Object Oriented Analysis & Design

Lecture 01

Course Instructor: **SYED SAQLAIN HASSAN**

Course Code: SE321

Spring 2023

Object-Oriented Analysis and Design

- Overall goals of the object-oriented paradigm is the selection of classes, the relationships among them and their realization to implement systems.
- Course Objectives
  - Learning concepts and techniques necessary to effectively use system requirements captured in use cases to drive the development of a robust design model
  - Focus on Training by applying UML 2.2 notation to fundamental OOAD

- At the end of this course students will be able to:
  - Develop a working understanding of formal object-oriented analysis and design processes.
  - Develop the skills to apply OO Analysis and OO Design techniques to any given project.
  - Develop an understanding of the risks inherent to large-scale software development.
  - Learn (through experience!) techniques, processes, and artifacts that can mitigate these risks.
  - Implement a pilot OO Application

Course Contents

- Feedback Session & Introduction to the course and course policies
- Introduction to OOAD
- Use case modeling
  - Exercise: Use case modeling
- System Sequence Diagrams
- Domain Modeling
  - Exercise: Domain Modeling
- Operation Contracts
  - Exercise: Operation Contracts
- Introducing Design
- Sequence Diagrams
- Interaction Diagrams
  - Exercise: Interaction Diagrams
- Class Diagrams
  - Exercise: Class Diagrams
- GRASP Patterns
- UML Diagrams
  - Package Diagrams
  - Component diagrams
  - Deployment diagrams
  - Activity diagrams

Books

- Text Book(s)
  - Craig Larman, Applying UML and Patterns, 2nd Edition.
  - Head First Object-Oriented Analysis and Design 1st Edition by Brett D. McLaughlin, Gary Pollice, Dave West
  - UML 2.0, Documentation://www.rational.com

Evaluation / Assessment

Class Participation	5%
Quizzes	10%
Individual Assignments	5%
Midterm Written Exam	20%
Final Written Exam	60%

- Evaluation percentages can be changed
- No retake of assignments, mids or final at all.
- No extension in deadline of assignments.
- Submission guidelines must be followed.

Importance of Feedback

- Participation and full involvement by every student is a must in this Course.
- Ask questions any time during lecture

- Instructor would provide feed back on work with expected revision

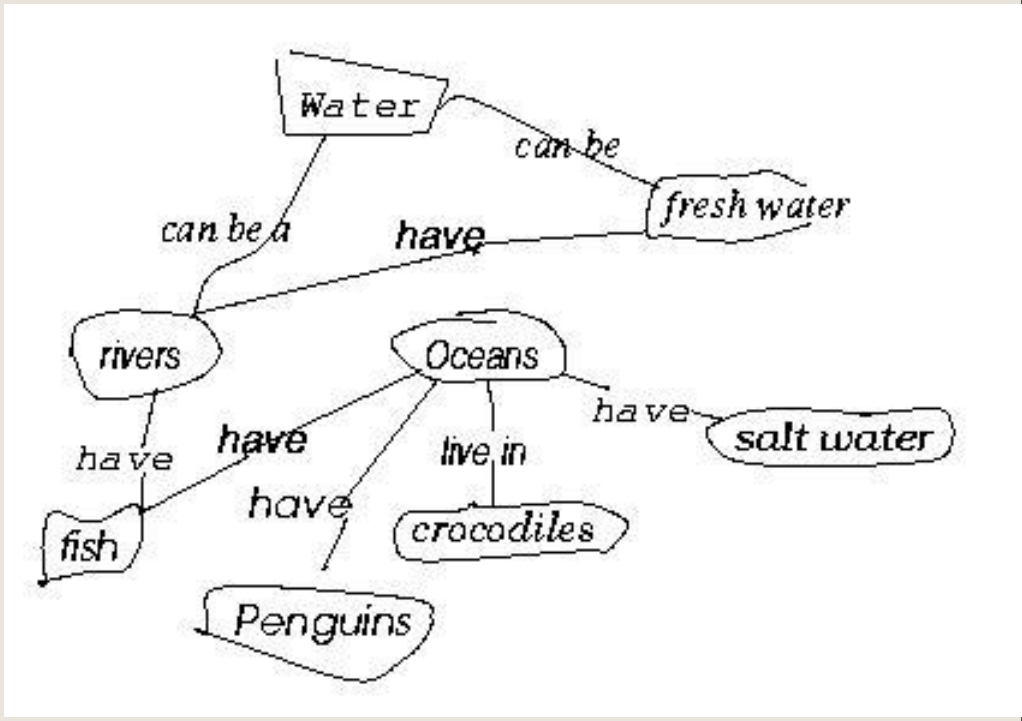
Classroom Conduct

- Behave as scholars in the University.
- No mobile telephones
- No browsing on Tablets/Phablets/Laptops/Computers
- Arrive on time
- Academic Dishonesty
  - Academic dishonesty (copying/cheating) in any portion of the academic work/exam would result in zero marks.

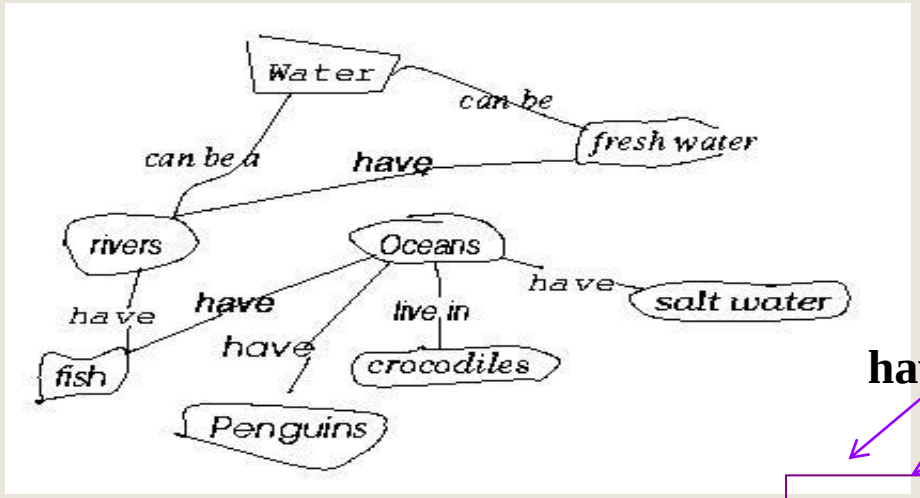
Object Oriented

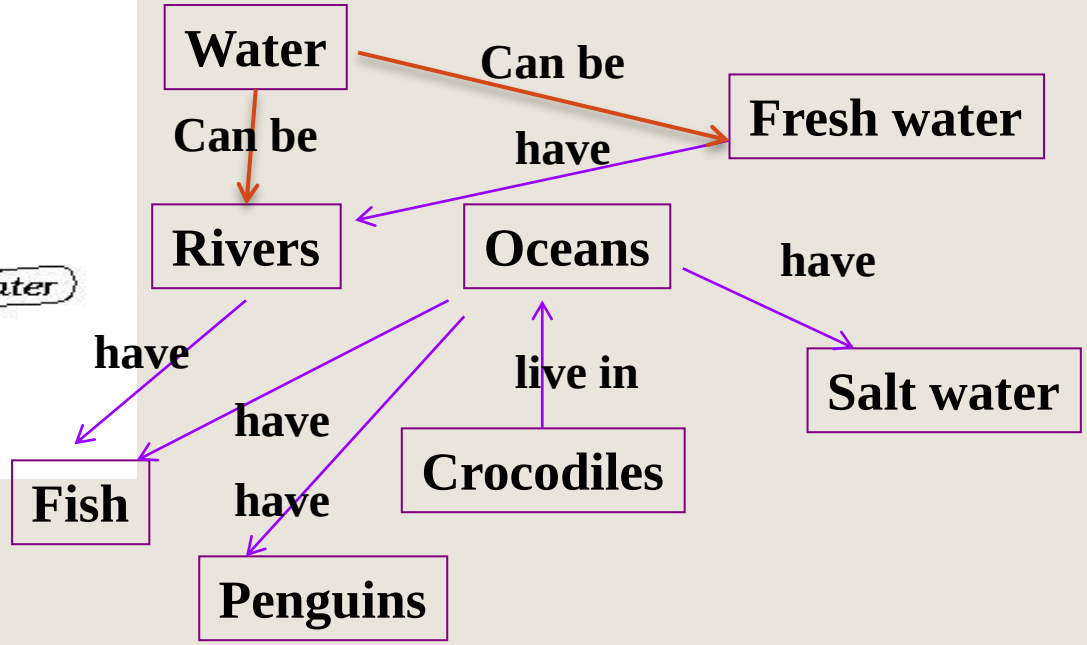
Concepts

Hasham is given a list of concepts (Water, salt water, Oceans, Penguins, Fish etc) and asked to think about their relationship and draw what he has in his mind.



Hasham constructed a concept diagram through which he explain his own understands of concepts and their relationship.


Water
can be
fresh water
can be a
have
rivers
Oceans
have
salt water
have
have
live in
have
fish
crocodiles
Penguins


Water
Can be
Fresh water
Can be
have
Rivers
Oceans
have
have
live in
Salt water
have
Crocodiles
have
Fish
have
Penguins

What is an Object?

- An "object" is anything to which a concept applies
  - Things drawn from the real life (problem domain or solution space).

  e.g., a living person, a job role, a software component in the solution space.

- A structure that has identity and properties and behavior
- It is an instance of a collective concept, i.e., a class

An Object has…

- Operations (Behavior):
  - Work
  - Drive
  - Jump
- Attributes:
  - Height
  - Eye color
  - Hair color
  - Weight

Class

- *Class* is a collection of objects that share common properties, attributes, behavior and semantics, in general.
  - A collection of objects with the same attributes (variables) and behavior (function/operations).
- Classification
  - Grouping of common objects into a class
- Instantiation.
  - The act of creating an instance.

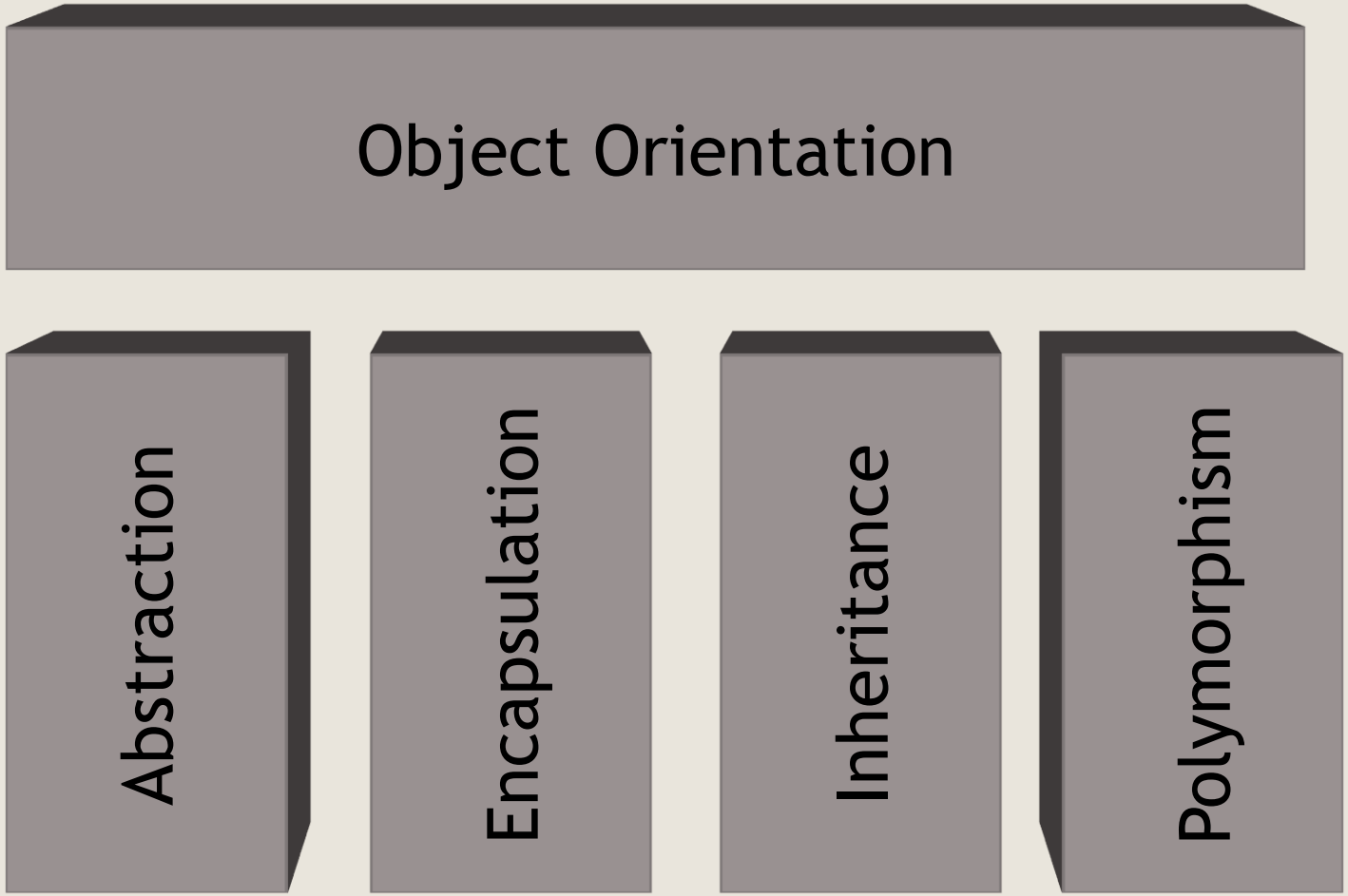
- A class represents a template for several objects and describes how these objects are structured internally
- Objects of the same class have the same definition both for their operations and their information structure
  - Class is an implementation of objects

- An instance is an object created from a Class
- System's behavior is performed via the interactions between instances

Relationships

- Static:
  - relations existing over a long time
  - objects know about each other existence
- Dynamic:
  - relations which two objects communicate with each other
  - object sending stimuli to other
  - stimuli - events, messages

Basic Principles of Object Orientation



Abstraction

- Abstraction means to focus on the essential features of an element or object
  - ignoring its extraneous or accidental properties.
- The essential features are relative to the context in which the object is being used.
- Example:
  - Class Student,
  - attributes :reg_number, name, course (Included)
  - height and size_of_shoe are excluded, *since they are* ***irrelevant*** *in the perspective of the educational institution*.

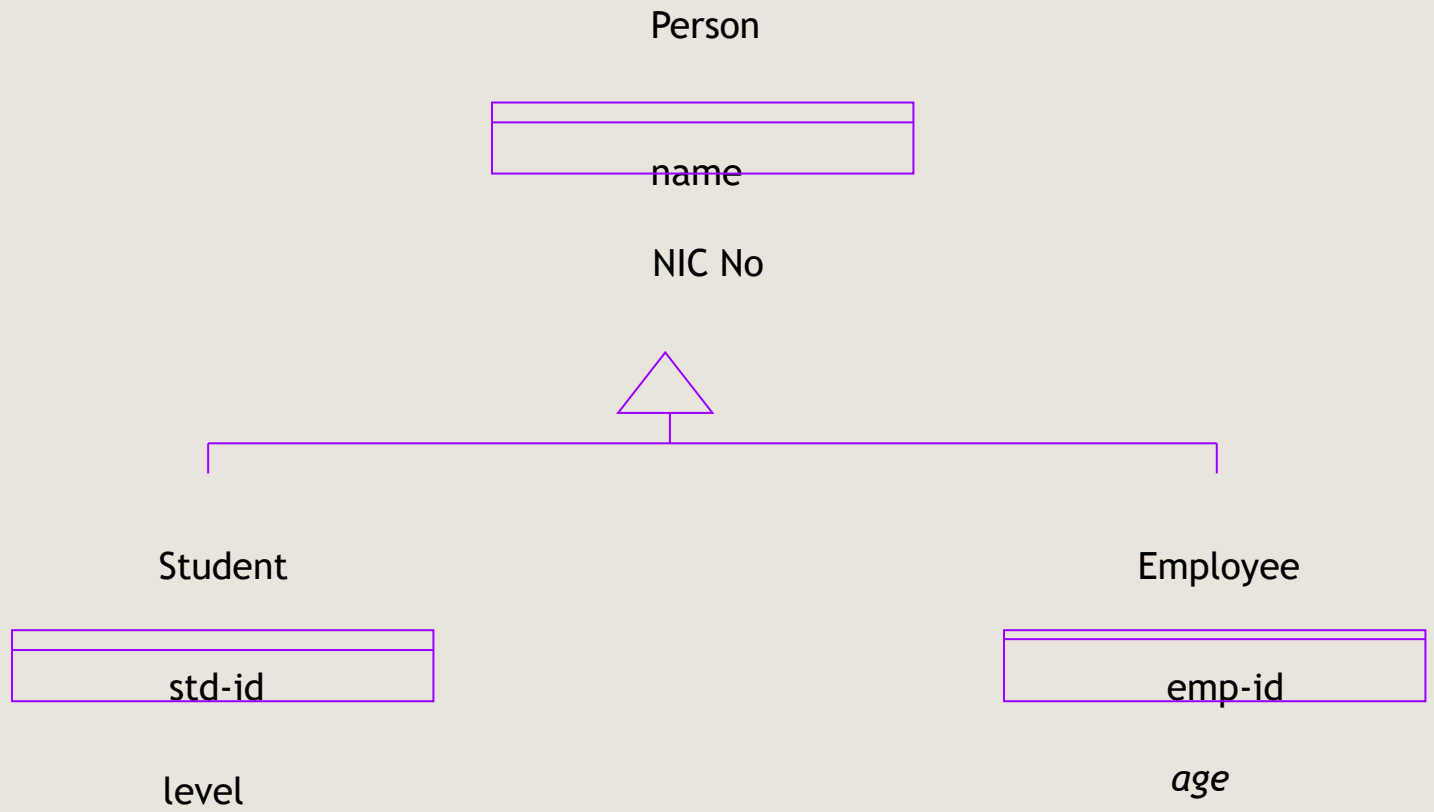
Encapsulation

- A concept of ‘Self-containing’
- Information hiding
  - ‘internal’ structure is hidden from their surroundings
- Behavior and information is represented or implemented internally
- Functionality and behavior characterized by ‘interfacing’ operations

Inheritance

- Specialization: The act of defining one class as a refinement of another.

- Inheritance: Automatic duplication of superclass attribute and behavior definitions in subclass.
  - Subclass: A class defined in terms of a specialization of a superclass using inheritance.
  - Superclass: A class serving as a base for inheritance in a class hierarchy

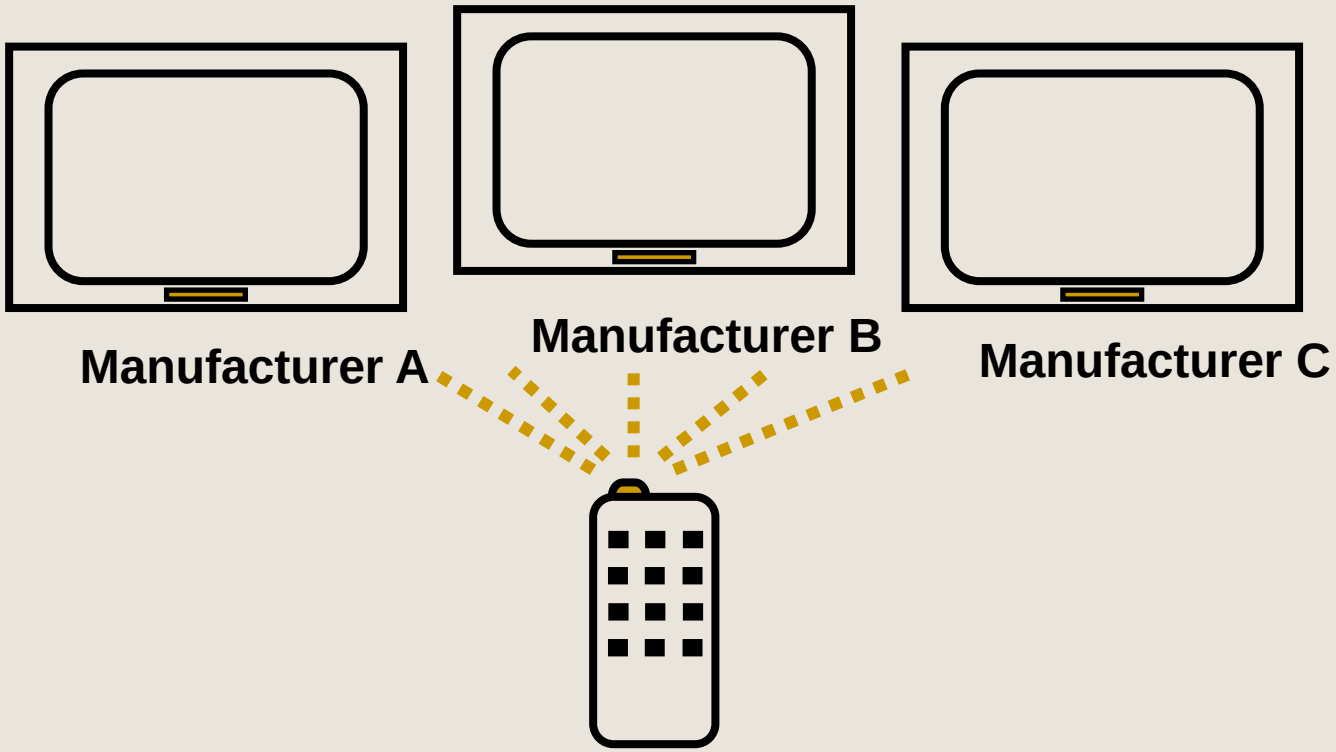

Person
name
NIC No
Student
std-id
level
Employee
emp-id
age

Why Inheritance?

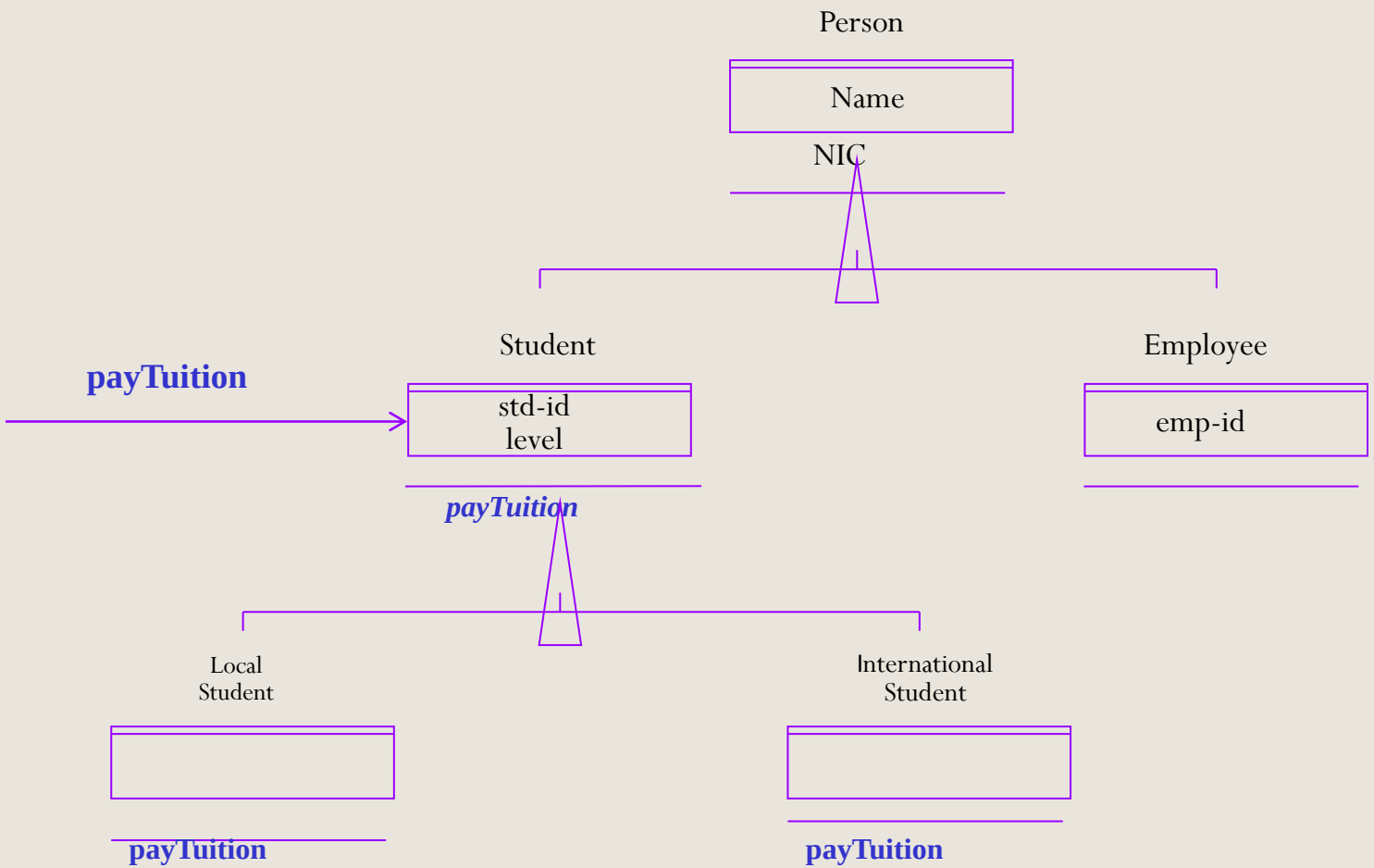
- Show similarities
- Reuse common descriptions
- ‘Software Reuse’
- Easy modification of model by performing modification in one place
- Avoid redundancy, leading to smaller and more efficient model, easier to understand

Polymorphism

- The ability to hide many different implementations behind a single interface



- Polymorphism:
  - Play command
    - There is a toddler sitting in front of some blocks and a teenager siting in front of a piano.
    - An adult walks into the room and says “play”.
    - The toddler plays with the blocks and the teenage plays the piano.
  - Car accelerator on different cars.



Objects of different classes respond to the same message differently.

Why Polymorphism?

- A very strong tool for allowing system designers to develop flexible systems (e.g. Student)
- Designer only need to specify what shall occur and not how it shall occur
- To add an object, the modification will only affect the new object, not those using it

State of an Object

- "State" is a collection of association an object has with other objects and object types
  - A "state change" is the transition of an object from one state to another.
  - An "event" is a noteworthy change in state [Rumbaugh]

What is Object-Oriented Application?

- An application that has collection of discrete (completely separate) objects, interacting with each other to solve the problem.
  - Interactions through message passing
  - A sender object sends a request (message) to another object (receiver) to invoke a method of the receiver object's)

- Are OO Application any good?
- A system which is designed and modeled using an object-oriented technology is:
  - Easy to understand
  - Directly related to reality
  - Natural partitioning of the problem
  - More flexible and resilient to change
  - Systems can be developed more rapidly and at a lower cost

What is OOAD?

Analysis

- understanding, finding and describing concepts in the problem domain.

Design

- understanding and defining software solution/objects that *represent* the analysis concepts and will eventually be implemented in code.

OOAD

- Analysis is object-oriented and design is object-oriented. A software development approach that emphasizes a logical solution based on objects.

Object Oriented Analysis and Design

Investigation of the problem

Analysis

Domain Concepts, Objects

Logical Solution

Design

Book
Title Author
Print

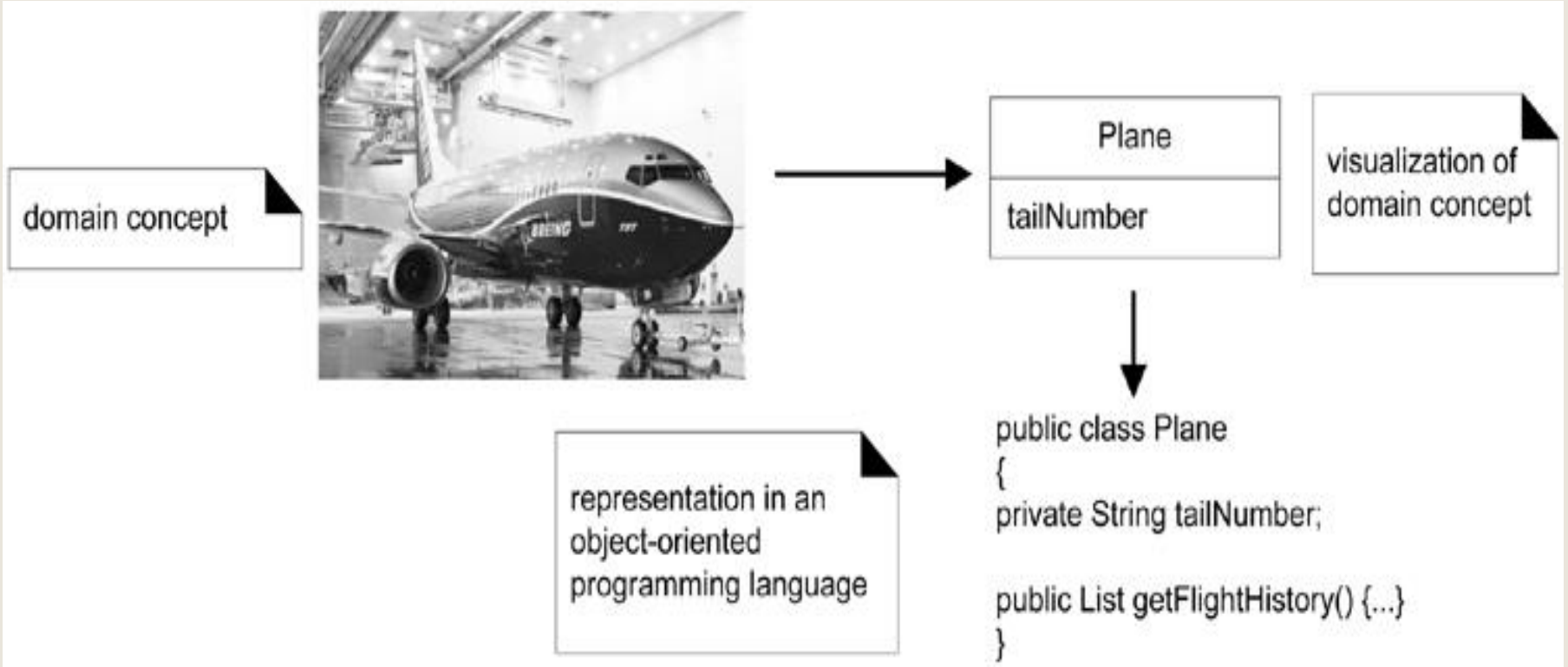
Design Class

Code

Construction

```
Public class Book
{
   Private String title;
   public void print ();
}
```

OO Language Code



- 1. Finding objects
- 2. Organizing objects
- 3. Describing how objects interacts
- 4. Defining the operations of objects
- 5. Defining objects internally

Object-Oriented Modeling

- Object-oriented modeling is the process of preparing and designing what the code will actually look like.
- We do OO modeling using UML notations
  - The UML is a standard diagramming notation
    - The UML is not OOA/D or a method, it is simply notation
      - a language for OOA/D and "software blueprints,"
      - a form of communication with others

Design Patterns

- Certain tried-and-true solutions to design problems can be (and have been) expressed as best-practice principles, heuristics, or patterns
  - Named problem-solution formulas that codify exemplary design principles.
- GRASP patterns
- Observer
- Model View Controller
- Controller
- Abstract Factory
- Singlton

Unified Modeling Language

The Unified Modeling Language (UML)

- To quote OMG (the creators of UML)
  - The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.
- UML is an open standard
  - It can be used with all processes, throughout the development life cycle, and across different implementation technologies

- The UML is the de facto and de jure standard diagramming notation for object-oriented modeling.
- The UML does not guide developer/Coder in how to do object-oriented analysis and design.
- UML supports
  - the entire software development lifecycle
  - diverse applications areas
  - based on experience and needs of the user community
  - Supported by many CASE tools

- UML is an expressive language that can be used to describe pretty much everything about software application.
  - object technology: objet, class, relationships, scenario, Use Case
  - component-based development: component, ActiveX/COM, CORBA
  - large scale system, application partitioning

- Why UML!!
- Having a standardized notation helps a developer
  - acquire skills in how to create a good design,
  - mastering a set of principles to identifying suitable objects.

- UML supports

Objects

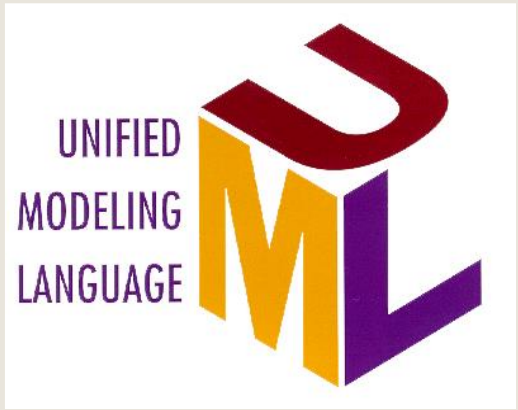
Business Objects

Relationships

large scale system

ORDBMS
Oracle

Classes



application partitioning

Components
Microsoft

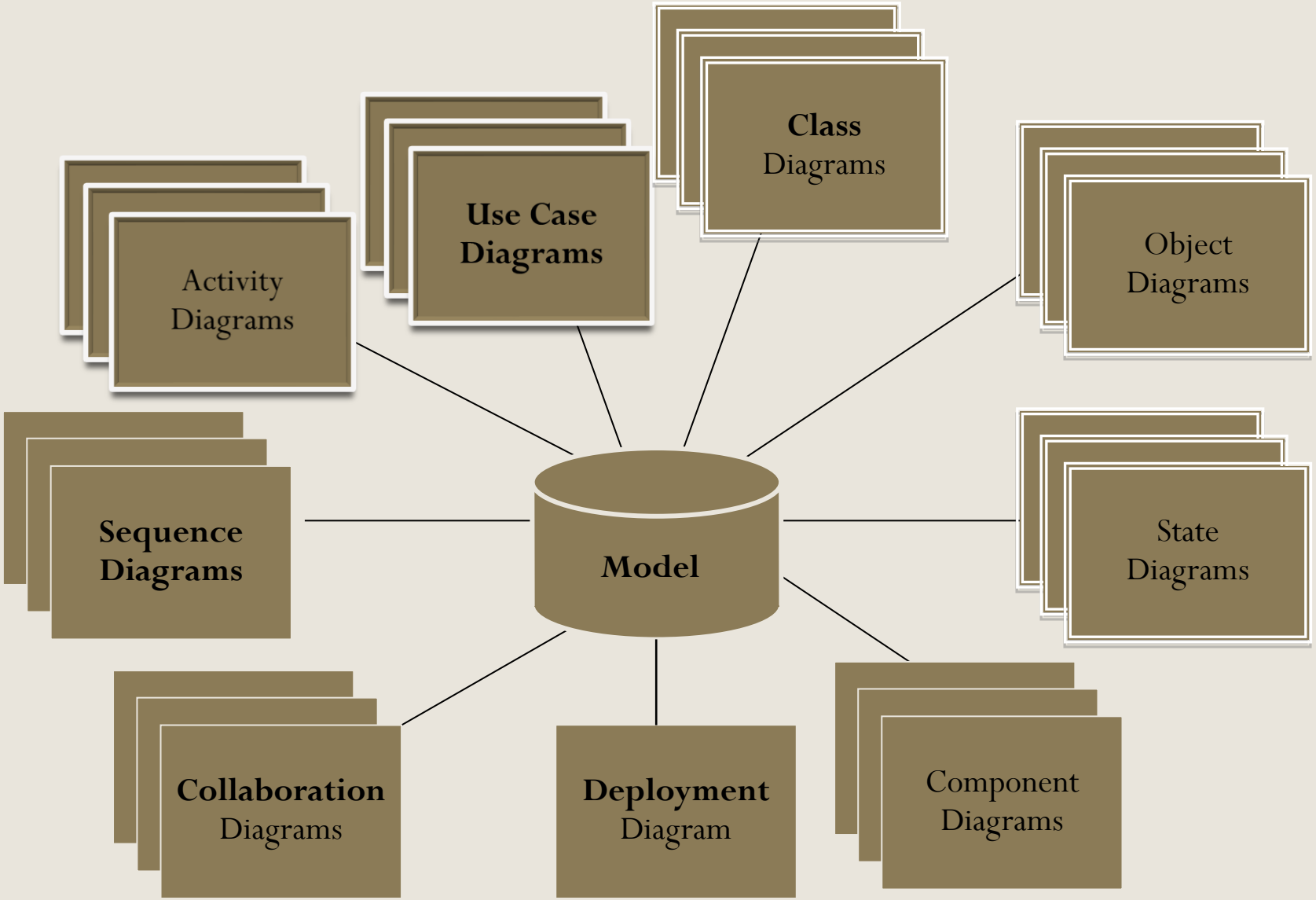
Scenarios

CORBA
OMG

Use Cases

ActiveX/COM
Microsoft

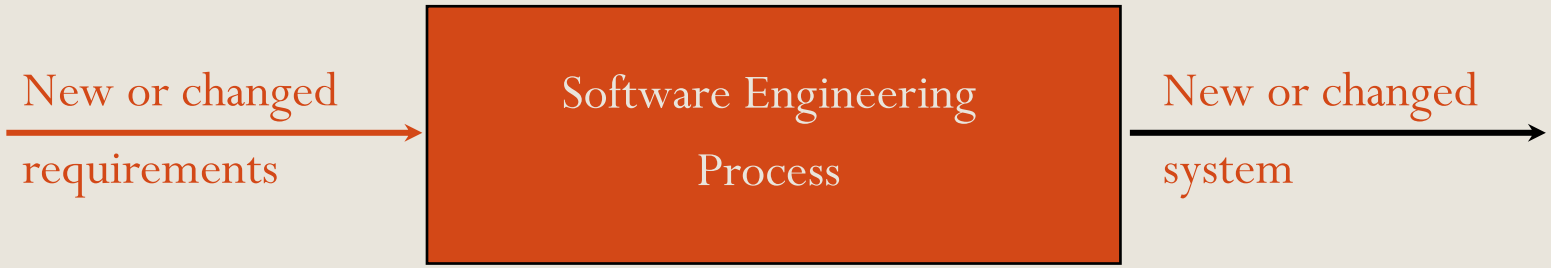
Business Process


Class Diagrams
Use Case Diagrams
Activity Diagrams
Object Diagrams
Sequence Diagrams
Model
State Diagrams
Collaboration Diagrams
Deployment Diagram
Component Diagrams

The Unified Process

Software Engineering process

- Software Engineering process defines **Who** is doing **What**, **When** and **How** to build a software product or to enhance an existing one.



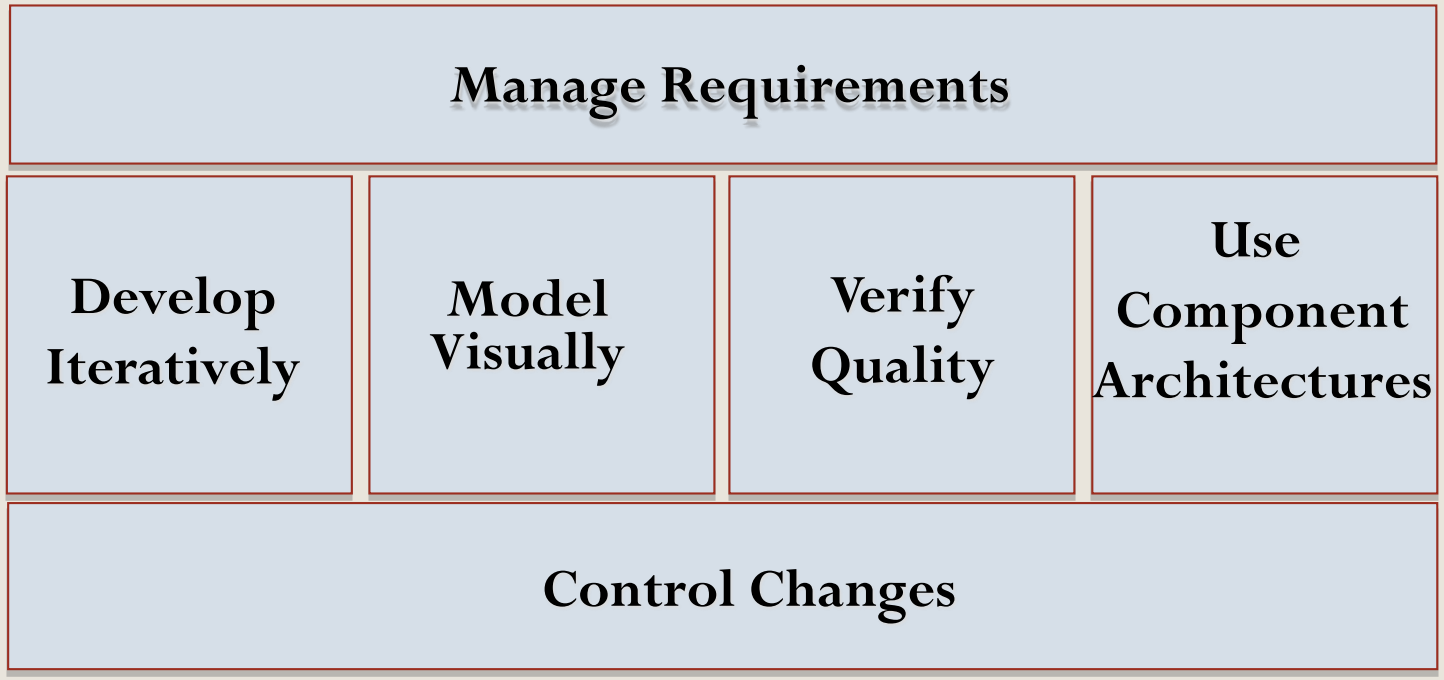
Unified Process

- The Unified Process is software development process for building object-oriented systems.
  - Rational Unified Process (RUP) is a refinement of the Unified Process and has been widely adopted.
  - Interactive and incremental development.

- Best Practices and Key Concepts in UP
  - UP is short time boxed iterative and adaptive development.
  - UP uses object technologies, including OOA/D and OOP.
  - Address high-risk and high-value issues in early iterations

- Continuously engage users for evaluation, feedback, and requirements.
- Continuously verify quality by testing early, often, and realistically.
- Apply use cases.
- Model software visually (with the UML).
- Carefully manage requirements.
- Practice change request and configuration management.

- Unified Process describes how to effectively implement the six best practices for software development



UP's Lifecycle Phases

- The Unified Process has four phases:
  - Inception - Define the scope of project
  - Elaboration - Plan project, specify features, baseline architecture
  - Construction - Build the product
  - Transition - Transition the product into end user community/environment

- The project work is organized in iterations across four major phases with clear milestones, outputs/artifacts:
- 1. Inception
  - approximate vision, business case, scope, vague estimates.
- 2. Elaboration
  - refined vision, iterative implementation of the core architecture, resolution of high risks, identification of most requirements and scope, more realistic estimates
- 3. Construction
  - iterative implementation of the remaining lower risk and easier elements, and preparation for deployment
- 4. Transition
  - beta tests, deployment

Thank you!!!

- Points to ponder!!
  - Difference between
    - Inheritance and Polymorphism.
    - Abstraction and Encapsulation
  - How Interfaces formalize Polymorphism?