

OBJECT-ORIENTED MODELLING (OOM) - 32536

MODULE 3

**Advanced Class Definitions
(POOD-Chapter 2); Extensibility
mechanisms (POOA-Chapter 5)**

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

1

Module Outline

- Revisiting Classes and Objects
- Representing Classes with UML Class Notation
- Separating Class representation in Analysis and Design
 - Names, Attributes, Operations, Parameters
- UML's Extensibility Mechanism
 - Stereotypes, Notes, Tagged Values and Constraints

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

2

Sub-Module

Representing a Class in UML

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

3

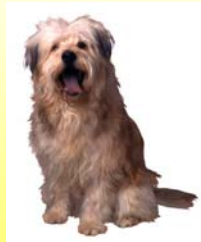
Revisiting Classes and Objects

- Objects are *instances* of a Class
- A Class is a *definition / template / (cookie cutter)* for all objects belonging to that Class
- However:
 - **A CLASS IS NOT A COLLECTION OF OBJECTS**

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

Classes and Objects

- An Object is *abstracted* to a Class;
- A Class is a *DEFINITION* - a *TEMPLATE* - for Objects
- NOTE: A Class is *NOT* a Collection of Objects;



- "DOG" in general is a CLASS.
- My DOG is an OBJECT.
- Dog named *Sparky* is an OBJECT.
- *The* DOG is an OBJECT.
- "DOGS" – representing a group of dogs is *NOT* a class, but a collection of Objects

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

Relevance of a Class in modelling

- A Class can represent:
 - A key business entity (during analysis in problem space; part of Business Object Models)
 - A key solution entity (during solutions design in the solution space)
- Can be a programming entity, but need not always be so
- Provides a template or a shell for objects

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

CIRT and other variations of Classes

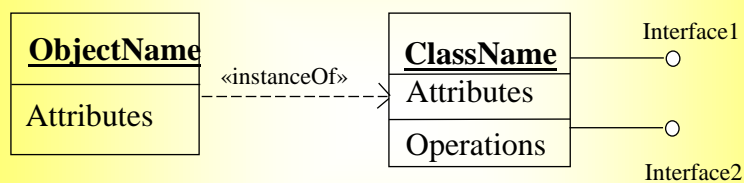
- The term Class is varying used to demonstrate Class, Interface, Roles and Types
- Henderson-Sellers has used the term CIRT in early OPEN modelling work
- Stereotypes can be used to meta-model classes for other representations in practice
 - Such as scripts, data bases, XML and even print

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

7

Classes, Interfaces and Objects: UML

[UML focussed on attributes and operations]



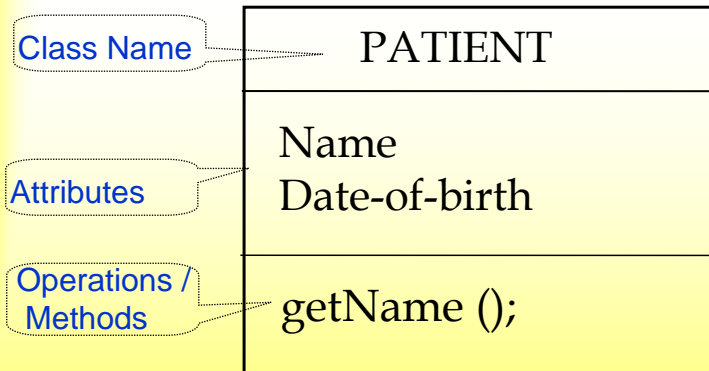
Extra boxes can be added
or existing ones omitted

Copyright OMG, 1997

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

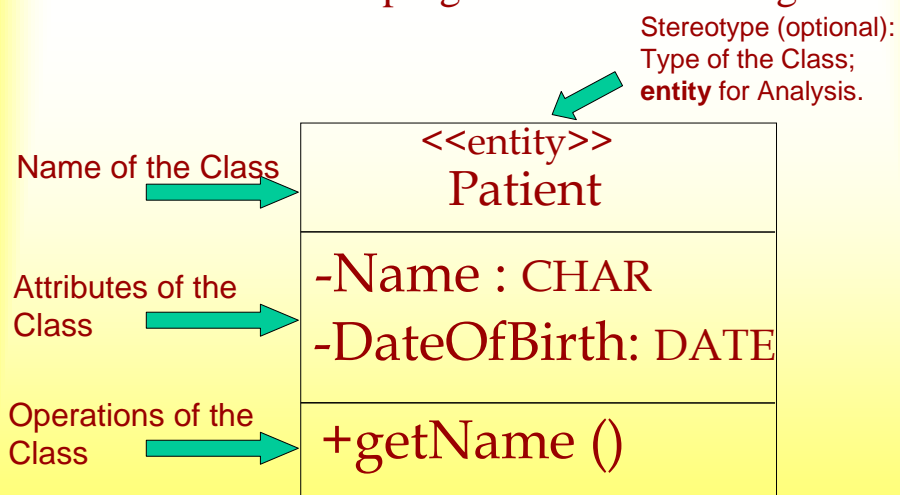
8

A Class in General is Defined in Three Parts



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

Class Definitions start becoming more detailed as we progress with modelling

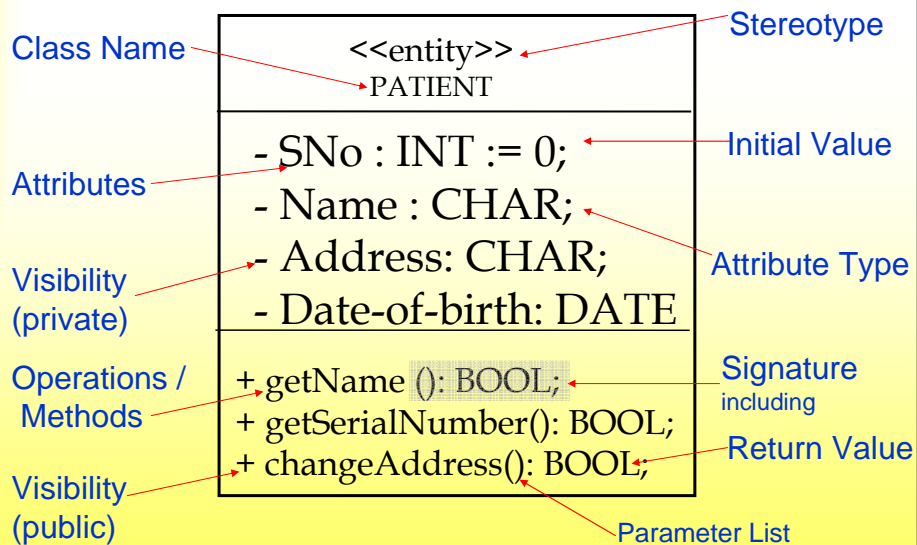


"Copyright Practical OO Analysis,
Practical OO Design, Thomson
Publishing, 2005"

Sub-Module

Detailed Class Notations and Design Details

A Class in Design has Greater Details in the Three Parts of its Definition

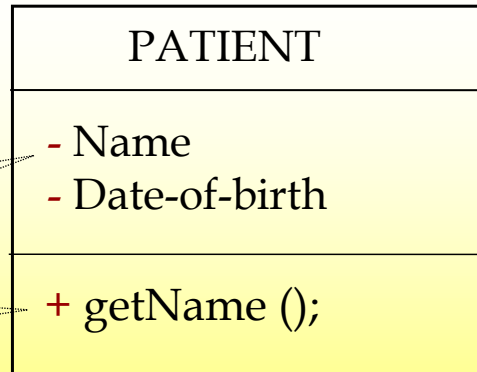


Access Control (Visibility) on Class Attributes & Operations

- Private
+ Public
Protected

Private
Visibility

Public
Visibility



Note: # Protected makes the attribute available to the *Inherited* class. Hence # is meaningful only in inheritance relationship. This, and other aspects of visibility will become clearer as we proceed with discussion on designing Attributes & Operations

Defining Attributes in Design

Generic Attribute Definition:

Visibility <<Stereotype>> Name : Type and Initial Value/Default

Visibility

Stereotype

Name of
Attribute

```

- SerialNo : INT := 0;
- <<entity>> Name : CHAR;
- Address: ADDRESS;
- Date-of-birth: DATE
  
```

Initial
Value

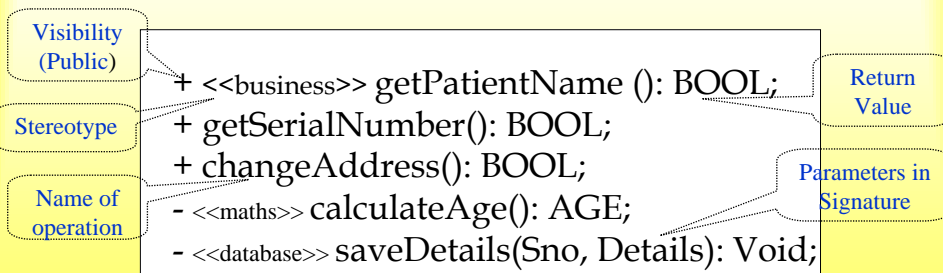
Attribute
Type

Defining and Displaying Operations in Design

Generic Operation Definition:

Visibility <<Stereotype>> Name Parameters in Signature : Return Value

– Shown in the Third compartment of a class



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

15

Module

UML's Extensibility Mechanism

Stereotypes; Notes; Tagged Values
and Constraints

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

16

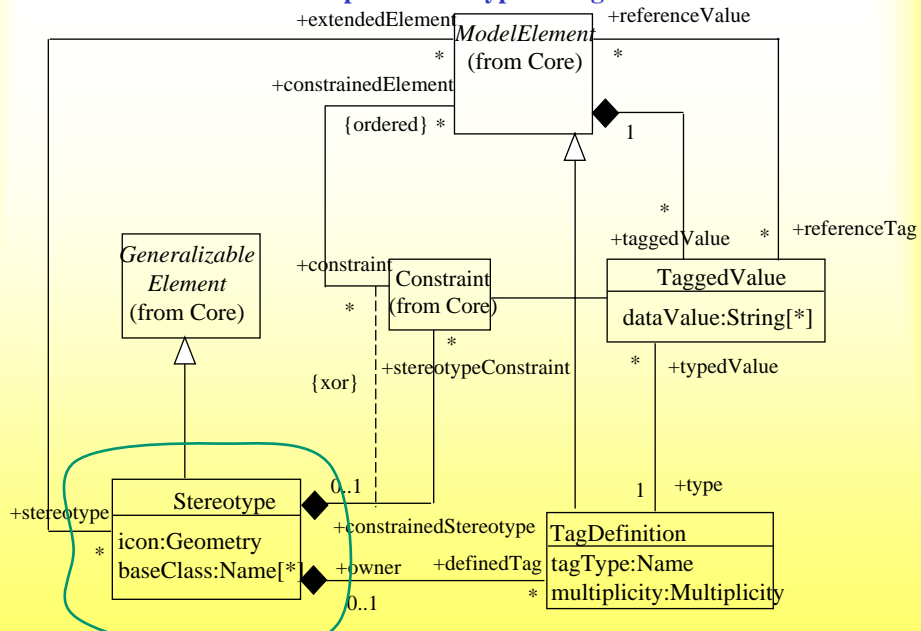
Extending UML

- UML provides three mechanisms to extend itself further
 - **Stereotypes** - Mechanism to classify anything and everything in UML, by far the most popular
 - **Tagged Values and Constraints**
(May not be used during Analysis)
 - **Notes** - Can be used to Extend diagrams and add further Value to UML diagrams

Stereotypes...

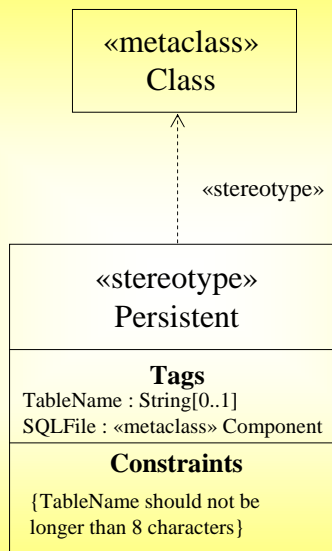
- Stereotypes are 'types' of types
 - For Example: Some Use Cases deal with Interactions, Others with Business Logic
 - Use Cases dealing with Business Logic may be categorized as «Entity» types of use cases or « Entity » Stereotypes
- « » is the notation for stereotypes (guillemets)
- Each UML artifact Has only one Stereotype

The UML Meta-model depicts Stereotype as a 'generalizable element'



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

Graphical form of the definition of a stereotype



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

Stereotypes

« » guillemets

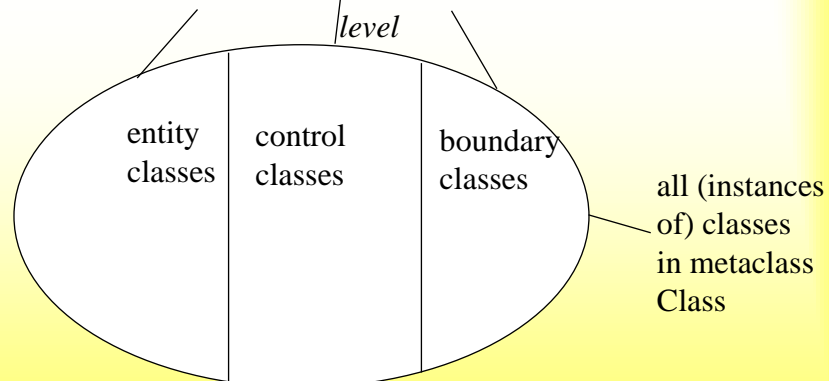


"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

21

Each M2 level partition can be represented using a stereotype

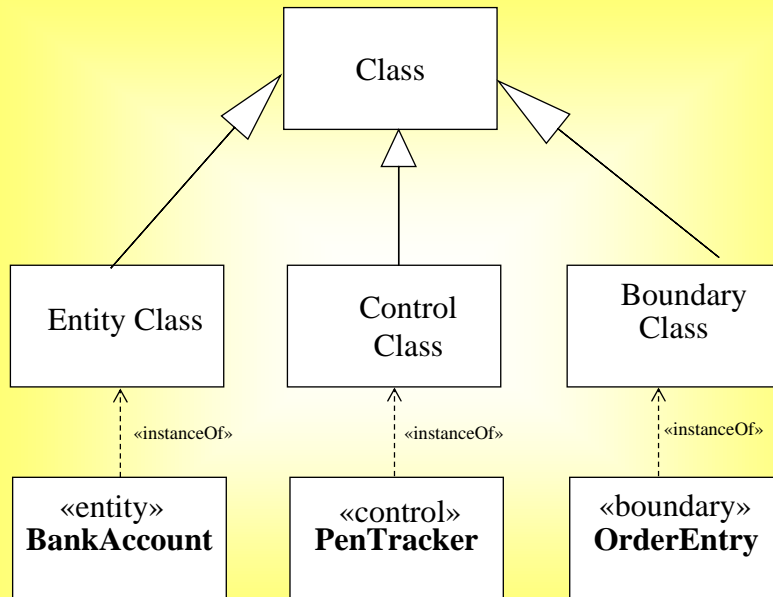
3 stereotypes form a partition
of the extension *at the M2*
level



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

22

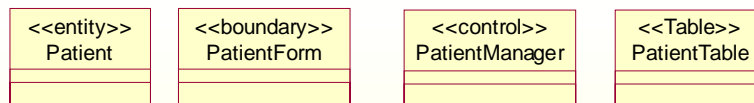
The partition name is the «stereotype» label



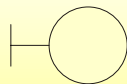
"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

23

Some Stereotyped Classes



Patient



PatientForm



PatientManager

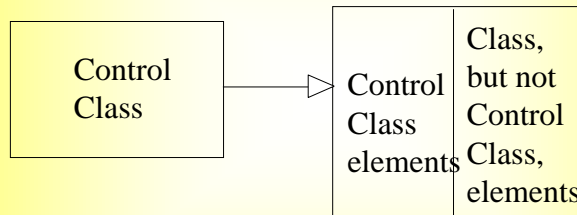


PatientTable

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

24

Can be viewed as a subset, often as a partition,
of the main UML metaclass, here Class.
The partition name is the «stereotype» label,
here «control»

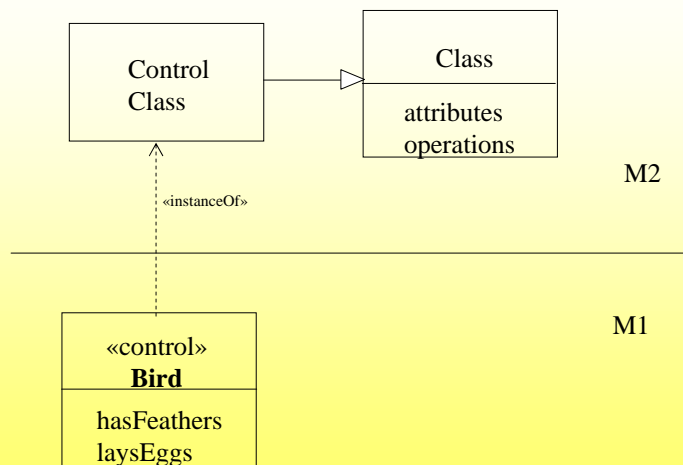


"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

25

Example

A pseudo or virtual metasubtype (often user-defined) e.g.
ControlClass. Think of as another implicit (rather than explicit)
subtype in the (M2) metamodel. For example, a stereotyped class
Bird



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

26

There are, of course, other metatypes that use the stereotype notation while not being stereotypes themselves e.g. «interface», «extend», «include»

UML2 stereotype mechanism is significantly changed.

(The intent in UML2 metamodel for stereotypes is the same as UML 1.4

However, it is still not yet precisely defined in both versions)

Constraints

- A semantic condition or restriction expressed in text (e.g. informal English, OCL)
- An assertion, non-executable
- Place inside braces { } + possibly connect by dashed line
- May be in a Note icon
- Can be used to express pre- and post-conditions of operation and for class invariants
- BUT note that constraints often lead to inter-dependencies between classes, hence thwarting reusability

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

29

Predefined Constraints

- {xor}

For Generalization

- {overlapping}
- {disjoint}
- {complete}
- {incomplete}

Stereotypes can also be
Applied to Constraints:

Such as

«Invariant»

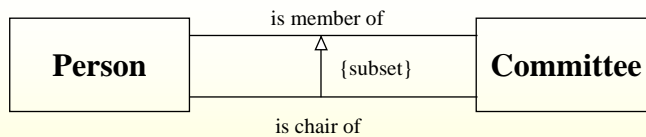
« Pre-condition »

« Post-condition »

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

30

Example



{subset} constraint on Dependency
or, in OCL:

Person

```
self.committeeMembers->includes(self.committeeChair)
```

Tagged Values

- Extends properties of model element but not its instances i.e. not the same as a class attribute - think of as metadata
 - Tagged value is a pair: tagName + value
 { tagName = value }
- or just { tagValue }

Predefined Tagged Values

- tagName=persistence
tagValue = {transitory}, {persistent}
- tagName=semantics
- tagName=usage (Activity Graph)
tagValue = {uses} {modifies}
- tag for profile packages
tagValue = {applicable Subset}

Other useful keywords

{abstract}
{leaf}

Other commonly used, but not predefined, tagged values

{readonly} {private} {obsolete}
{Version=1.4} {Author = ABC}
as enumeration

Example

BankAccount
{abstract, author=BH-S}

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

35

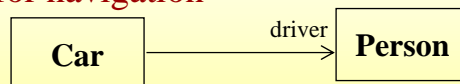
Example constraints

For Person class, `person.age > 21`

Set operations

`person.address->size`

Rolenames of associations (or else class names)
used for navigation



gives context
or focus → Car

or Car

`self.driver`
`self.person` ↑
specific instance

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

36

More complex examples

Car

```
self.driver->forAll(d|d.age>=21)
```

ensures all drivers are aged at least 21

or

Car

```
self.driver->forAll(age>=21)
```

To find all such drivers

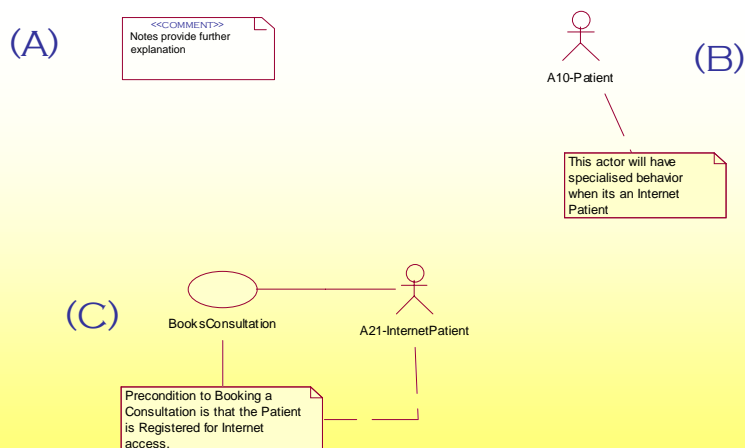
Car

```
self.driver->select(age>=21)
```

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

37

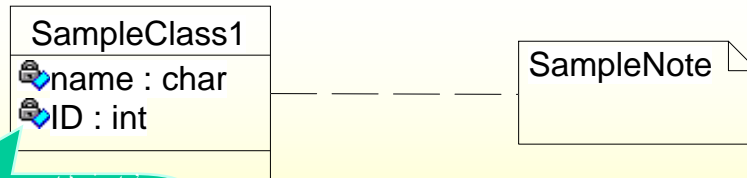
Notes or Comments



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

38

NOTES: Invaluable in Enhancing Quality of UML diagrams



Icons represent 'private' visibility '-'. Variation due to the ROSE tool.

Notes can be drawn on ALL diagrams; They SHOULD be added to ALL diagrams to Enhance Readability;

Notes help provide Additional Explanations on Diagrams;

Notes are represented by "Dog ear Rectangle";

They are then linked to any other 'things' on the diagram.

Notes/Comments may also be stereotyped, if required.

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

39

Conclusions

- Discussed Classes, Interfaces and Objects
- Discussed UML's extensibility mechanisms that allow UML to be customised for use in specific projects.
- These mechanisms included stereotypes, constraints, tagged values and notes.

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

40

WORKBOOK



EXERCISES

For The Class Room!!

Workbook Exercises

1. Identify 5 Classes from your surroundings or from any domain and list them.
2. Select any 2 classes and identify 10 attributes and 8 operations and document the class and show whether the class is abstract or not. Show the class stereotype.
3. Fully design and document any two attributes with their accessibility, stereotypes, types and initial value.
4. Fully design and document any two operations with their accessibility, stereotypes and signatures including return value.



PROJECT WORK : **(DURING TUTORIALS IN THE LAB);**

Follow the Project Work Requirements given at the End of
the Chapter;
Discuss Project Work with Team Members and Tutor;
Carry Out the Project Work