

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)*

1

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

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

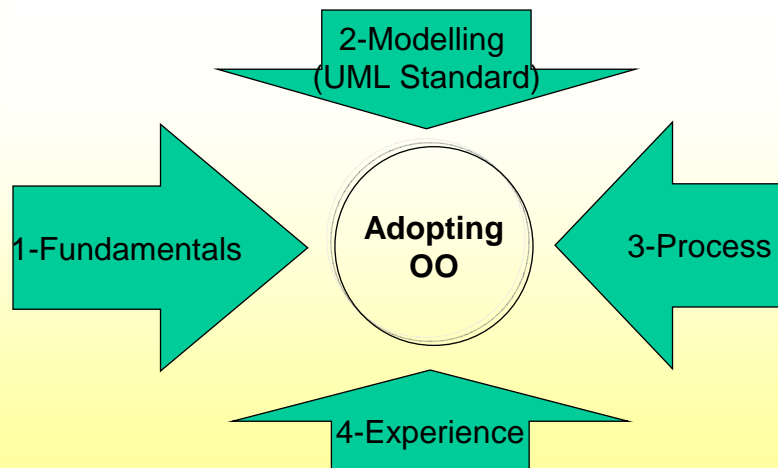
"Copyright Practical OO Analysis &
Practical OO Design, Thomson
Publishing, 2005-8"

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)

Learning and Adopting Object Orientation



Sub-Module

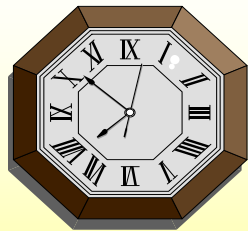
Fundamentals of Object Orientation

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

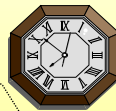
What is a Class?

- 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



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

9

Abstraction: 1st Level is from Objects to Class

Examples

a Frog

FROG

a Hat

HAT

a Clock

CLOCK

a Cat

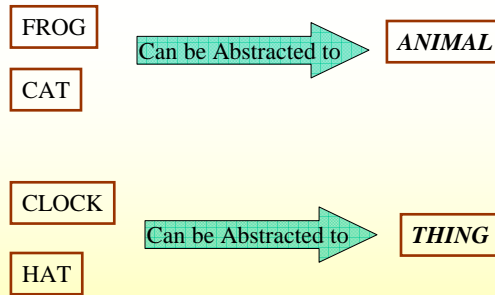
CAT

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.

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

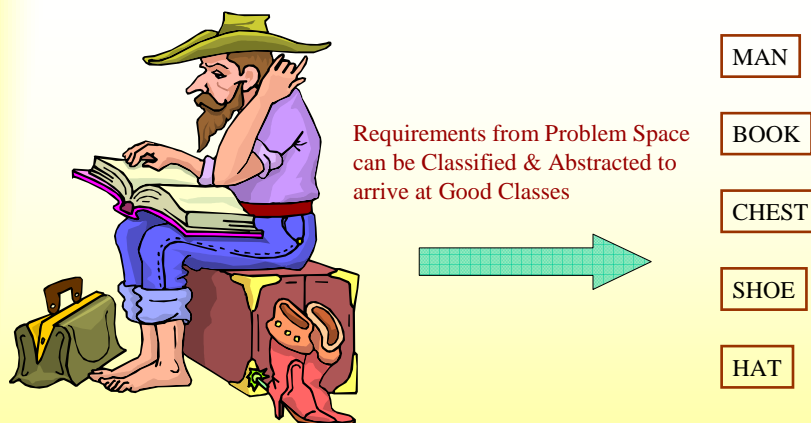
10

Abstraction: 2nd Level is from Classes to Classes

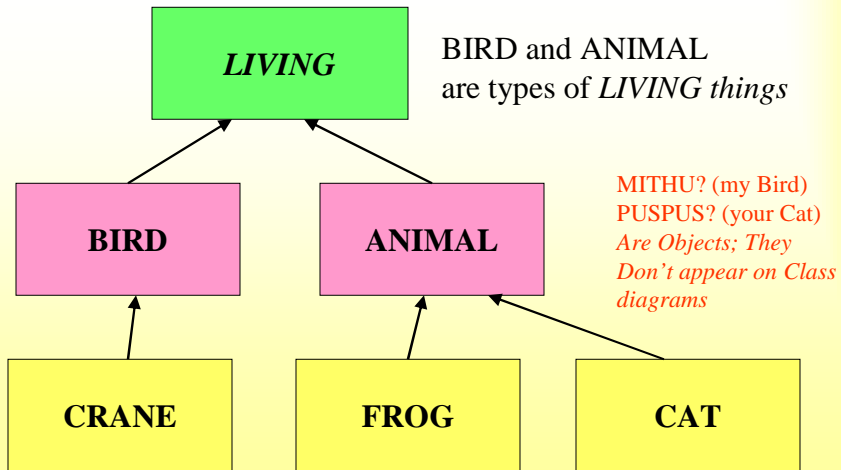


Classes are abstracted to higher level class

Classification & Abstraction



Identifying more general groupings to create Inheritance Hierarchy



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

13

Moving on the Inheritance Tree

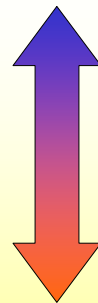
Finding Superclasses

Needs *abstract* thinking
Domain knowledge helpful
Essentially *FOR* Reuse

Finding Subclasses

Concrete lower level
Implementation thinking
Essentially *WITH* Reuse

Generalization



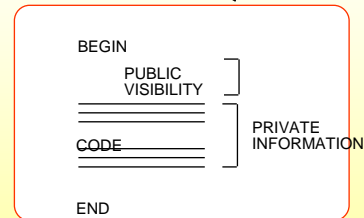
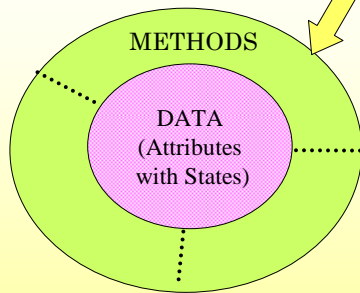
Specialization

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

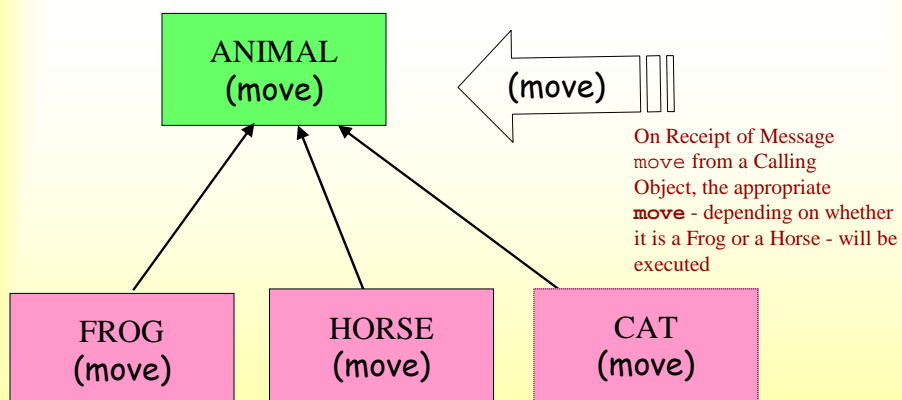
14

Encapsulation

Encapsulation means
DATA is accessed *only*
by METHODS



Polymorphism



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

Polymorphism..

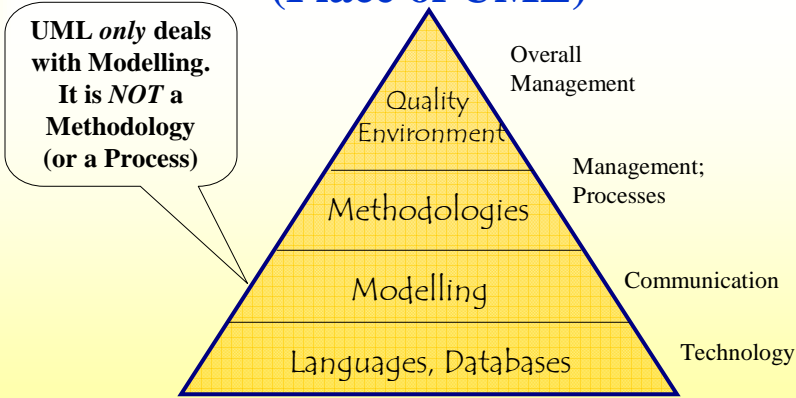
- Essentially the ability of an Entity to behave differently - even after receiving the same message
- Translates to the ability of the software to substitute different Objects at runtime
- Implemented via Inheritance in Object-orientation

Sub-Module

The Need for Modelling

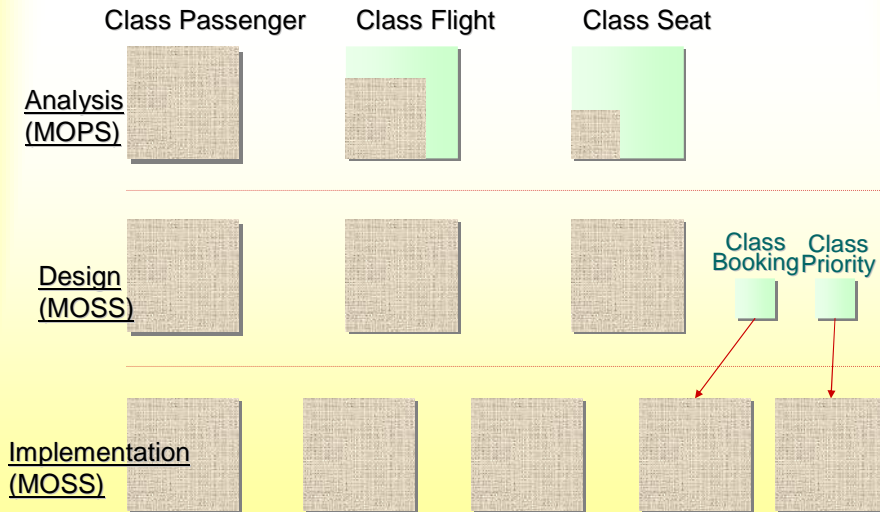
Software Development Layers

Modelling and Conceptual Software Development Layers (Place of UML)



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

OO facilitates Seamless Transition from Problem to Solution Space (from Unhelkar, CRC Press, 1999)



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

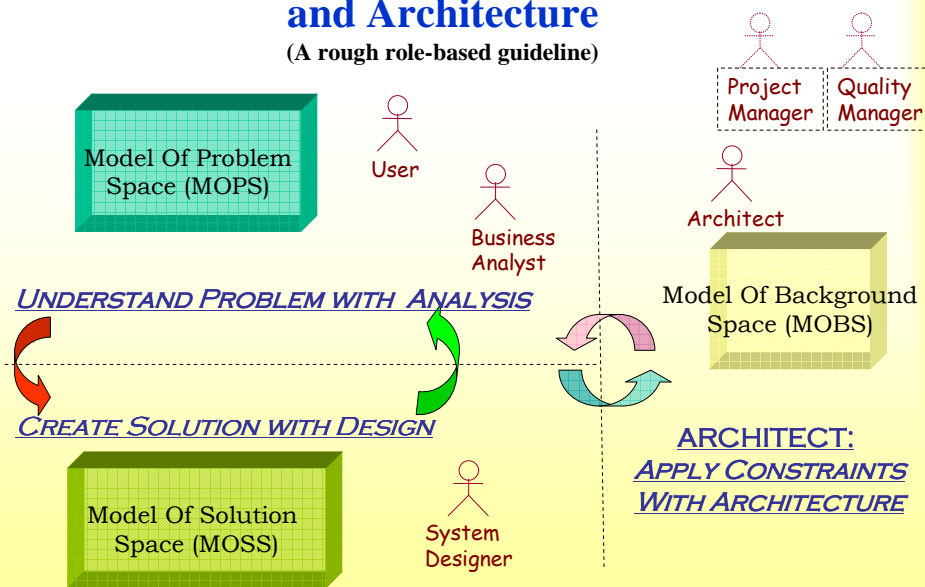
Sub-Module

Modelling Spaces

And Relating them to the UML

Modelling Spaces and Roles in Analysis, Design and Architecture

(A rough role-based guideline)



UML 2.0 Diagrams and Purpose

<i>UML diagrams</i>	<i>Represent the...</i>
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
Publishing, 2005, 8"

23

UML diagrams and Modelling Spaces

<i>UML diagrams</i>	<i>MOPS (Business Analyst)</i>	<i>MOSS (Designer)</i>	<i>MOBS (Architect)</i>
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
Publishing, 2005, 8"

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

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

27

Major Ingredients of a Package Diagram



PACKAGE —
REPRESENTS A SUBSYSTEM



DEPENDENCY —
OPTIONAL

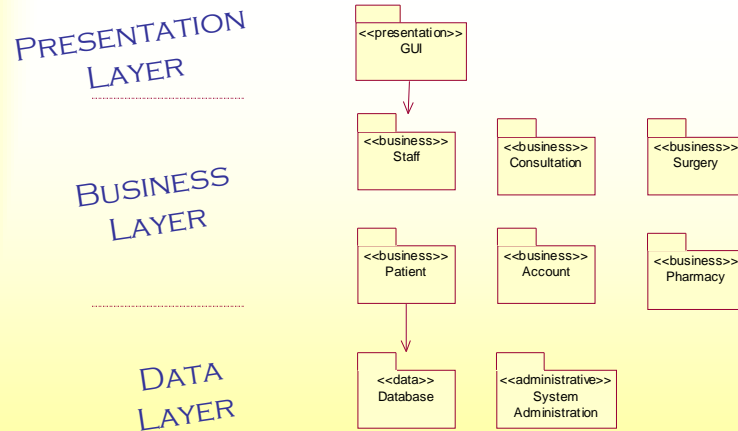


NOTE —
CLARIFIES THE DIAGRAM

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

28

Hospital Management System Package Diagram



"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

29

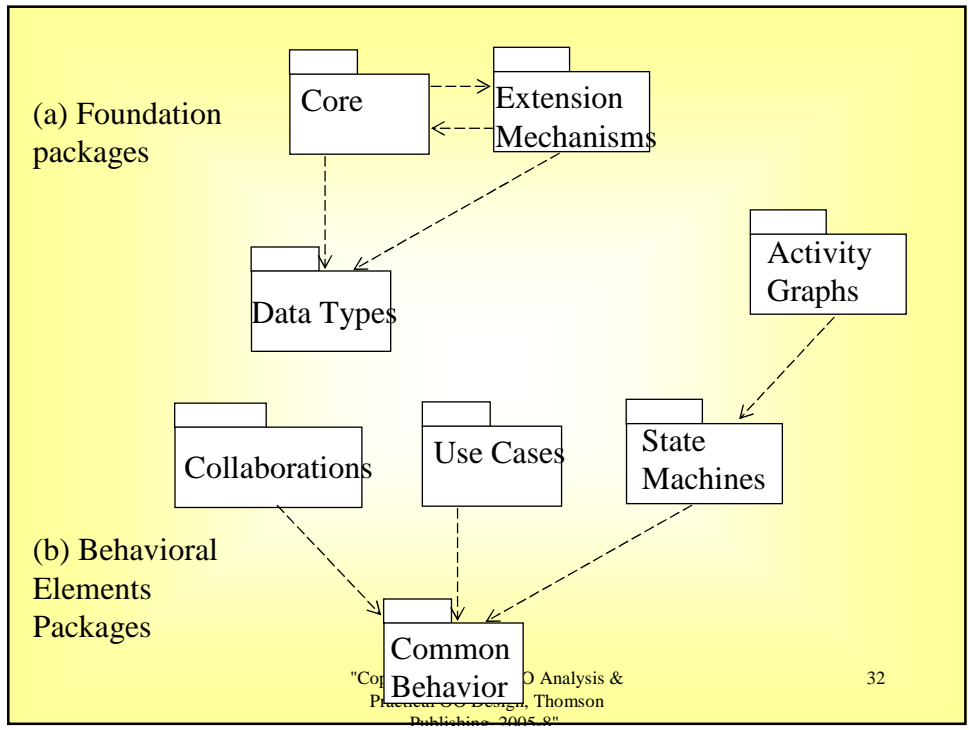
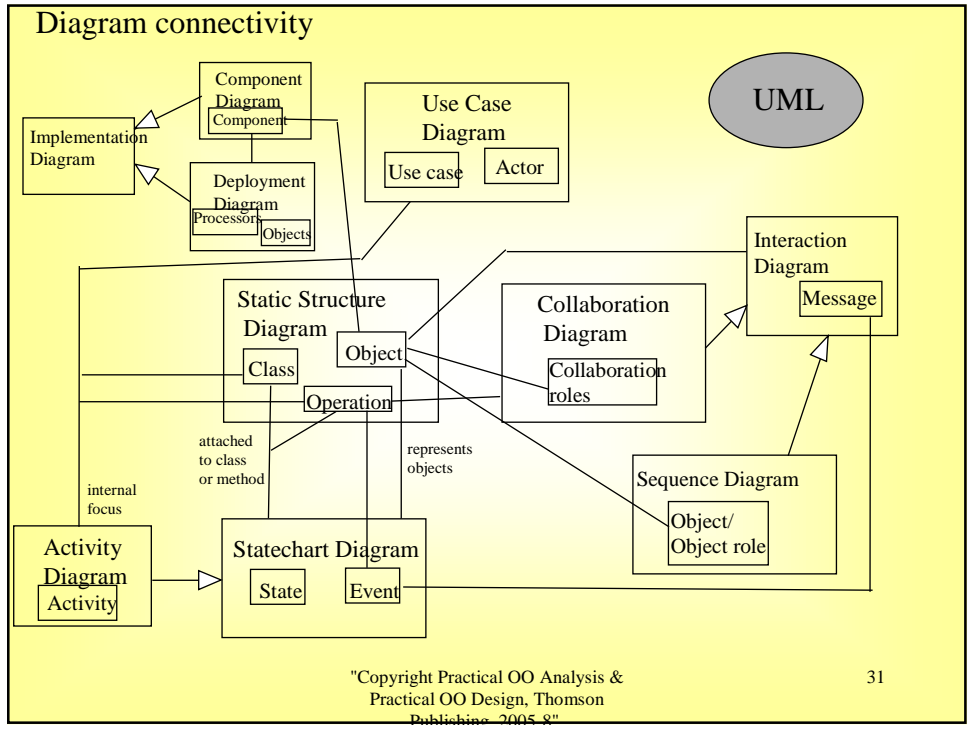
Sub-Module

UML – Meta Models

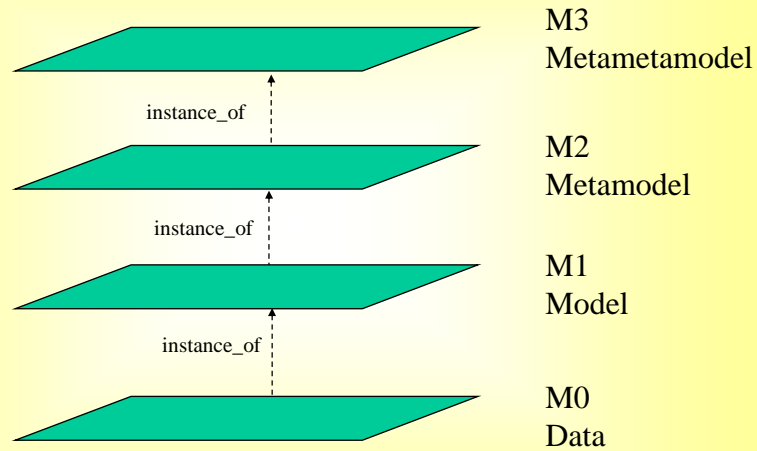
and Layers

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

30



UML lies within a four layer architecture



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

33

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
Publishing 2005 8"

34

WORKBOOK

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