CS202: Programming Systems

## Week 1_3: OO Design

10/2018

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

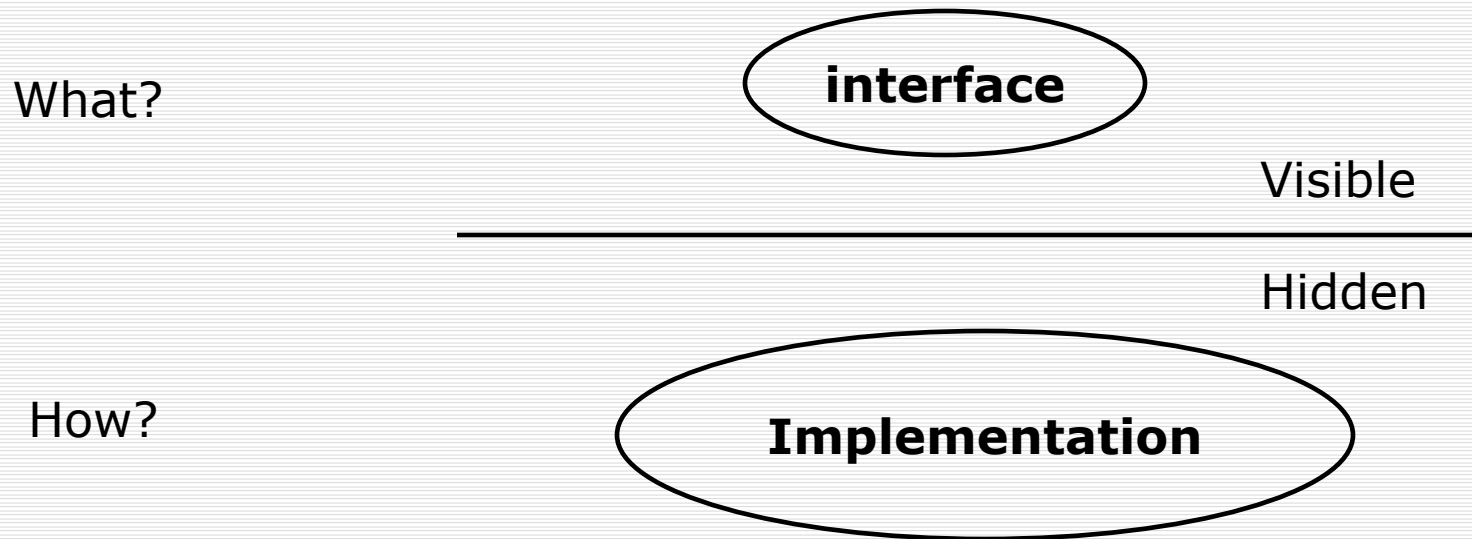# Mapping the real world to software

Real world　　　　**Abstraction**　　　　**Software**

Entity

Attributes → Data

Behaviors → methods
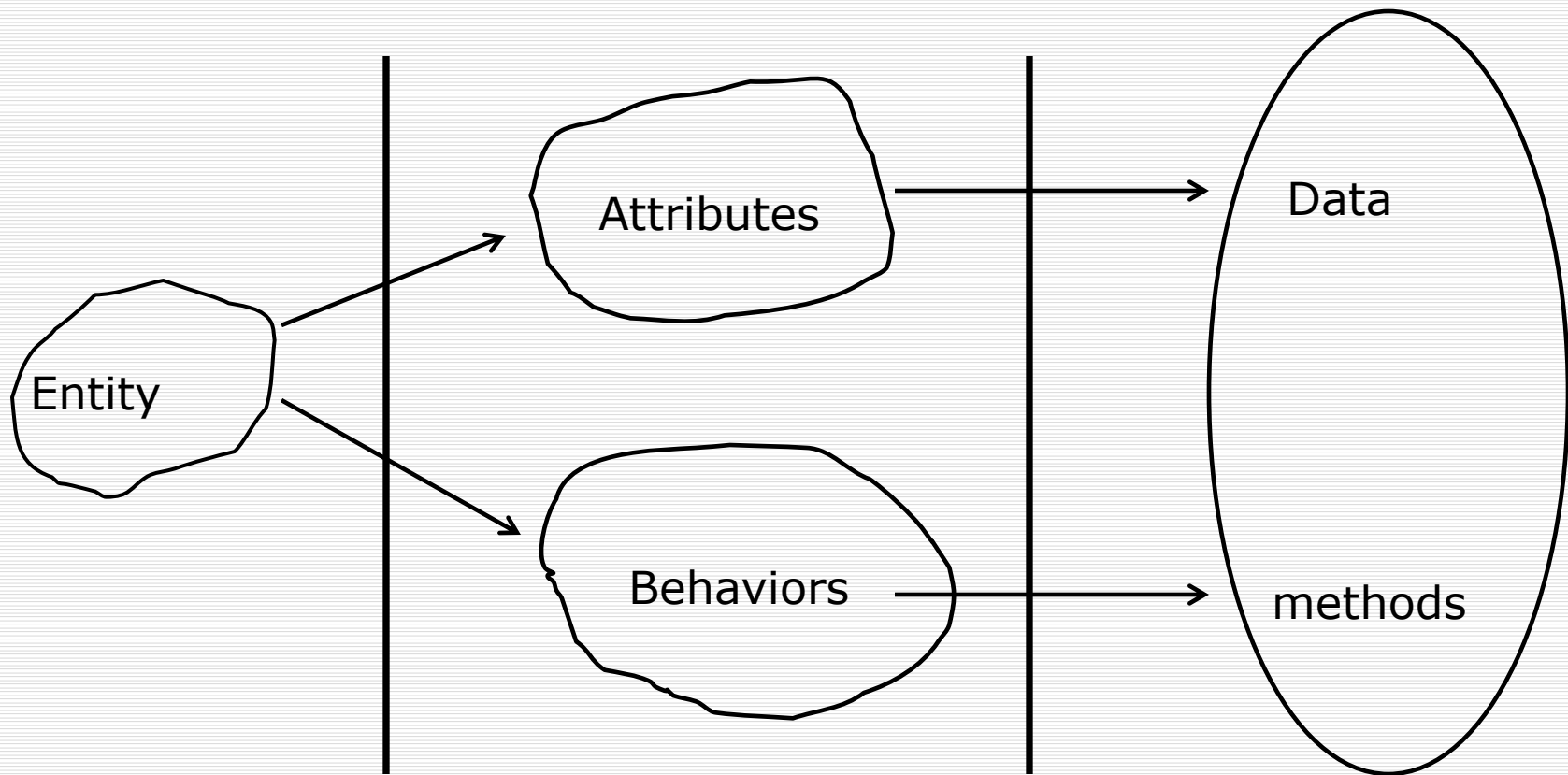
Classes in OO Programming

☐ 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
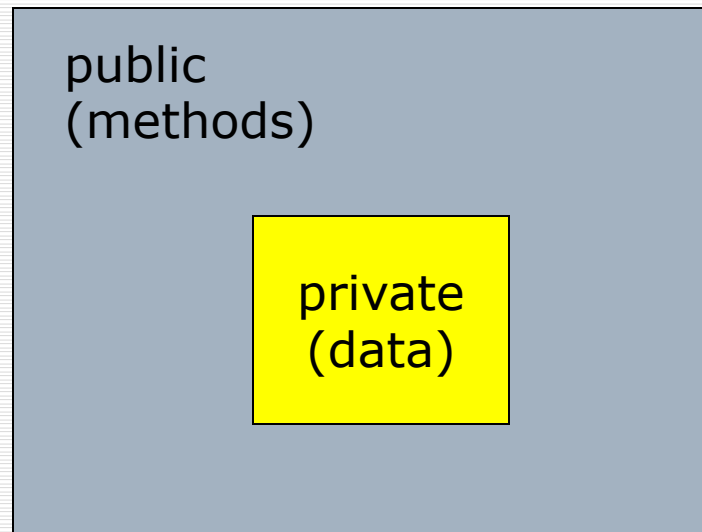
class

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

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

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 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  Date
{
    public:
        Date();
        Date(int iNewDay, int iNewMonth, int iNewYear);
        void moveDays(int k); //move k days
        int compare(const Date& anotherDate);
        ...
    private:
        int   iDay, iMonth, iYear;
};
```

Scope resolution operator ::

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

For example:

```
void Date::moveDays(int k)
{

}
```

Separation declaration from definition

```
//keep in 1 file
class  Date
{
    public:
        void moveDays(int k);

    private:
        ...
};


void Date::moveDays(int k)
{
  //...
}
```

```
// header file .h
class  Date
{
    public:
        void moveDays(int k);
    private:
        ...
};


// implementation file .cpp
void Date::moveDays(int k)
{
        return iDay;
}
```

How to use the `Date` class

```cpp
int main()
{

    Date today(20, 10, 2008);

    Date tomorrow, someDay;


    //can I do this?

    cout << today.iMonth;

    //how about this?

    today.moveDays(5);

    ...

}
```

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:** an operation that creates a new instance of a class

■ **Mutator:** an operation that changes the state of of the data members of an object

■ **Observer:** an operation that reports the state of the data members (aka Accessors, Getters)

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

Exercises

☐ List member functions of the following classes:

■ Fraction with numerator and denominator

■ Employee