Object Construction and Usage

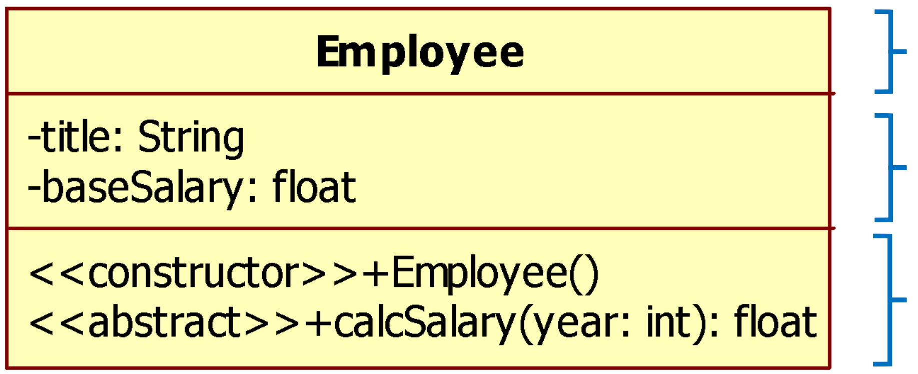
Nguyen Van Vu
nvu@fit.hcmus.edu.vn

Topics

- Describe classes in UML
- Define classes
- Initiate and use objects
- Access modifiers/scope

Class

- Defines the set of common objects that have same the same attributes, operations, relationships, and semantics
- Represents a thing
- Notation



Employee
-title: String
-baseSalary: float
<<constructor>>+Employee()
<<abstract>>+calcSalary(year: int): float

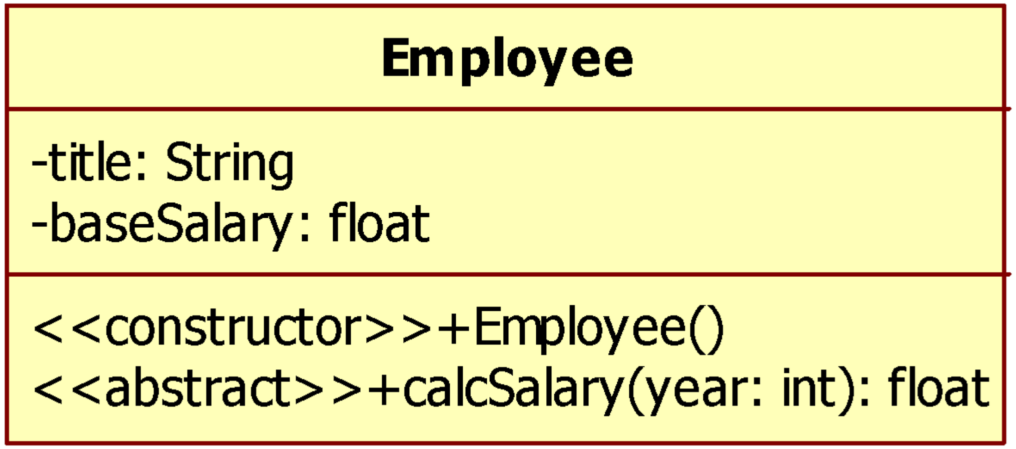
Name: must be unique within its group

Attributes

Operations

Attribute

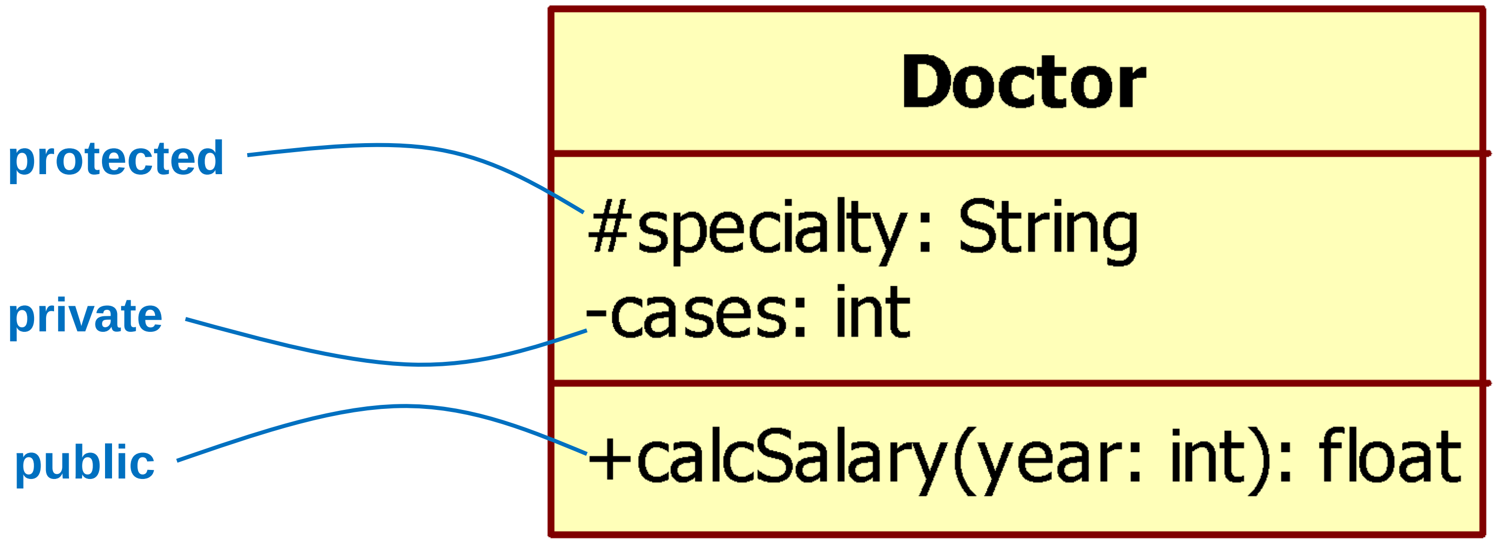
- Defines data that characterize a class
- An abstraction of the kind of data or object
  - *title* is an attribute of the kind of *String* object
- Data type is specified by a semicolon ":"


Employee
-title: String
-baseSalary: float
<<constructor>>+Employee()
<<abstract>>+calcSalary(year: int): float

Title and *baseSalary* are two attributes of *String* and *float* data types, respectively

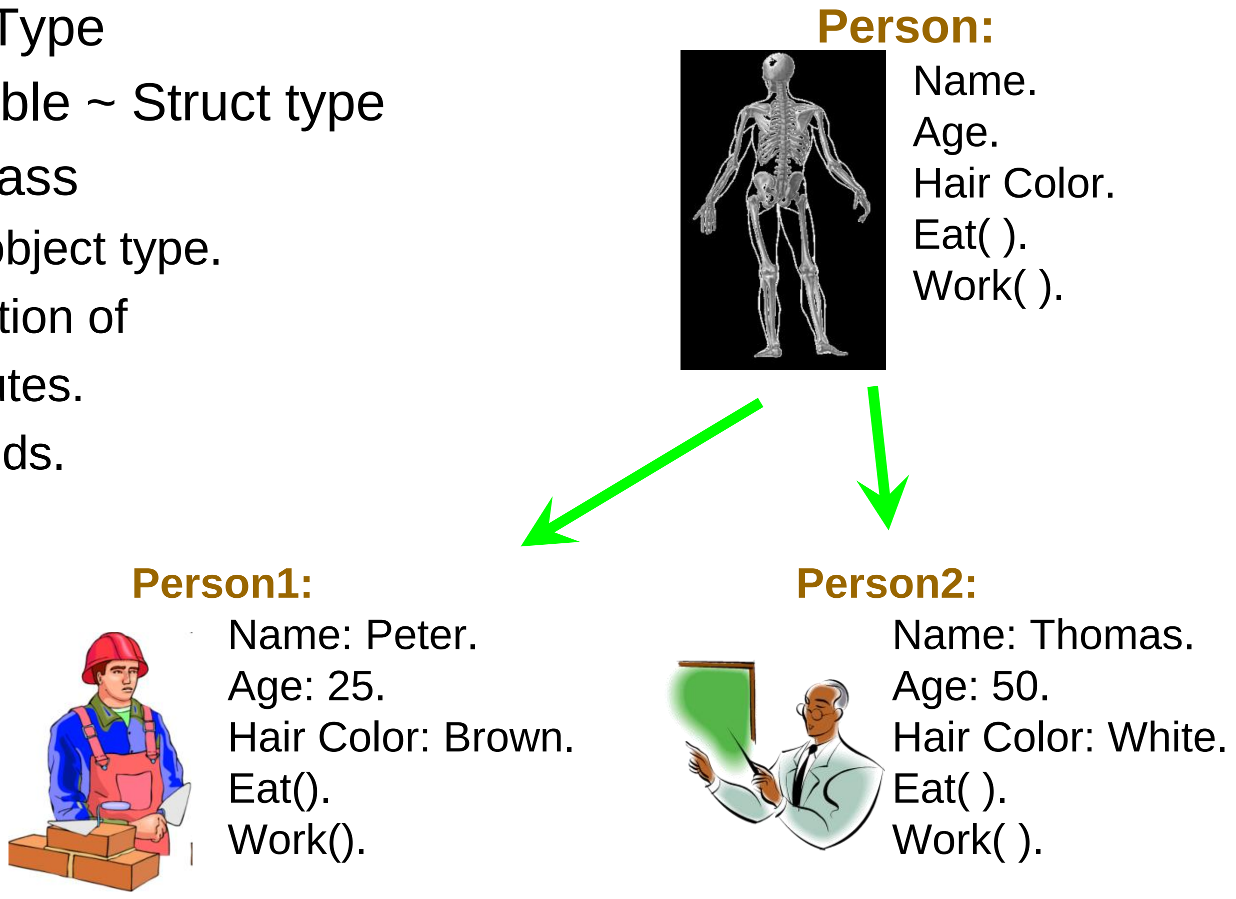
Operation

- An operation specifies a service that can be requested from objects of the class
- Attribute and operation visibility



Object and class

- Class concept
  - Variable ~ Type
  - Struct variable ~ Struct type
  - Object ~ Class
    - Class is object type.
    - A description of
      - Attributes.
      - Methods.



Define a class in C++

- Same as struct
- Usage
  - Declare class (file .h):

```
class <Class Name>
{
        <Attribtes>;
        <Methods>;
};
```

  - Implement class (file .cpp):
    - Implement methods same as functions
  - Create object from class (main( ) function):
    - Declare variables from class

* Adapted from Mr. Nguyễn Minh Huy's leccture 9

Product class

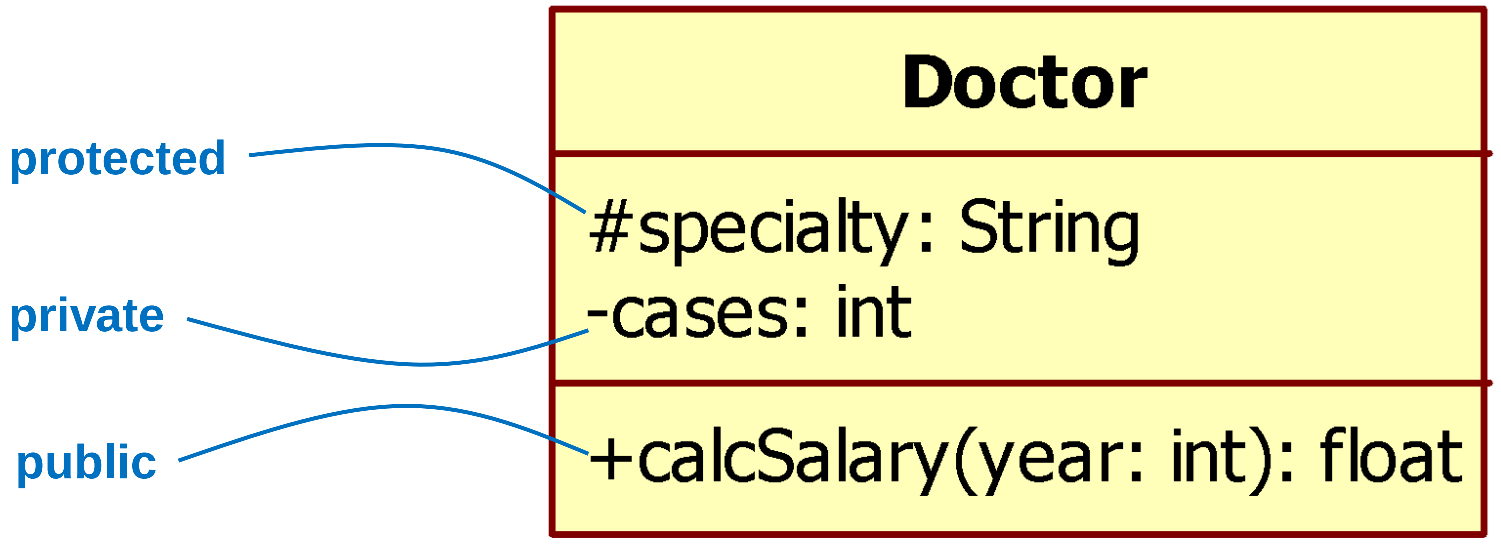
```
// Declare class, file Product.h
class Product
{
private:
        String   m_Name;
        String   m_Description;
        float   m_Weight;
        float   m_Price;
public:
        void setPrice(float newPrice);
};

// Implement class, file Product.cpp
void Product::setPrice(float newPrice)
{
        // …
}
```

Use Product class

```
// Use object, file main.cpp
void main()
{
        Product p1;
        Product p2;

        p1.setPrice(10.5);
}
```

Scope/Visibility

- Attribute and operation visibility



Scope

- Scope concept:
  - Working range:
    - Variable — Declared block
    - Function — Whole program
    - Struct members — Declared block of struct varaible
    - Object members — Can be control
- Scope control:

Keyword	Scope
private	Inside class only.
public	Inside and outside class.
protected	Inside class and children class.

Scope

- Example: private vs. public.

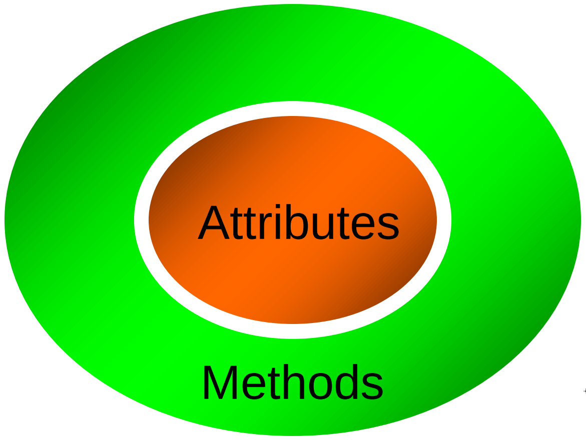
```
class A
{
private:
        int   x;
public:
        int   y;

public:
        int getX(  );
private:
        void calculate( );
};
```

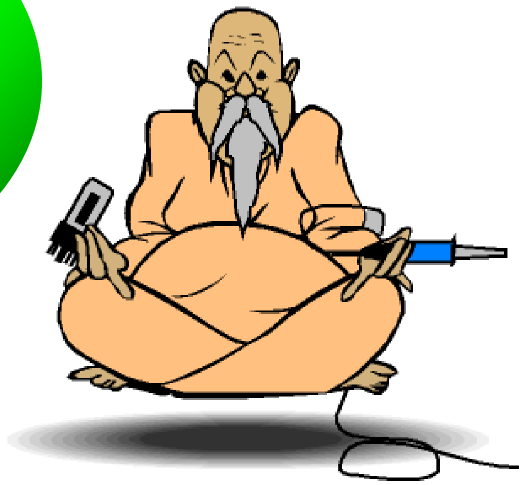
```
void main()
{
        A   obj;

        int   x = obj.x;         // Wrong
        obj.x = 1;               // Wrong

        int   y = obj.y;         // Right
        obj.y = 2;               // Right

        int   t = obj.getX( ); // Right
        obj.calculate( );      // Wrong
}
```

Scope

- General rule
  - Attributes: use **private** to hide inside
  - Methods: use **public** to provide functions

```
class Product
{
private:
    String   m_Name;
    float   m_Price;
public:
    void setPrice(float newPrice);
};
```





Constructor

- Defined to initiate an object
- Same name as class' name
- Called when an object is created
- Has no return type
- Can be overloaded

```
//  Declare class, file Product.h
class Product
{
private:
          String   m_Name;
          String   m_Description;
          float   m_Weight;
          float   m_Price;
public:
          Product(); //default constructor
          Product(string Name);
};
```

Copy constructor

- A special constructor used to create an object from another
  - Problem: how do you create a new product whose name and description are the same as another existing product?

```
//  Declare class, file Product.h
class Product
{

		public Product(const Product p) {} // copy constructor
}
```

- Memory leak problem:
  - Memory allocated to pointer must be deleted.

```
class Student
{
private:
        char *m_name;
};
```

```
void main()
{
        Student   s;
}
// Memory leak!!
```

  - Use delete method:

```
class Student
{
private:
        char *m_name;
public:
        void deleteMemory() {
                delete [ ]m_name;
        }
};
```

```
void main()
{
        Student   s;
        s.deleteMemory();
}
```

Forget to call?!

Destructor

- OOP has a better approach: using desctructor
- Used to **release** memory allocated to a pointer or any dynamic memory allocation (e.g., using *malloc*)
- Automatically called when an object is destroyed
- Only one destructor

```
class Student
{
private:
        char    *m_name;
public:
        ~Student() {
            delete [ ]m_name;
        }
};
```

```
void main()
{
        Student  s;
        Student  *p = new Student;
        delete p;
}
```

Static members

- Object sharing
  - Each object has its own
    - attributes
    - methods
    - Object members
  - How to share information among objects of the same class?
    - Using static members

Static members

- Static members
  - Class-level attributes and methods
  - Shared among objects of the same class

  - In C++
    - Keyword **“static”**
    - Initialization: outside class
    - Use **::** to access

```
class Fraction
{
private:
        static    int m_maxValue;
public:
        static    int getMaxValue();
};
```

```
int Fraction::m_maxValue = 10000;

void main()
{
    int   x = Fraction::getMaxValue();
}
```

Summary of concepts

- Class
- Object
- Operation and method
- Attribute/variable
- Constructor
- Copy constructor
- Destructor
- Static members

Practice 1 - MyString

- Create MyString class to work just like a string of characters
  - char *m_Data;
  - Int m_Length;
- Constructor/destructor
- Copy string
- Concatenate two strings

Practice 2

- Define a class named Fraction to represent and implement operations for a fractional number
- A factional number is defined by a numerator and denominator
- Define and implement constructors, a copy constructor
- Define and implement member functions to get the value of a fraction, plus and minus two fractions

Practice 3

- Minh is a student at the University of Science. This semester he is taking 5 courses. The school allows him to take up to 6 courses per semester.
- A student's attributes include name, student id
- A course has id, name, lecture total hour, practice total hour

- Let's implement the above situation using OOP in C++