HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

System Analysis and Design

AC3010

ONE LOVE. ONE FUTURE.

Chapter 2: *Object-oriented systems analysis and design and Unified Modeling Language*

Course syllabus

- 2.1 Object-oriented model and object-oriented perspective
- 2.2. Basic features of an object-oriented system
- 2.3. Object-oriented analysis and design process
- 2.4. Unified Modeling Language UML

2.1 Object-oriented model and object-oriented perspective

Object concept

- Object in the real world is an entity that we can perceive normally (physical entity, conceptual entity, etc.).
- It has states (attributes, properties) and behavior (operation)

Object	State	Behavior
	- Speedometer: How fast is it moving? - Odometer: How many miles has it driven? - ….	- Move forward - Stop - Reverse - ….
	- Author: Who is the author? - Pages number: How many pages does it contain ? -…	- Buy - Borrow - Count the number of pages….

Object concept

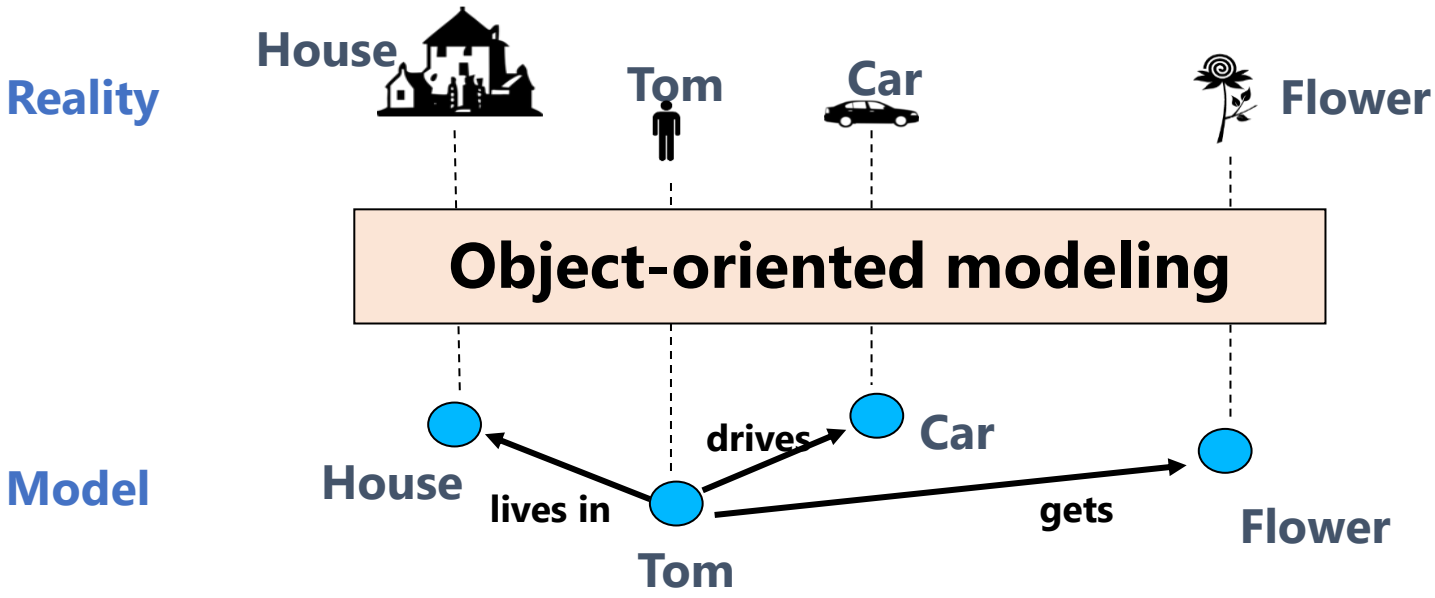
- State of an object: the conditions in which the object exists.
- State of an object can change over the time.
- Behavior determines how the object acts and responds to the outside world.
- = a set of messages that object can respond to (the operations that the object performs).

Name: J Clark
Employee ID: 567138
Date Hired: July 25, 1991
Status: Tenured
Discipline: Finance
Maximum Course Load: 3 classes

Professor Clark's behavior
Submit Final Grades
Accept Course Offering
Take Sabbatical

Object-oriented perspective

- Object orientation is a technique of modeling a system into **multiple objects** and **interactions** between them
  - All element representations in a system are “objects”

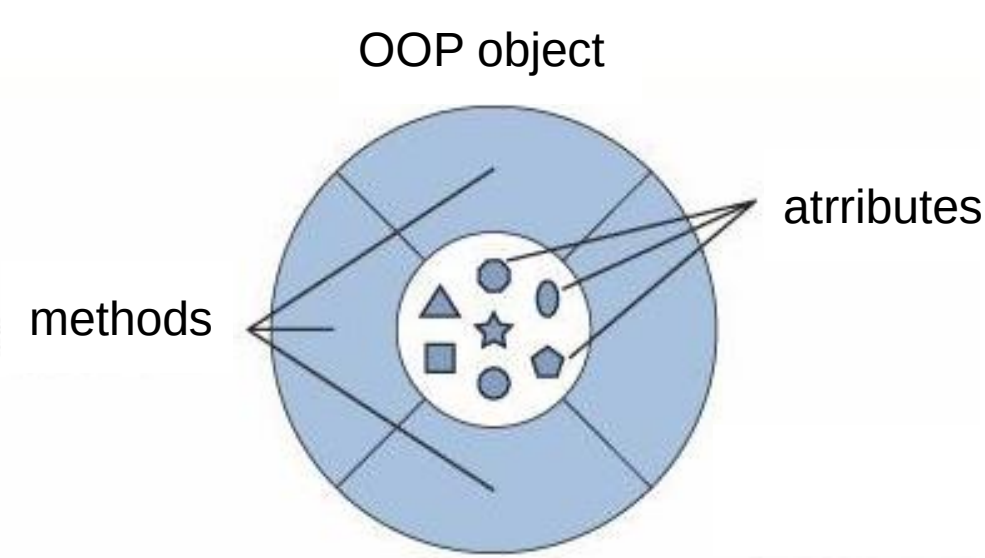


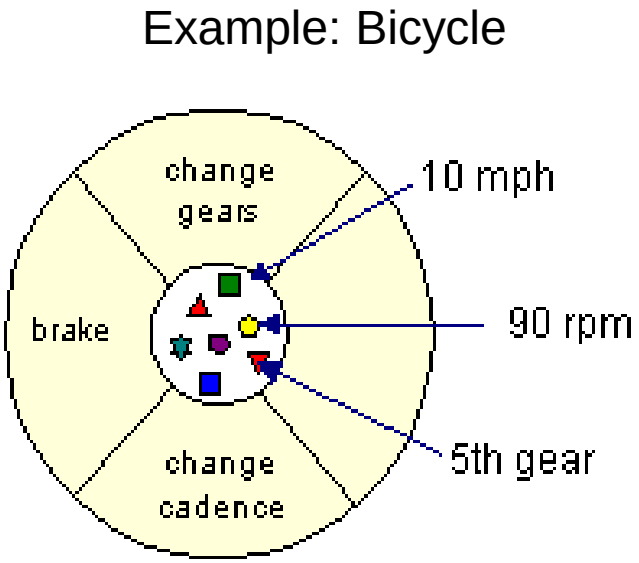
Object-oriented perspective

- According to Alan Kay:
  1. All things are objects
  2. Each object has specific attributes (state) that form Class of objects
  3. Each object in the program has its own independent attribute value and takes up its own memory
  4. All objects belonging to the same class have the same methods (behavior)
  5. A software program can be considered as a set of objects that interact with each other through object methods

Object in OO Programming

- Object in OO Programming is the software entity encapsulating (wrap) associated attributes and methods

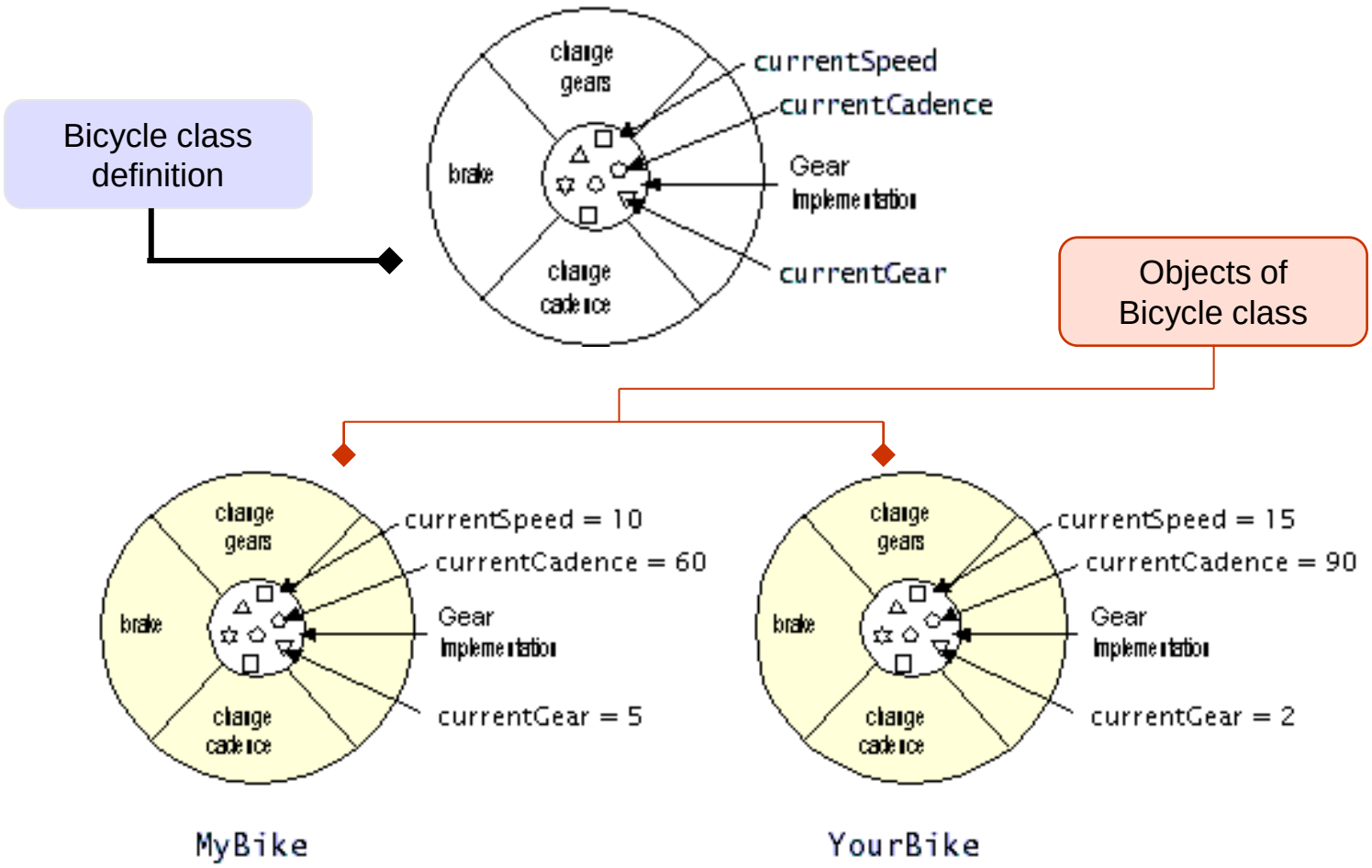




Class in OO Programming

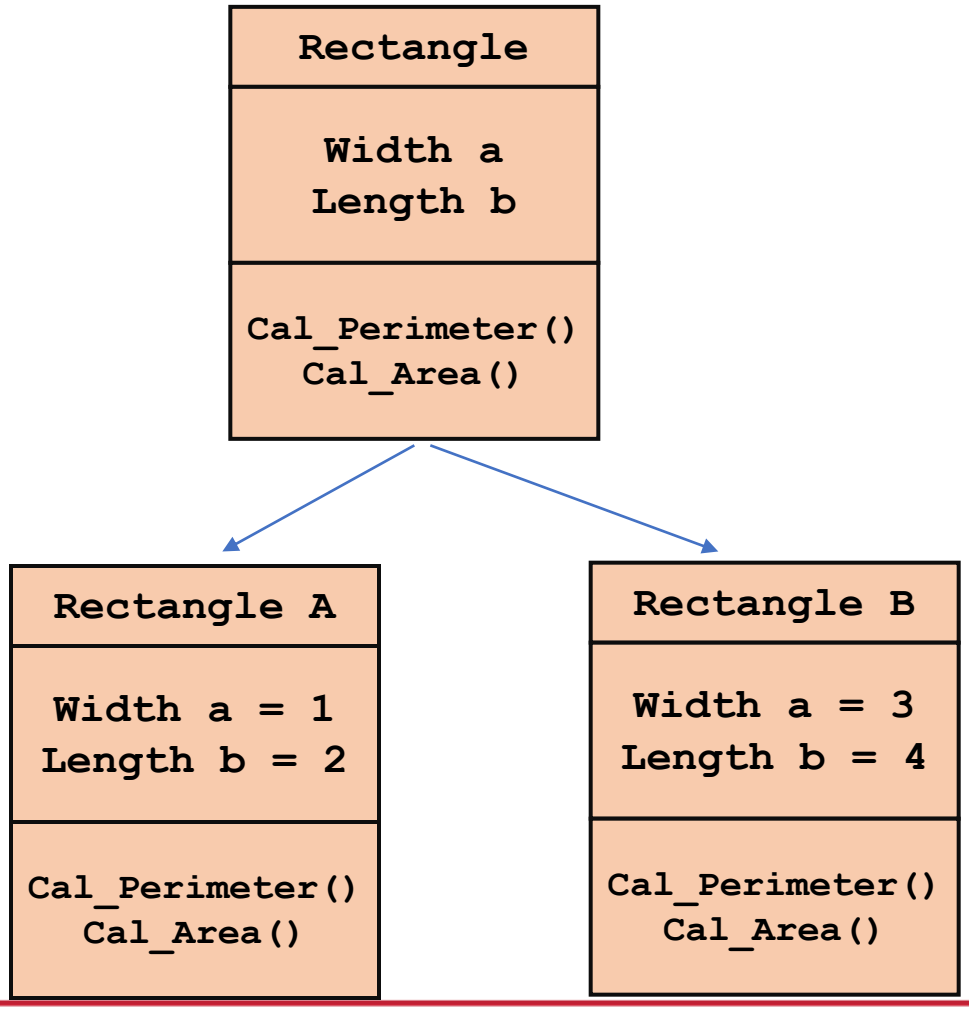
- Class is a design (blueprint) or a prototype (template) for all objects of the same kind
  - Exp: Bicycle class is a common template for all bicycle objects
- A *class* specifies the common attributes and methods of all individual *objects* of the same kind
- Each object is a specific *instance* of a class
  - Exp: each bicycle object is an instance of the Bicycle class
- Each instance of a class has its own *instance attribute*
  - Exp: the first bicycle instance is set at 5 gears while the second bicycle instance is set at 3 gears.

Bicycle Class example



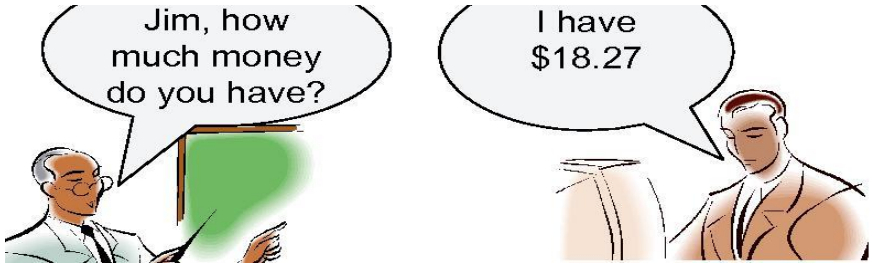
Relationship between object and class in OO Programming

- Objects are also called *instances* of a class
- Each instance of a class has its own *instance attribute*

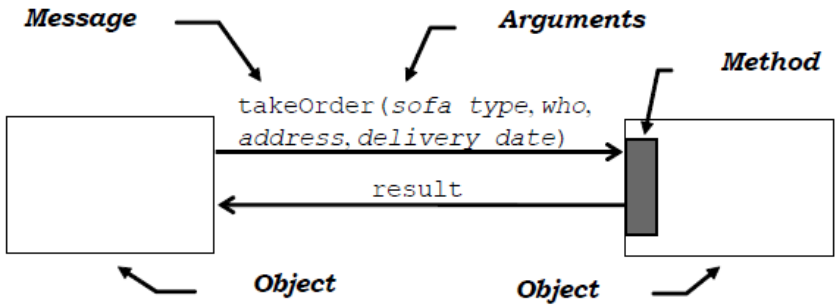


Interaction between objects

- In the real world:

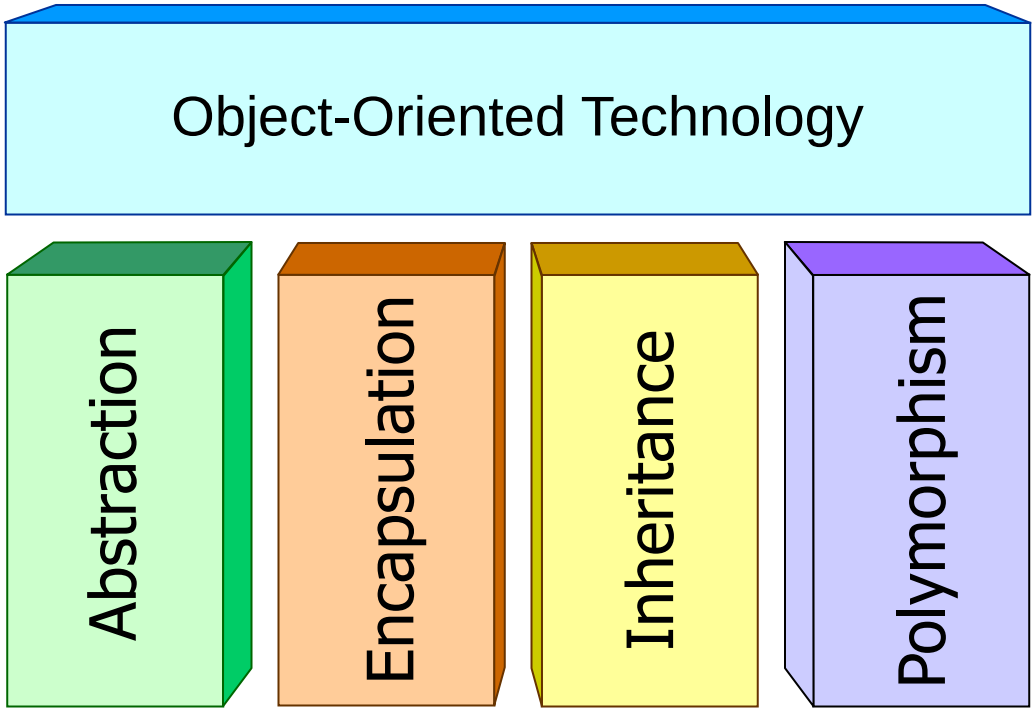


- Objects in OOP communicate via messages:
  - Message passing is the way to interact between objects
  - One object can send a message to another object (message sender)
  - The target object receives the message and responds to the message by performing some actions (a method).



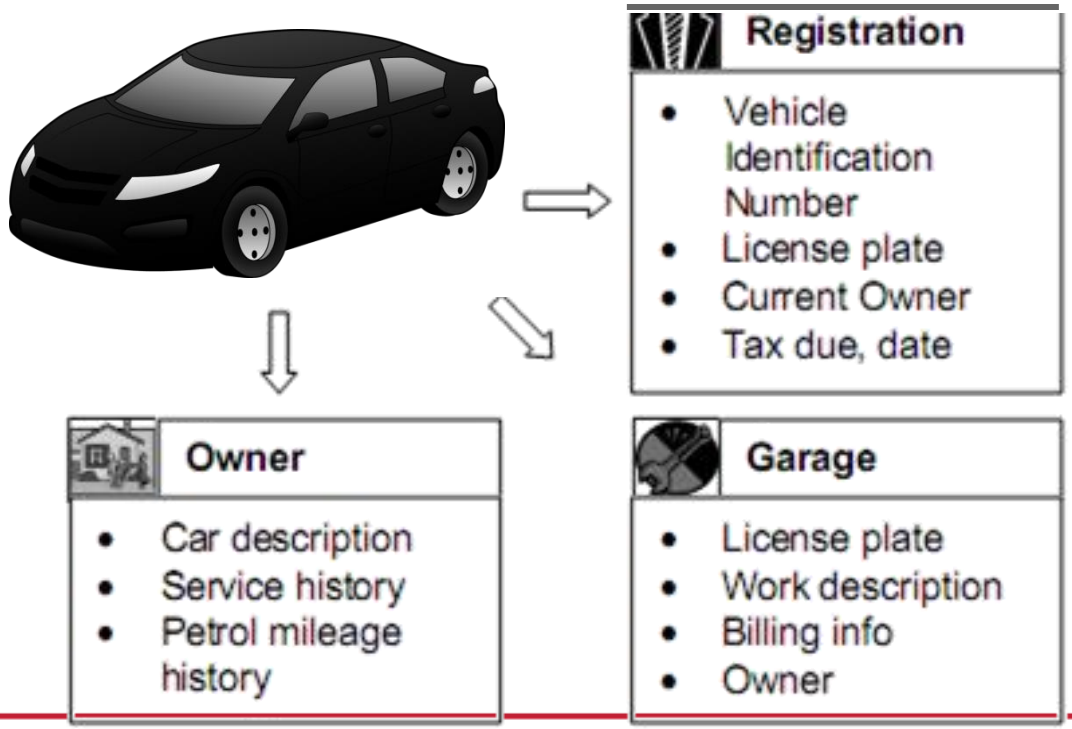
2.2. Basic features of an object-oriented system

2.2. Basic features of an object-oriented system



Abstraction

- A conceptual process : describe general information / properties/ features of objects.
- Focus on the basic features of an entity, features that distinguish one entity from other entities.
- Depends on the viewing angle
  - Some properties could be important in certain situation but not necessary in other situations

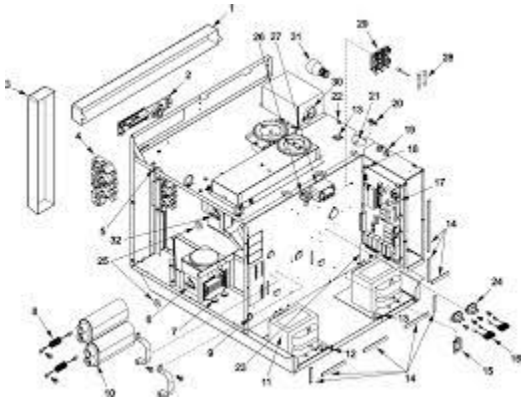


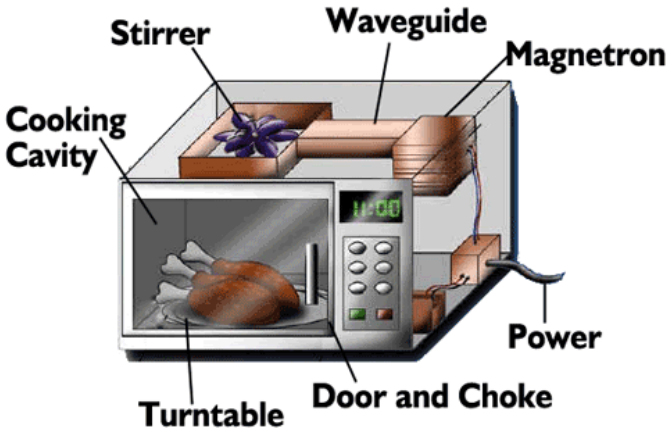
Understand the abstraction in OOP

- Data abstraction:
  - Describe data in different ways depending on the problem
  - To distinguish different entities in that context
- Class is the result of data abstraction
  - Class is a conceptual model, describing the entities
  - A class is an abstraction of a set of objects
  - Attribute are also abstract

Encapsulation

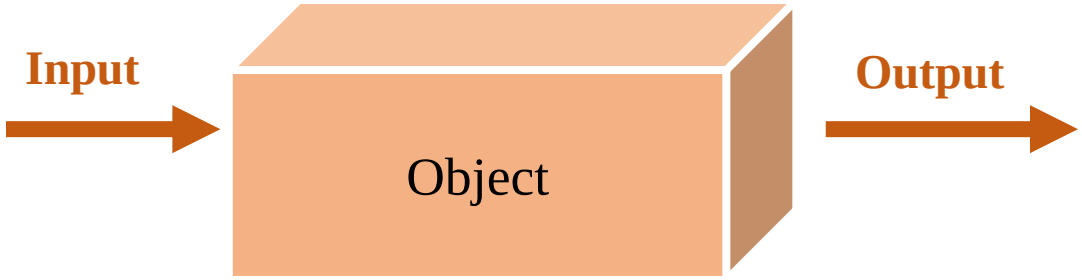
- Hide the inside implementation details
- Provide an interface to the outside world
- Usage is unaffected by internal details (Users don't have to care about the execution inside an object).





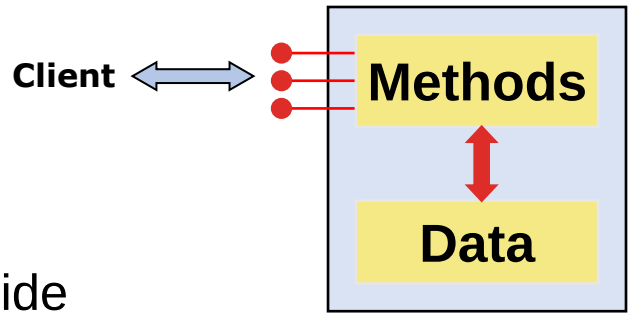
Understand the Encapsulation in OOP

- Data/attributes and behaviors/methods are packaged in a class
- An object is an entity encapsulated with the purpose:
  - Provides set of certain services
  - The encapsulated object can be seen as a black box - the jobs inside are hidden from the client
  - Although the design/source code inside is changed, the external interface is not changed



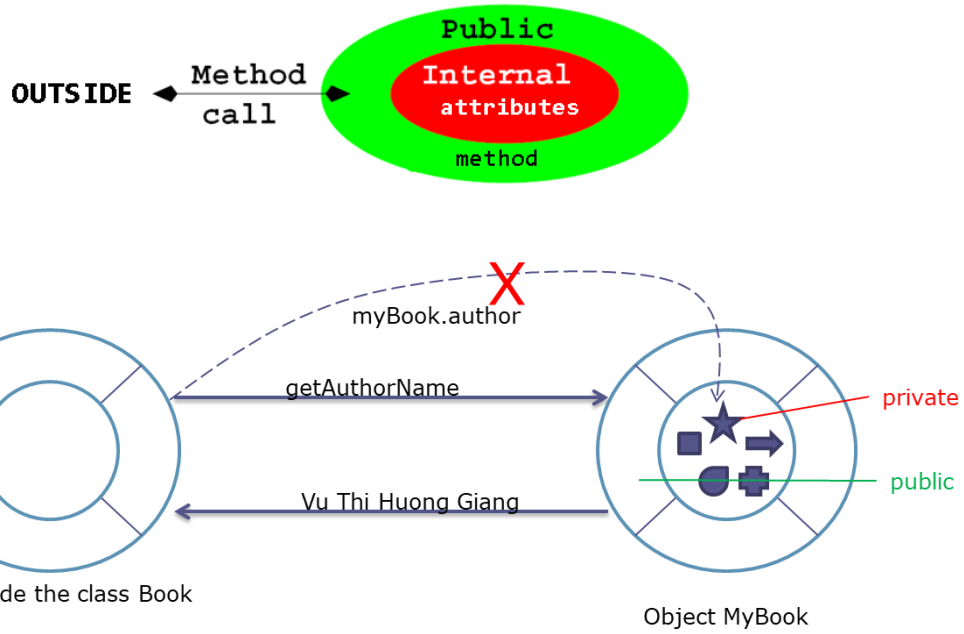
Understand the Encapsulation in OOP

- Once encapsulated, an object has two views:
  - Inside: Details of the properties and methods of the class
  - Outside: Services that an object can provide and interact with the rest of the system
- Access specifier/modifier
  - determines the visibility of a member to the others
  - *private* property or method : Inside only
  - *public* property or method : Open for outside



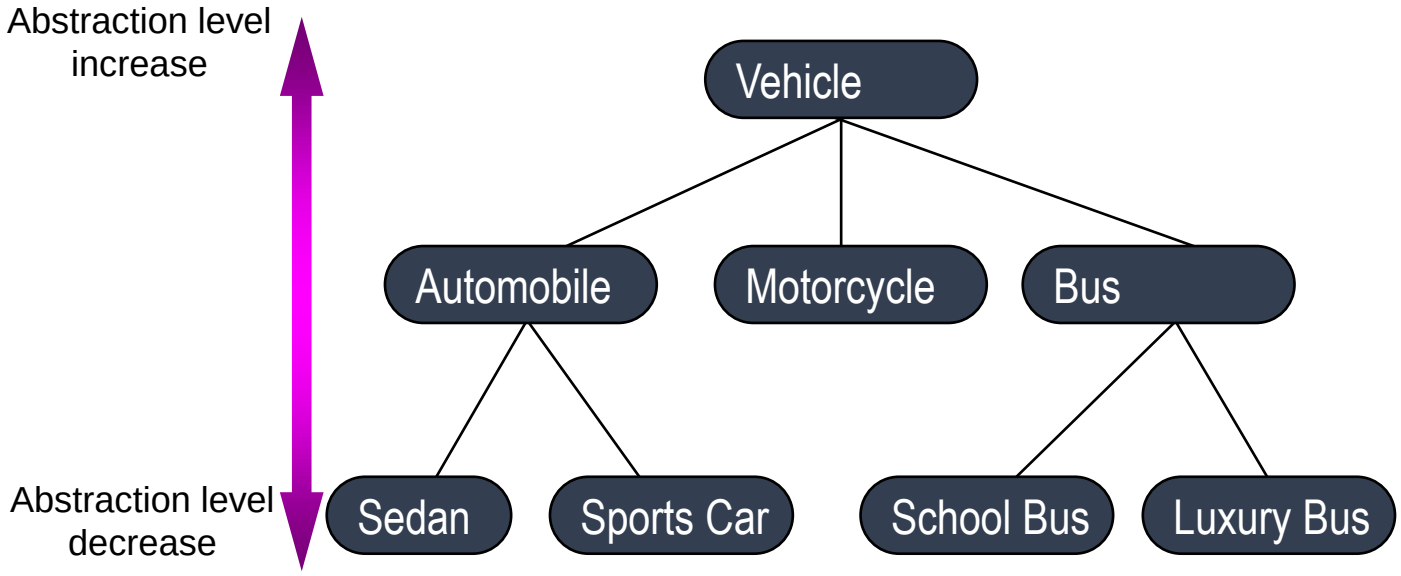
Understand the Encapsulation in OOP

- Once encapsulated, the data can be hidden
  - By assigning *private* access modifier
  - Avoid unauthorized changes or falsification of data
  - Other objects that want to access this private data must go through the methods of the class that have *public* access modifier



Inheritance

- Order (rank) abstraction level into a tree structure
- Help on understanding the similarities and differences between classes
  - Objects at the same level in the hierarchy have the same level of abstraction
- Understand the Inheritance in OOP
  - Class Inheritance



Polymorphism

- Polymorphism: “one name, many forms”
  - Describe an object in different ways
  - Perform an action in different ways
- For example:
  - One person is considered as an Intern, or a Staff of a company.
  - Depend on this person is “treated” as an Intern or a Staff, the suitable method is applied at a time.
- For example:
  - If you travel, you can choose a car, boat, or plane
  - No matter what vehicle you go by, the result is the same: getting where you need to go

    → Different ways to perform a service
- Understand the Polymorphism in OOP
  - Method overloading
  - Method overriding
  - Upcasting/Downcasting for objects

2.3. Object-oriented analysis and design (OOAD) process

2.3. OOAD process

- In SDLC: Turn the requirements of the problem into a clear design with object-oriented perspective.
- A continuum of ideas and practices for software analysis, design, and programming that focuses on ***objects*** and their ***interactions***, rather than processes and data
- Strengths of OOAD
  - Reflects real world entities more closely
  - Reusability: Encourages architectural, design, and code reuse
  - Robustness: Promotes a consistent communication technique that is understandable by both business and technical participants
  - Extensibility: Builds systems that are more easily extended or changed
  - Reliability: Promotes quality of solutions
  - Works well for large, complex software systems

2.3. OOAD process

- OOAD is divided into 2 phases
  - Object Oriented Analysis (OOA)
  - Object Oriented Design (OOD)
- OO Analysis focuses on finding and describing objects or concepts in the problem domain
- OO Design focus on:
  - Defining and detailing the software objects found in OO Analysis
  - How software objects interact to fulfill the requirements
  - OO design models are developed close to implementation
  - OO systems promote re-use of programming elements

Object Oriented Analysis

- Define software requirements
- Specification of software requirements through the model of objects and the interactions between them
- Creating a model with real-life concept and object components, easy to understand for users
- Modeling entities, keeping their structure, relationships, and behavior between them

Object Oriented Analysis

- Example with a car sales room:
  - Entities:
    - Client
    - Salesman
    - Order slip
    - Payment slip (invoice)
    - Car
  - Interactions and relationships between the above entities:
    - The salesman leads customers on a tour of the car showroom.
    - Customer chooses a car
    - Customer writes car booking ticket
    - Customer pays for car
    - Cars are delivered to customers

Object Oriented Design

- Implementing conceptual models which is the output of the OOA step
- Concepts in OOA are mapped to implementation classes.
- Constraints, interfaces are designed.
- The result is a detailed specification of the system to be built, according to the selected particular technology

Object Oriented Design

- Organize the program into collaborative object sets
- Each object is an instance of a class
- Design on the results of OOA
- Improve, optimize more
  - Design the Methods (operations), Attributes, Relationship between classes
- Generate static and dynamic charts
  - Static: denote classes and objects
  - Dynamic: represents interactions between classes & activity methods

2.4. Unified Modeling Language (UML)

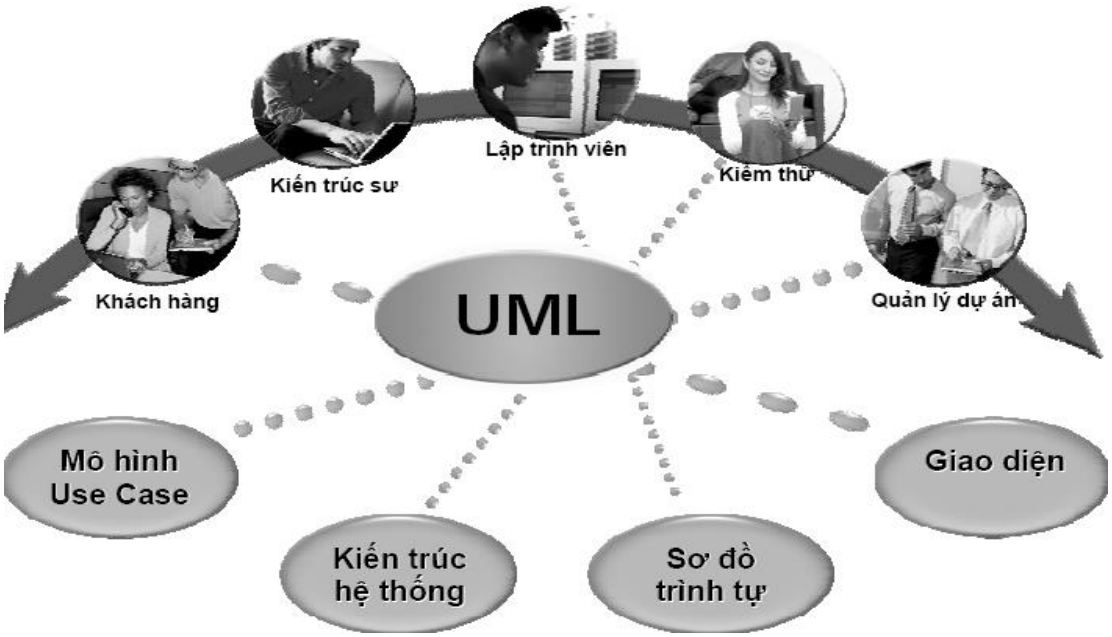
2.4. Unified Modeling Language UML

- UML : “Unified Modeling Language”
- UML is a visual language for :
  - visualizing
  - specifying
  - constructing
  - documenting

  the components of a software system
- The development work will be handled consistently, reducing errors
- Makes it easier to visualize the structure of the system
- More effective in communication and exchange

2.4. Unified Modeling Language UML

- Establish a unified way for building and “drawing” requirements following an object-oriented design during software development.



Working with UML

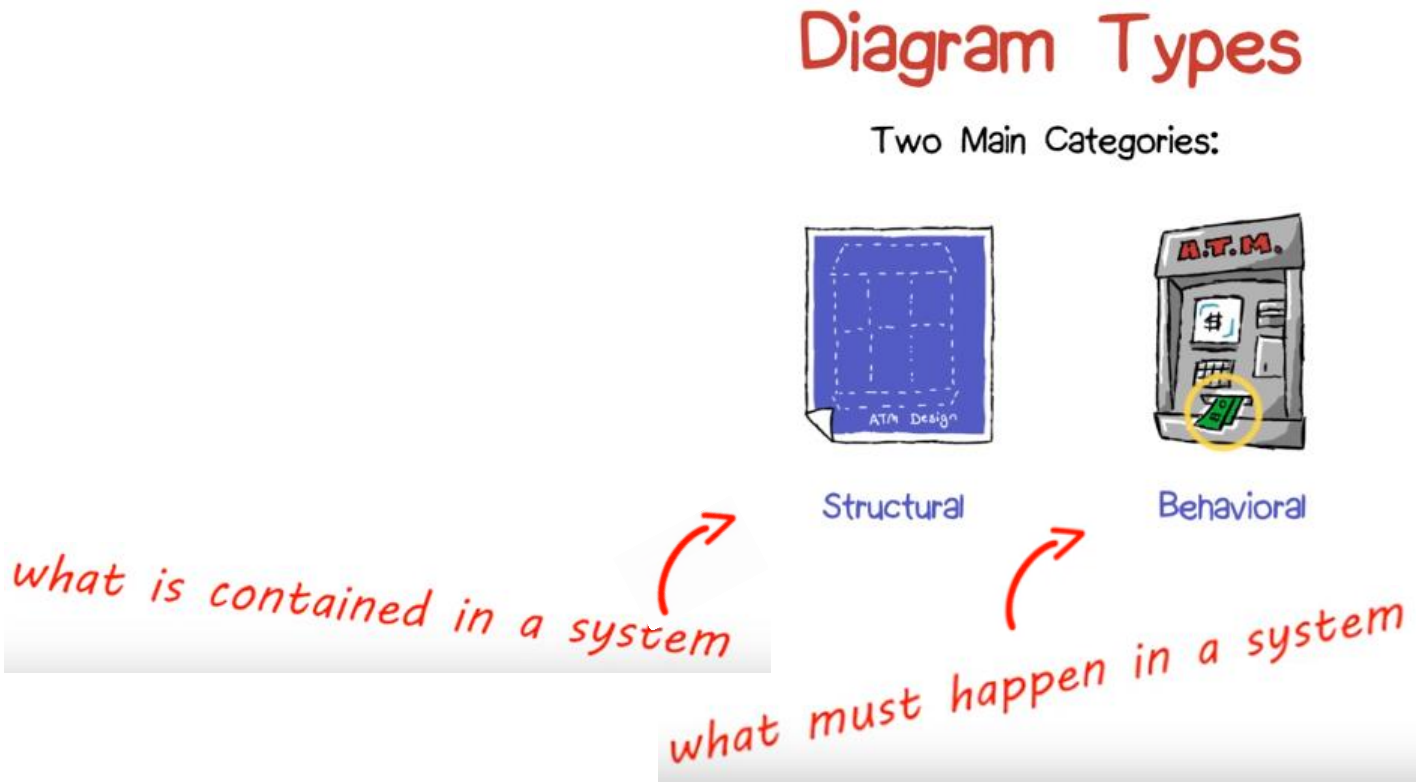
- UML models can be connected directly to a wide variety of programming languages.
  - Mapping to Java, C++, Visual Basic…
  - Tables in RDBMS or repository in OODBMS
- UML tools
  - Open source tools: EclipseUML, UmlDesigner, **StarUML**, Argo UML ...
  - Commercial tools: Enterprise Architect, IBM Rational Software Architect, Microsoft Visio, Visual Paradigm for UML, SmartDraw...

Basic UML Diagrams

- Diagrams:
  - The drawings include the modeling element symbols
  - illustrates a particular component or aspect of the system.
- A system model usually has many types of diagrams, each of which includes many different diagrams.
- A diagram is a made from a particular view
- Certain types of diagrams can be part of many different views
- UML 2nd generation has up to 13-14 diagram types.
- In a project, use only the most relevant diagrams

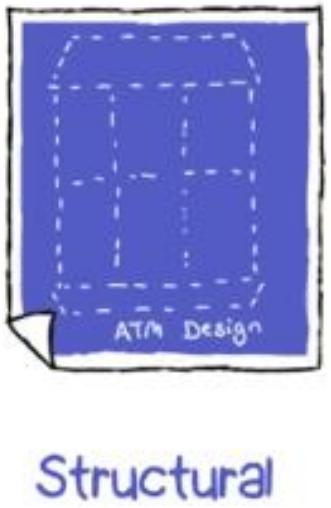
Basic UML Diagrams

- Distinguish:
  - Structural Diagram: describes the static, permanent components of the system and the relationships between them
  - Behavioral Diagram: describes how the system works.



Structural Diagram

- Static Structural Diagrams:
  - Class Diagram
  - Object Diagram
  - Package diagram
- Execution diagrams:
  - Component Diagram
  - Deployment Diagram
  - Composite Diagram
- Profile Diagram

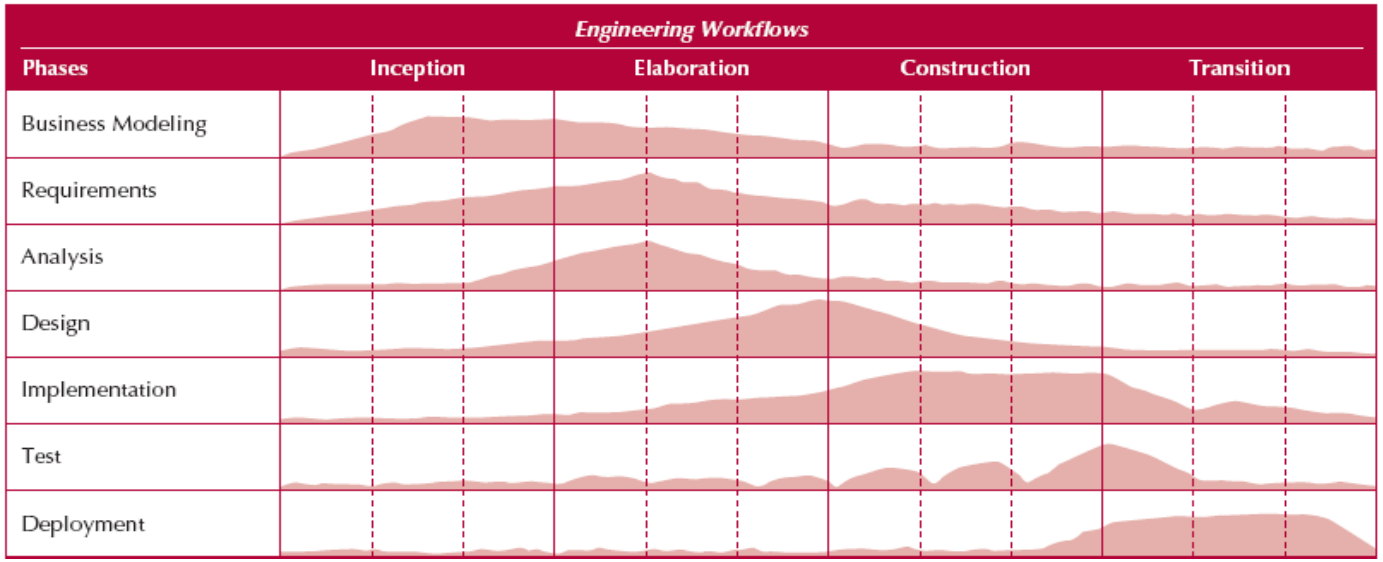


Behavioral diagrams

- Use Case Diagram
- Activity Diagram
- Interaction diagrams
  - Interaction overview diagram
  - Sequence Diagram
  - Communication/Collaboration Diagram
  - Timing Diagram
- State Machine Diagram

Unified process

- A system development methodology that prescribes when to use UML techniques and how to use them during system analysis and design.
  - (not a complete software development process)



Booch, Jacobsen, Rumbaugh