Module 2:
Class and object concepts and declaration

Dr. Tran Minh Triet

Acknowledgement

- Slides
  - Course CS202: Programming Systems
    Instructor: MSc. Karla Fant,
    Portland State University
  - Course CS202: Programming Systems
    Instructor: Dr. Dinh Ba Tien,
    University of Science, VNU-HCMC
  - Course DEV275: Essentials of Visual Modeling with UML 2.0
    IBM Software Group

Outline

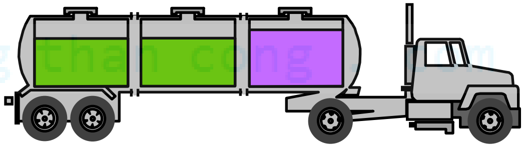
- What is an object?
- What is a class?
- OO Design
- Class identifying
- Class declaration

What is an object?

What Is an Object?

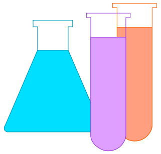
- Informally, an object represents an entity, either physical, conceptual, or software.

  - Physical entity



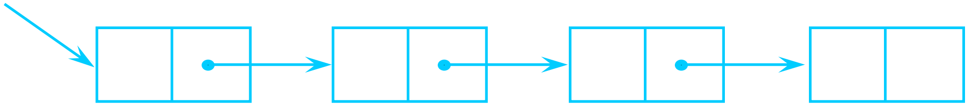
Truck

  - Conceptual entity



Chemical Process

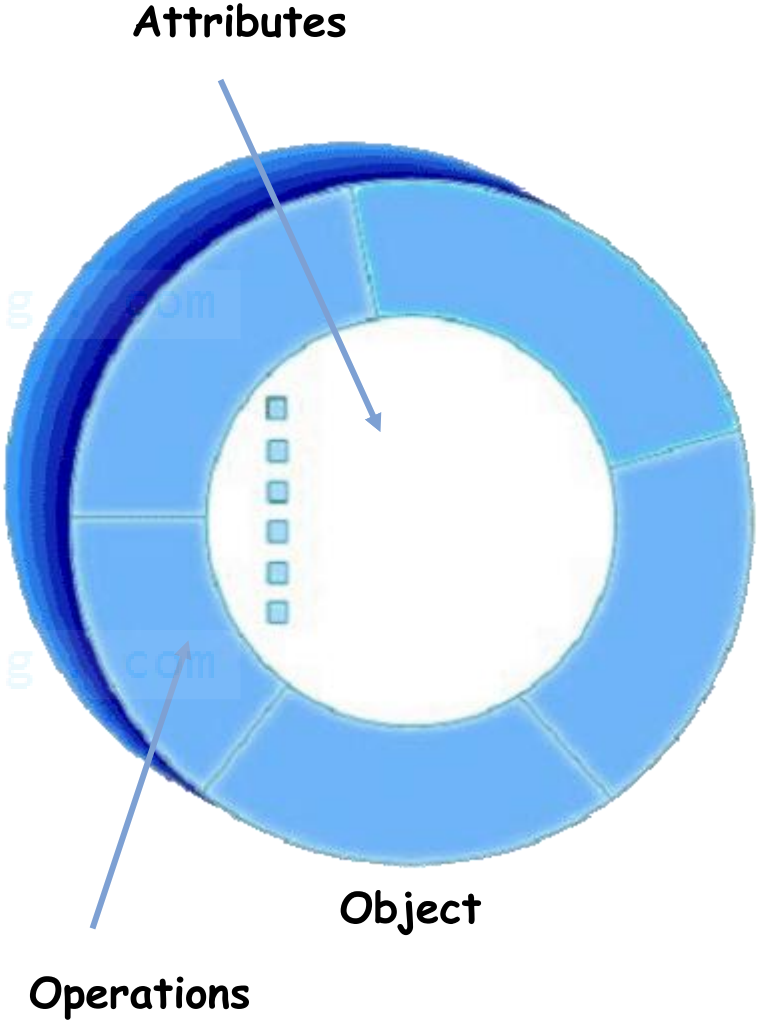
  - Software entity



Linked List

A More Formal Definition

- An object is an entity with a well-defined boundary and *identity* that encapsulates *state* and *behavior*.
  - State is represented by attributes and relationships.
  - Behavior is represented by operations, methods, and state machines.

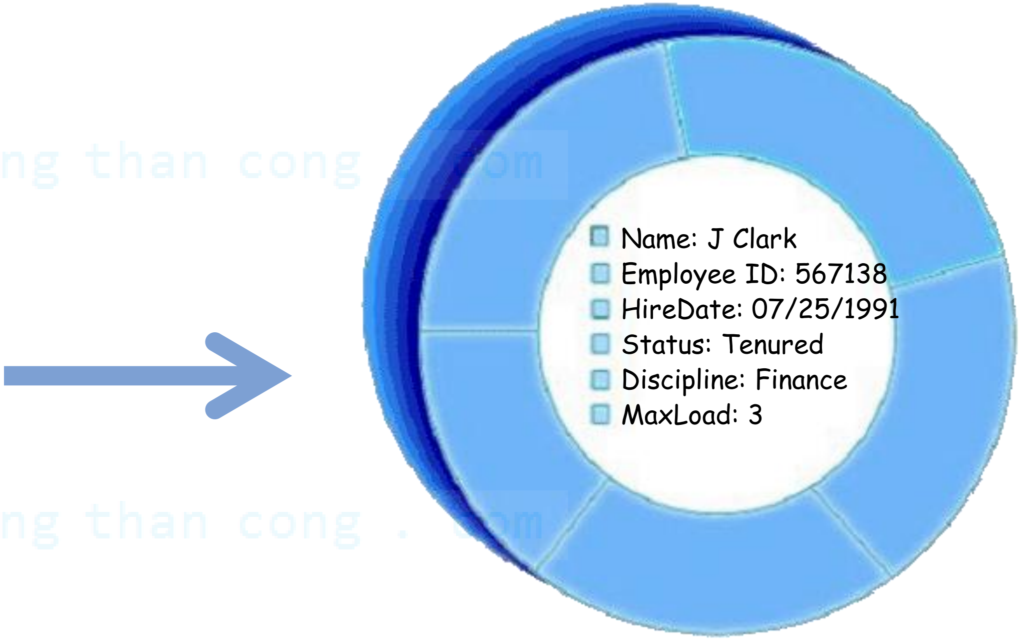


An Object Has State

- State is a condition or situation during the life of an object, which satisfies some condition, performs some activity, or waits for some event.
- The state of an object normally changes over time.

An Object Has State

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



- Name: J Clark
- Employee ID: 567138
- HireDate: 07/25/1991
- Status: Tenured
- Discipline: Finance
- MaxLoad: 3

Professor Clark

An Object Has Behavior

- Behavior determines how an object acts and reacts.
- The visible behavior of an object is modeled by a set of messages it can respond to (operations that the object can perform).

An Object Has Behavior

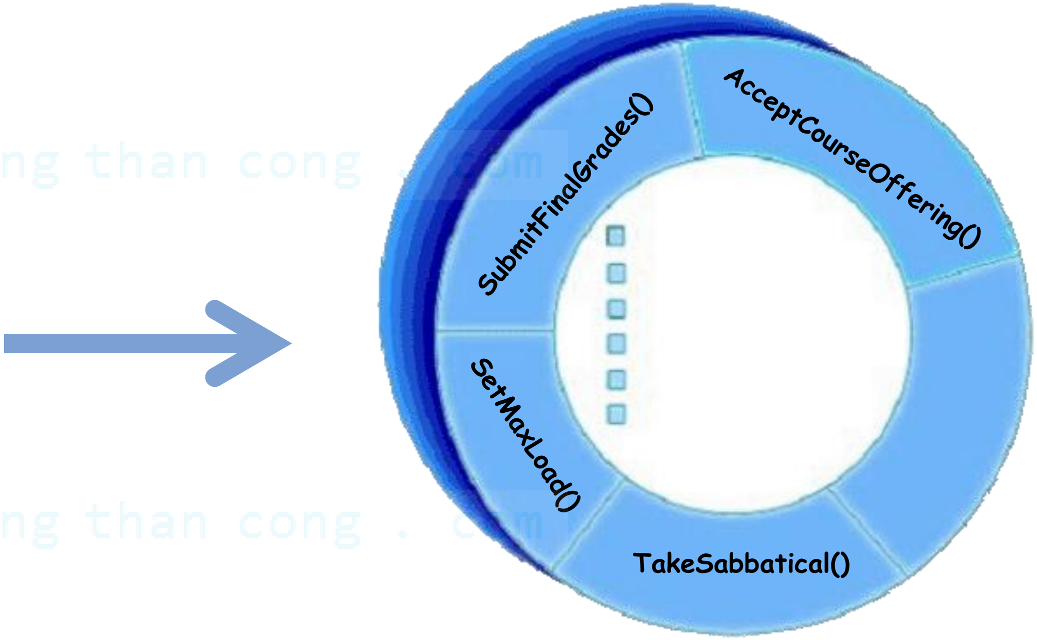
Professor Clark's behavior

Submit Final Grades

Accept Course Offering

Take Sabbatical

Set Max Load



Professor Clark

An Object Has Identity

- Each object has a unique identity, even if the state is identical to that of another object.

Professor "J Clark" teaches Biology

Professor "J Clark" teaches Biology

What is a class?

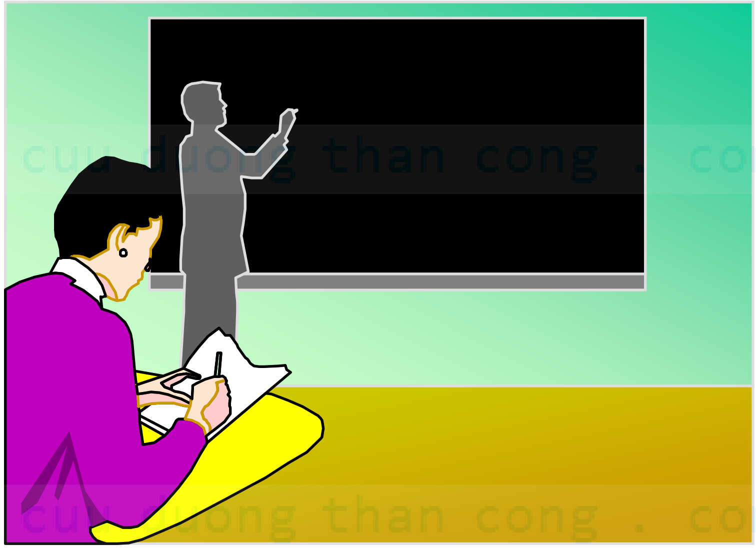
What Is a Class?

- A class is a description of a set of objects that share the same *attributes*, *operations*, *relationships*, and *semantics*.
  - An object is an instance of a class.
- A class is an abstraction in that it
  - Emphasizes relevant characteristics.
  - Suppresses other characteristics.

A Sample Class

Class
Course

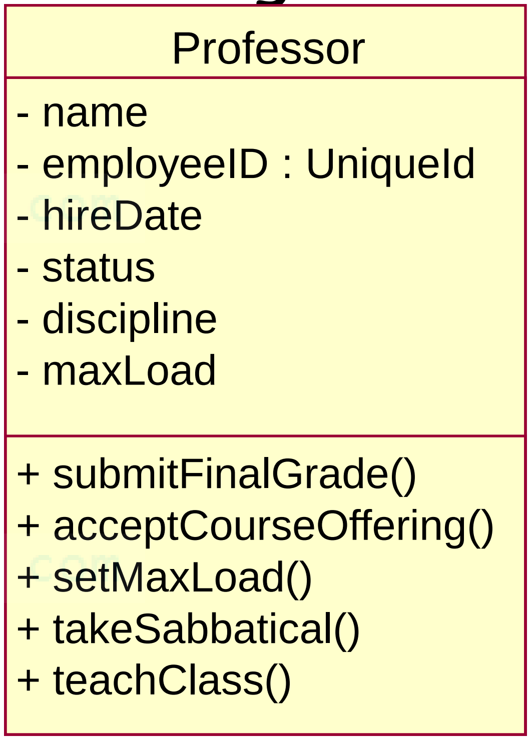
Properties
Name
Location
Days offered
Credit hours
Start time
End time



Behavior
Add a student
Delete a student
Get course roster
Determine if it is full

Representing Classes in the UML

- A class is represented using a rectangle with three compartments:
  - The class name
  - The structure (attributes)
  - The behavior (operations)



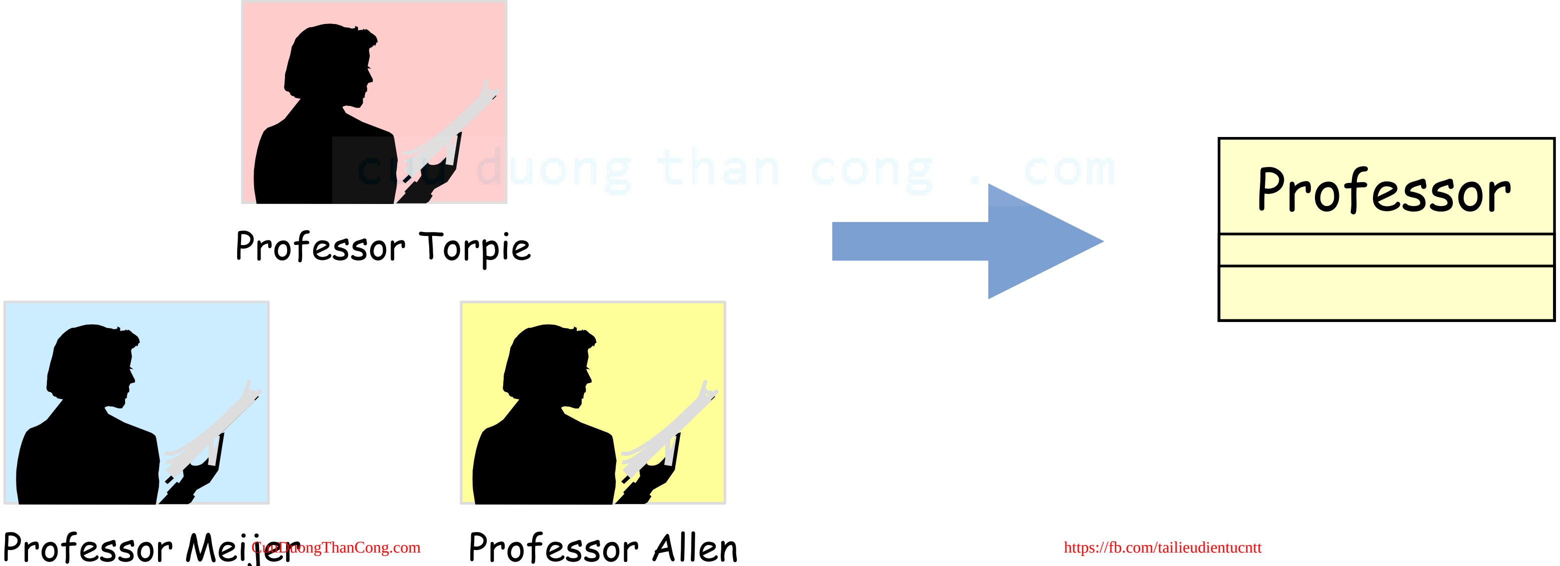
Professor

- name
- employeeID : UniqueId
- hireDate
- status
- discipline
- maxLoad

+ submitFinalGrade()
+ acceptCourseOffering()
+ setMaxLoad()
+ takeSabbatical()
+ teachClass()

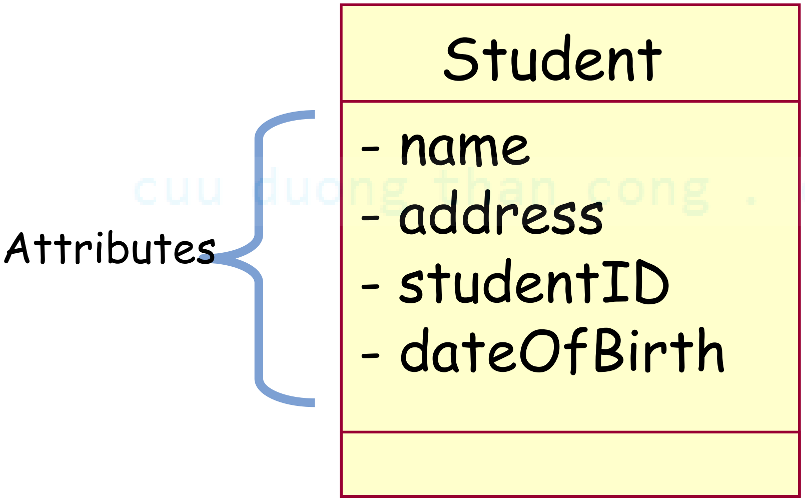
The Relationship between Classes and Objects

- A class is an abstract definition of an object.
  - It defines the structure and behavior of each object in the class.
  - It serves as a template for creating objects.
- Classes are not collections of objects.

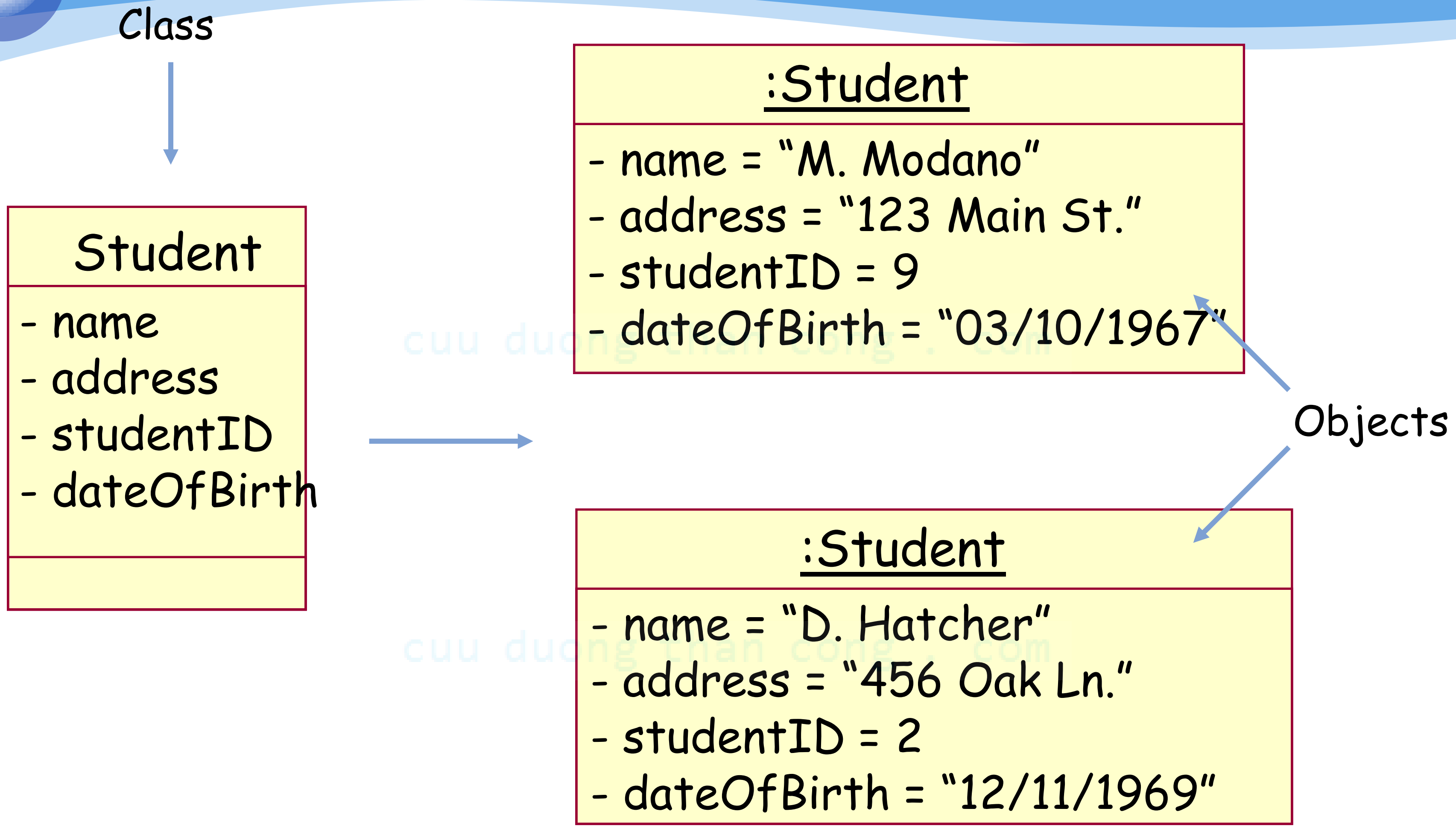


What Is an Attribute?

- An attribute is a named property of a class that describes the range of values that instances of the property may hold.
  - A class may have any number of attributes or no attributes at all.



Attributes in Classes and Objects



Class

Student

- name
- address
- studentID
- dateOfBirth

Objects

:Student

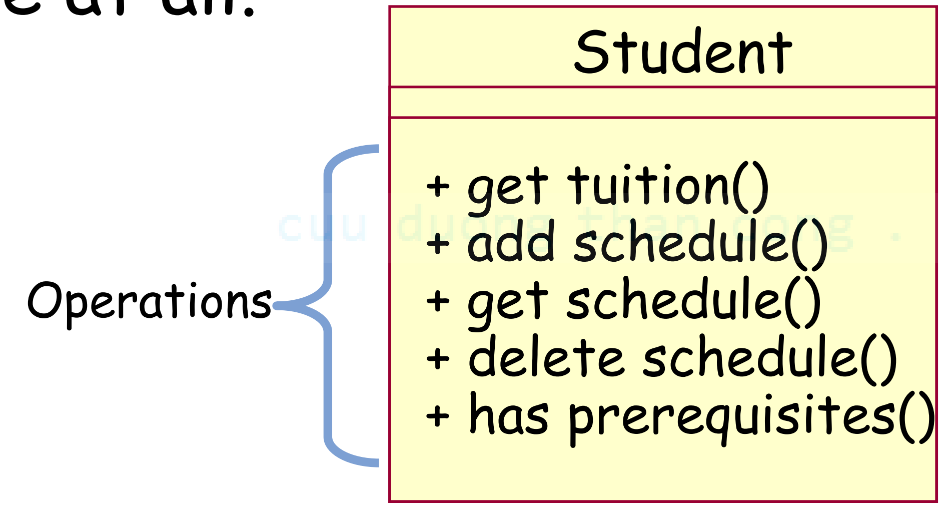
- name = “M. Modano”
- address = “123 Main St.”
- studentID = 9
- dateOfBirth = “03/10/1967”

:Student

- name = “D. Hatcher”
- address = “456 Oak Ln.”
- studentID = 2
- dateOfBirth = “12/11/1969”

What Is an Operation?

- A service that can be requested from an object to effect behavior. An operation has a signature, which may restrict the actual parameters that are possible.
- A class may have any number of operations or none at all.



Student

+ get tuition()
+ add schedule()
+ get schedule()
+ delete schedule()
+ has prerequisites()

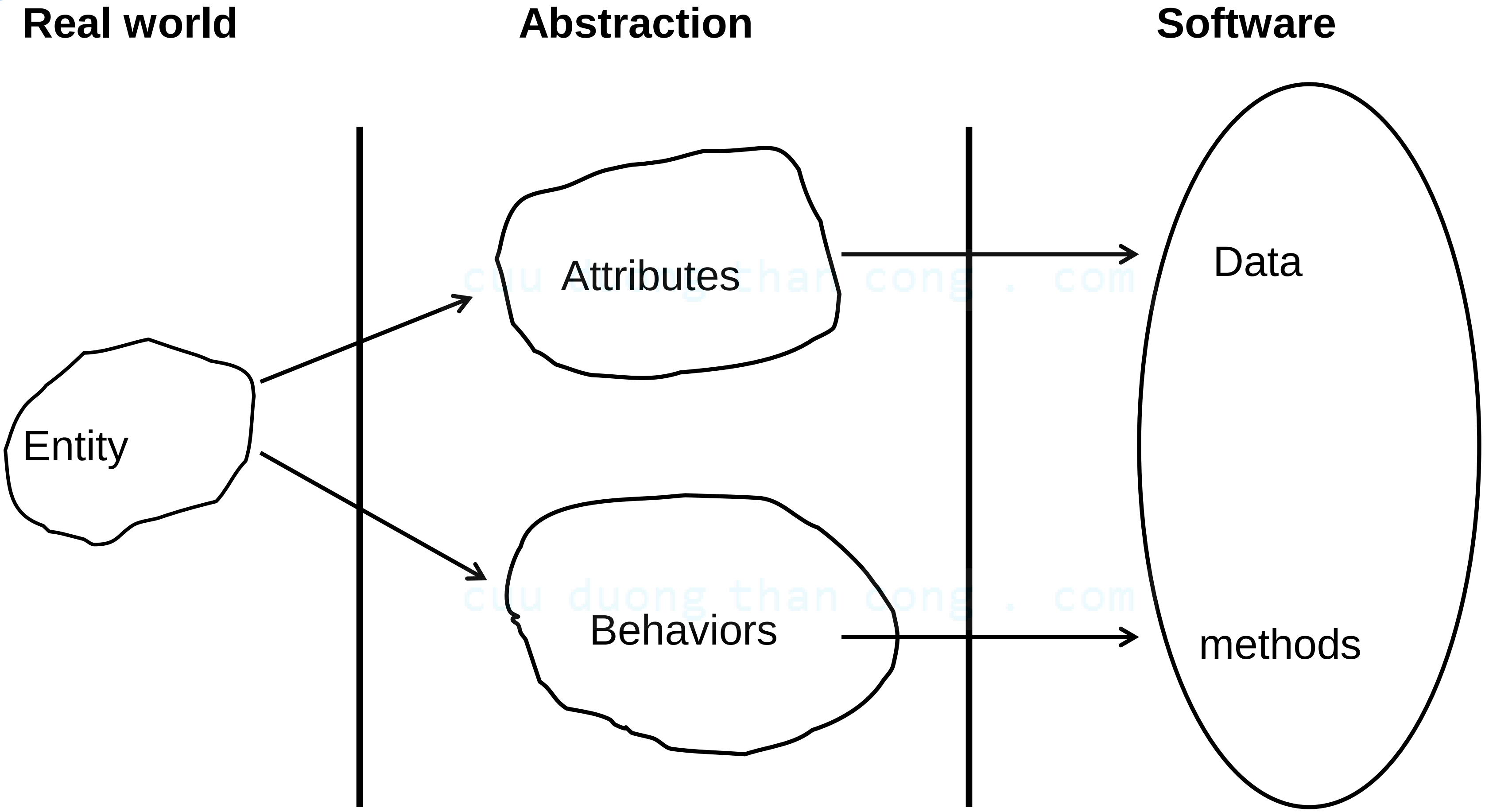
Operations

OO Design

Object-oriented design

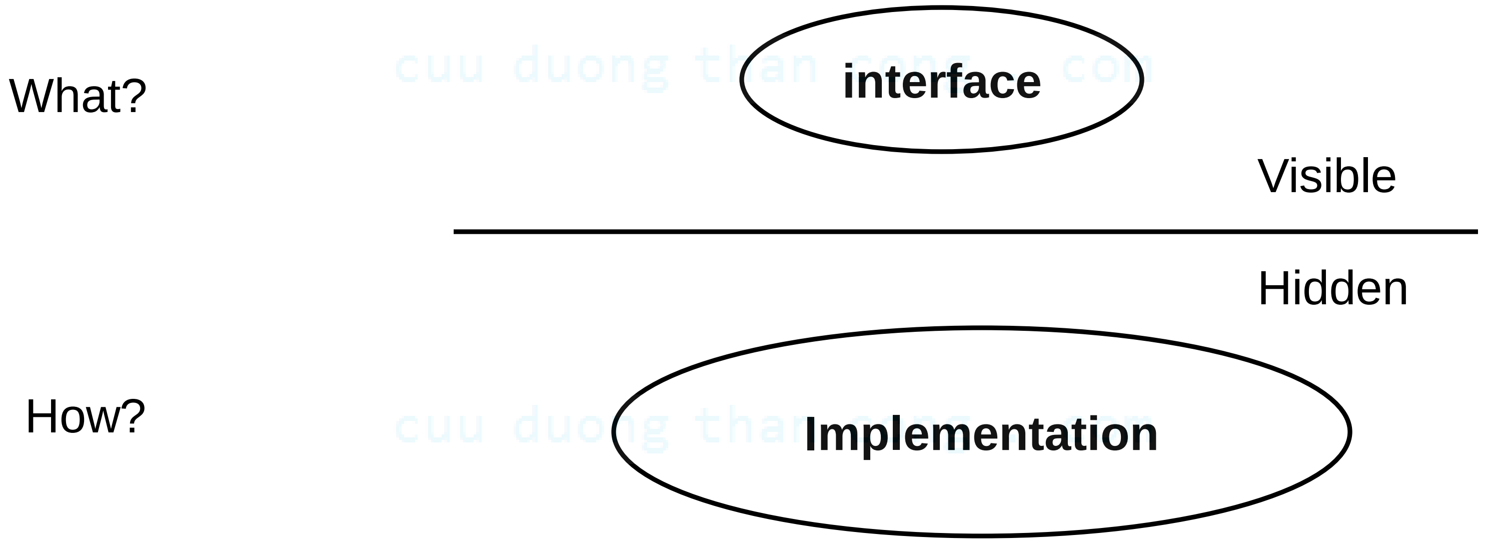
- Abstract Data Types (ADT)
- Divide project into a set of cooperating classes
- Each class has a very specific functionality
- Think of a class as similar to a data type
- Class can be used to create instances of objects

Mapping the real world to software



Classes in OO Programming

- Separation interface from implementation



What?

interface

Visible

Hidden

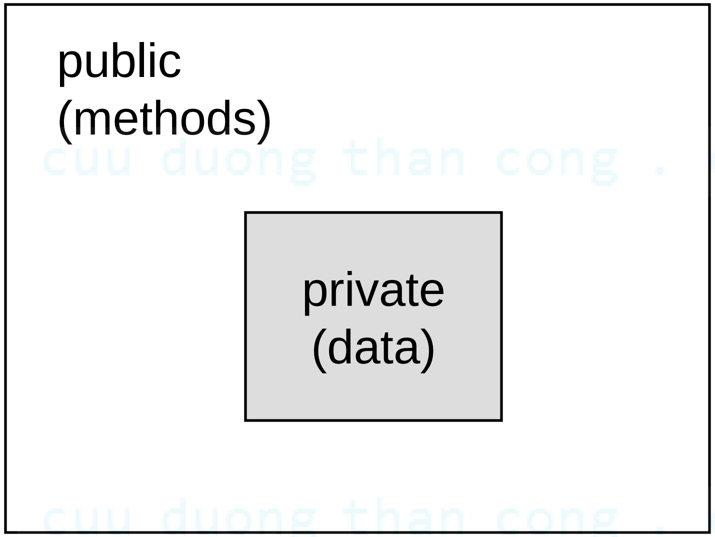
How?

Implementation

Structure of a class

- A class models an entity in real world
- A class represents all members of a group of objects
- A class provides a public interface and a private implementation
- Hiding the data and “algorithm” from the user

Structure of a class



class

Class Identifying

Designing process

- Identifying classes
- Identifying behaviors
  - Decide whether behavior is accomplised by a single class or through the collaboration of a number of "related" classes
  - Static behavior: behavior always exists
  - Dynamic behavior: depending of when/how a behavior is invoked, it might or might not be legal

Identifying classes

- Abbott and Booch:
  - use nouns, pronouns, noun phrases to identify objects and classes
  - Note: not all nouns are really going to relate to objects
- Coad and Yourdon:
  - identify individual or group "things" in the system/problem
- Ross: common object categories: people, places, things, organizations, concepts, events

Example:

- Game "Tetris":
  - possible classes:
    - Board,
    - Block (square block),
    - Piece (composed of several blocks),
    - Player (is it necessary?),
    - Line of Blocks

Example:

- e-Shopping Website
  - possible classes:
    - Product
      - Attributes: Name, ID, price, status, manufacturer's name, images, technical description.
    - Product Category:
      - Attributes: Name
    - Manufacturer
      - Attributes: Name, Country, Website

Example

- Website “National Foolball Competition”
  - People: Player, Referee, Coach, Team Manager…
  - Places: Stadium, City…
  - Things: Ball (is it necessary?)
  - Organizations: Team, National Football Association
  - Concepts: Half, Round, Season…
  - Events: Match (is this a concept or an event?), Goal

Class

- A class should:
  - be a real-world entity
  - be important to the discussion of the requirements
  - have a crisply defined boundary
  - make sense; (i.e. can identify the attributes and behaviors)
  - closely related

Object

- An “object” is an instance of a class
  - Just like a “variable” is an instance of a specific data type
- We can zero or more variables (or objects) in our programs

```
/* DataType          Variable*/
   int               x;
   Fraction          f;
```

Class and object

- A class is a blueprint for an object.
- When you instantiate an object, you use a class as the basis for how the object is built.
- A class can be thought of as a sort of higher-level data type. For example:

```
myClass myObject;
```

Class and object

- Each object has its own attributes and behaviours .
- A class defines the attributes and behaviours that all objects created with this class will possess.

- Classes are pieces of code.
- Objects are created from classes,

Class declaration

Class declaration in C++

```
class  <Name of the class>
{
        public:
                <public attributes and methods>
        private:
                <private attributes and methods>
};
```

Scope

- **private**: only visible to methods of the class itself.
- **public**: can be use from inside of the class or any client outside

An example

```
class  CDate
{
    public:
        CDate();
        CDate(int iNewDay, int iNewMonth, int iNewYear);
        int     getDay();       // return day
        int     getMonth();     // return month
        int     getYear();      // return year
        ...
    private:
        int     m_iDay, m_iMonth, m_iYear;
};
```

Scope resolution operator ::

- Tell the compiler the method or attribute belongs to a certain object

For example:

```
CDate::getDay()
CDate::getMonth()
```

Separation declaration from definition

```
//keep in 1 file
class  CDate
{
   public:
          int   getDay();

  private:
          ...
};

int   CDate::getDay()
{
          return m_iDay;
}
```

```
// header file .h
class  CDate
{
   public:
          int   getDay();
  private:
          ...
};

// implementation file .cpp
int   CDate::getDay()
{
          return m_iDay;
}
```

How to use the Date class

```
int main()
{
    CDate today(20, 10, 2008);
    CDate tomorrow, someDay;

    //can I do this?
    cout << today.m_iMonth; //!!!
    //how about
    cout << today.getMonth();
    ...
}
```

Encapsulation and data hiding

- Encapsulation:
  - A C++ class provides a mechanism for packaging data and the operations that may be performed on that data into a single entity
- Information Hiding
  - A C++ class provides a mechanism for specifying access

Taxonomy of member functions

- The types of member functions may be classified in a number of ways. A common taxonomy:
  - **Constructor/Initalization**: an operation that creates a new instance of a class
  - **Observer**: an operation that reports the state of the data members (aka Accessors, Getters)
  - **Mutator**: an operation that changes the state of the data members of an object
  - **Iterator**: an operation that allows processing of all the components of a data structure sequentially

Taxonomy of member functions

- **Constructor/Initalization**: an operation that creates/initalize a new instance of a class
  - Constraint Checking methods?

Taxonomy of member functions

- **Observer**: an operation that reports the state of the data members
  - Provides value of an internal attribute
  - Provides some value calculated from internal attributes only
  - Provides some value calculated from internal attributes AND some external parameter(s)

Taxonomy of member functions

- **Mutator**: an operation that changes the state of the data members of an object
  - Updates value of an internal attribute
  - Transforms values of internal attributes
  - Constraint Checking methods?

Taxonomy of member functions

- **Iterator**: an operation that allows processing of all the components of a data structure sequentially

Exercises

- List member functions of the following classes:
  - Date
  - Fraction with numerator and denominator
  - Employee