO Analysis & Practical OO Design, Thomson

18



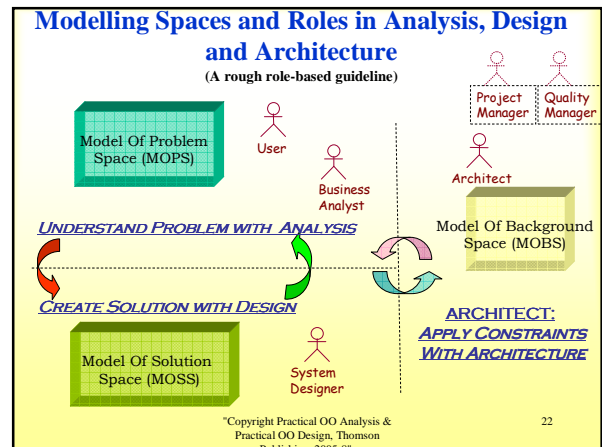
Sub-Module

Modelling Spaces

And Relating them to the UML

Copyright Practical OO Analysis & Practical OO Design, Thomson
2005-06

21



UML 2.0 Diagrams and Purpose

UML diagrams	Represent the...
Use case	functionality from user's viewpoint
Activity	the flow - within a use case or the system
Class	Classes, entities, business domain database
Sequence	the interactions between objects
Interaction Overview	interactions at a general high-level
Communication	the interactions between objects
Object	objects and their links
State Chart	the run-time lifecycle of an object
Composite Structure	Class behavior at run-time
Component	the executables, linkable libraries etc.
Deployment	the hardware nodes and processors
Package	Subsystems, organizational units
Timing	time concept during object interactions

Copyright Practical OO Analysis & Practical OO Design, Thomson
2005-06

23

UML diagrams and Modelling Spaces

UML diagrams	MOPS (Business Analyst)	MOSS (Designer)	MOBS (Architect)
Use case	*****	**	*
Activity	*****	**	*
Class	***	*****	**
Sequence	****	*****	*
Interaction Overview	****	**	**
Communication	*	***	*
Object	*	*****	***
State chart	***	****	**
Composite Structure	*	*****	****
Component	*	***	*****
Deployment	**	**	*****
Package	***	**	****
Timing			*****

Copyright Practical OO Analysis & Practical OO Design, Thomson
2005-06

24

Sub-Module

Package Diagrams

Organizational Technique to Group UML artifacts

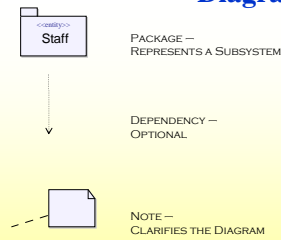
What is a Package?

- A Logical Collection of anything
 - of Classes; of Components;
 - of Use cases etc.
- A Package may map to a Component, but is usually treated differently from a Component
 - a package may not be an executable entity
- Increasingly, a Package is considered a good starting point for Business Analysis
- Treated as a separate diagram in UML 2.0

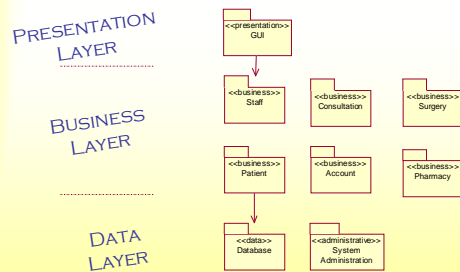
Packages: Tips

- Packages may be the first Diagrams created in the modelling work
- Domain experts can Start with Package Diagrams
- Package Diagrams can have Levels
 - Packages within Packages
- Relationship on Package Diagram during Analysis is not mandatory

Major Ingredients of a Package Diagram



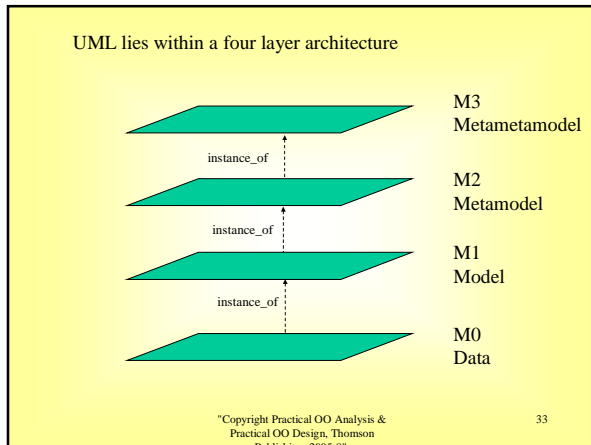
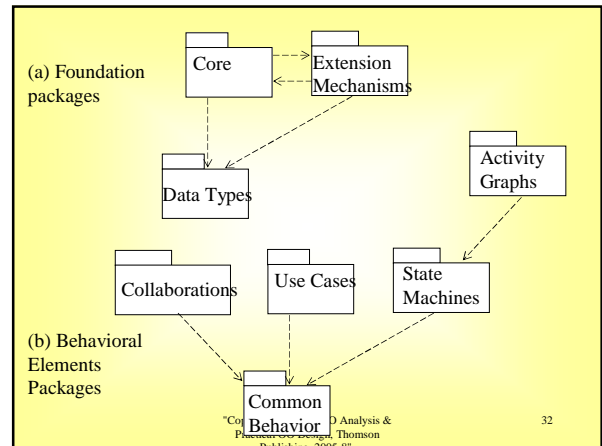
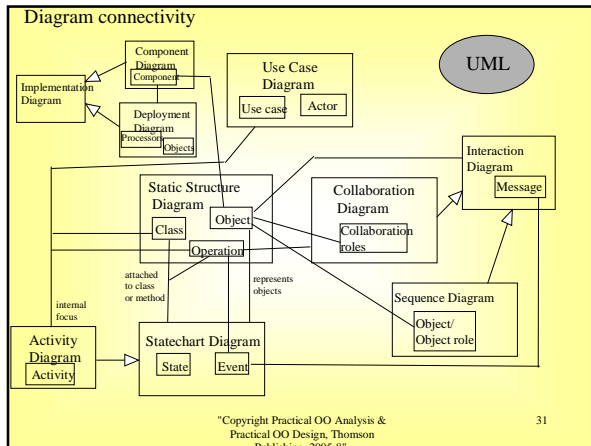
Hospital Management System Package Diagram



Sub-Module

UML – Meta Models

and Layers



- ## Conclusions
- Software Development benefits with Process and Modelling; UML is OMG's standard for Modelling;
 - OO has Five Fundamentals
 - ✓ Classification; Abstraction; Inheritance; Encapsulation and Polymorphism
 - ✓ Class is a Template; Object is an Instance;
 - All Modelling Work happens in 3 Spaces:
 - ✓ MOPS; MOSS; and MOBS
 - ✓ Meta-Models
- Copyright Practical OO Analysis & Practical OO Design, Thomson
34

WORKBOOK

Exercises – Based on POOA /
POOD book Chapter 1 & 2
For Classroom

Copyright Practical OO Analysis & Practical OO Design, Thomson
35