Inheritance

SE271 Object-Oriented Programming (2020)

Yeseong Kim

Original slides from Prof. Shin at DGIST

Short Notice

- 수고하셨습니다
- The deadline for proposal is approaching
- Allowing HW2 late submission
  - Please understand if you already have done
  - Penalty per day: -30% points
    - Grade with the maximum score for all submission
    - I will download them every morning (7am)
  - I will upload the preliminary score after this class
    - Not checked code clone yet

Short Notice

- HW3 will be released on Wednesday (again?!) ☺
  - PLEASE comply the way-to-submit written in the instruction
  - PLEASE don't include main
  - Zip filename: HW3_학번.zip
    - Files in the zip file:
      - hw3.cpp hw3.h (O)
      - hw3_학번.cpp 학번_hw3.cpp (X)

Today’s Topic

- Inheritance
  - What is it?
  - Why/When do we need?
  - Syntax
    - Access Control
    - Constructor/Destructor
    - Casting
    - Multiple Inheritance

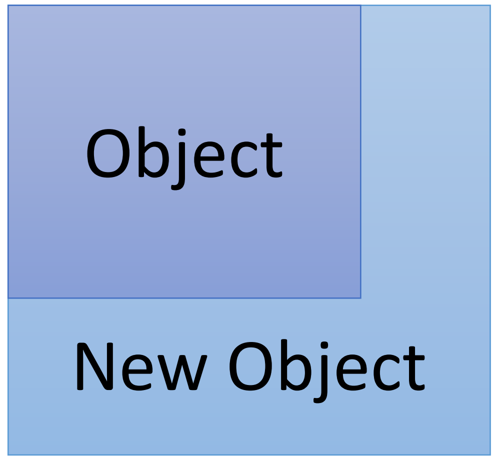
How to create Unicorn!

[Recap] Short Introduction to OOP

- OOP supports following concepts based on Objects
  - Abstraction
    - The process of generalization
  - Encapsulation
    - The process of keeping the details about how an object is implemented hidden away from users of the object
  - Polymorphism
    - The provision of a single interface to entities of different types
  - Inheritance

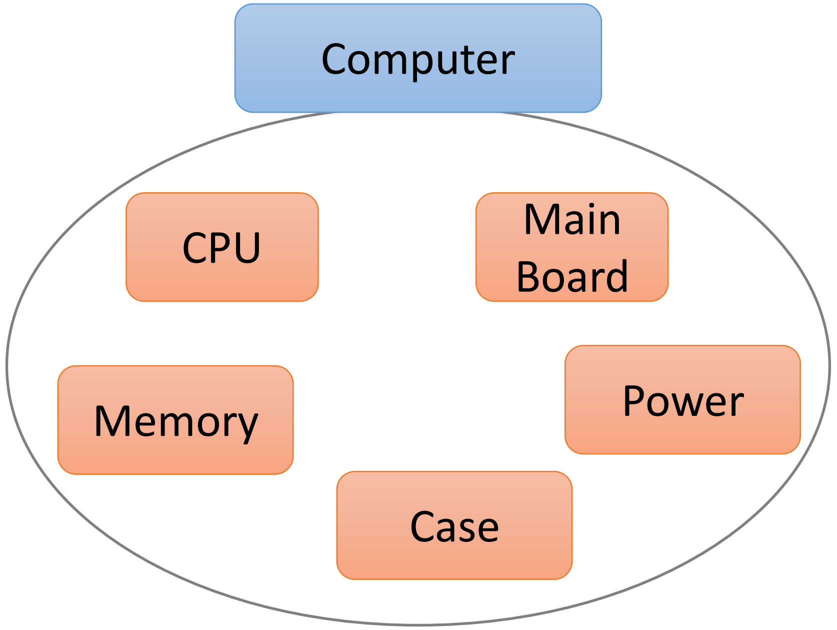
Inheritance

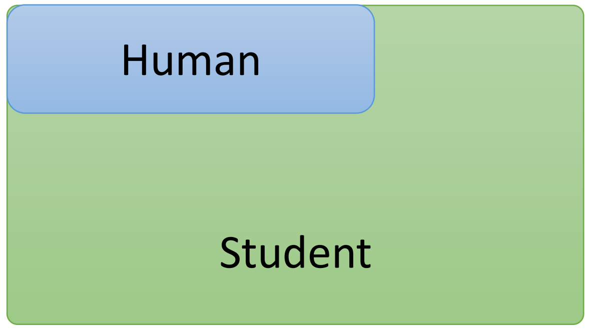
- What is it?
  - Creating new objects by directly **acquiring the attributes and behaviors** of other objects
  - Then extending or specializing them



Object Relationships

- Dependencies
- Association
- “has-a” relationship
  - Composition
  - Aggregation
- “is-a” relationship
  - Inheritance





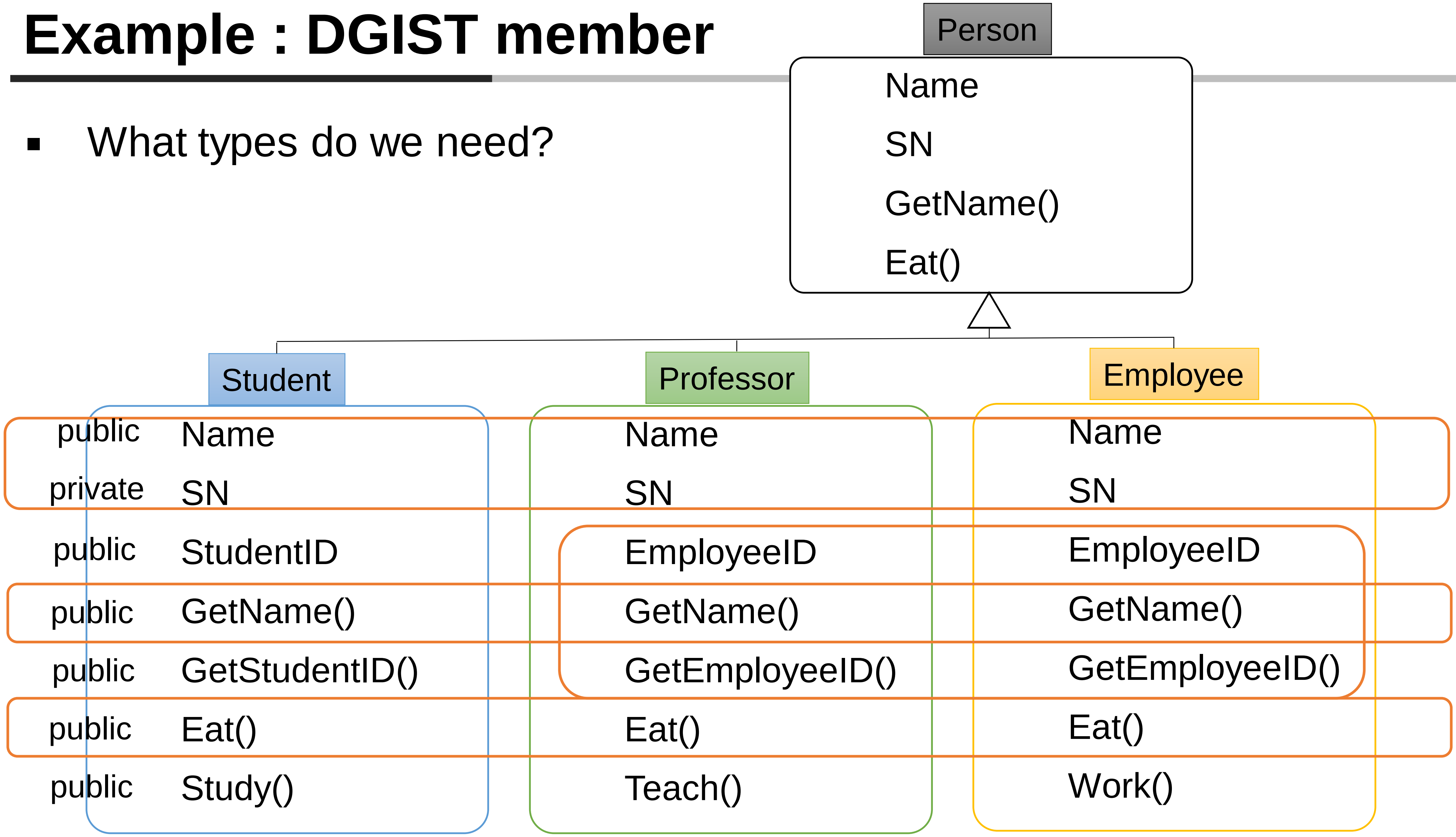
Inheritance

- Why do we need?
  - Reuse code(class)
  - Maintain hierarchical class structure
  - Need more specified objects

- When do we use?
  - 2 objects has “is-a” relationship
  - (possibly “has-a” relationship)

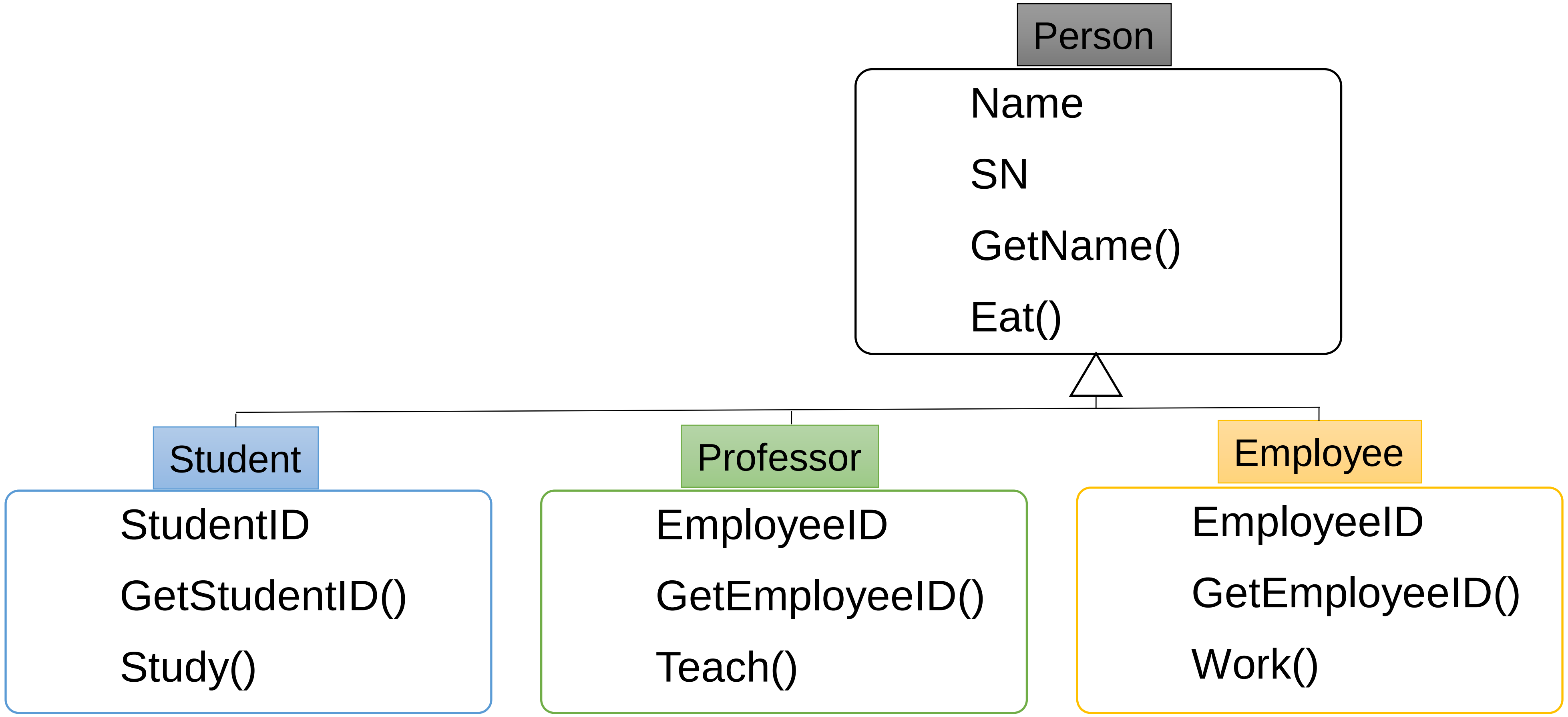
Example : DGIST member

- What types do we need?



Example : DGIST member

- What types do we need?



Inheritance

- Syntax

```
class Base {
    …
};
class Derived : inheritance_type Base {
    …
};
```

  - Inheritance type
    - public, protected, private

```
class Person {
    string name;
};
class Student : public Person {
    int studentID;
};


class Eat {
    void drink(Beverage b);
    void eat(Food f);
};
class Student : public Eat {
    int studentID;
};
```

Inheritance Type

- Access control
  - public, protected, private

```
class Base {
private:            // accessed by member method
    int private_var;
    void private_method;
protected:          // accessed by member method of derived class or itself
    int protected_var;
    void protected_method;
public:             // accessed by everyone
    int public_var;
    void public_method;
};
```

```
class Derived1 : private Base {
private:


};


class Derived2 : protected Base {
protected:


};


class Derived3 : public Base {
public:


};
```

Inheritance Type

Parent	Inherited with	Child
private	private	private
private	protected	private
private	public	private
protected	private	private
protected	protected	protected
protected	public	protected
public	private	private
public	protected	protected
public	public	public

Type Conversion(1)

- Up Casting

```
class Base {
};
class Derived : public Base {
};

Derived d;
Derived * pD = &d;

Base * pB = &d;   // ???
```

```
class Base {
protected:
    int baseVal=10;
    void BasePrint() {}
};
class Derived : public Base {
public:
    int a=10;
    void print() {}
};
Derived * pD = new Derived;
Base * pB = pD;
pD->print();
pB->print(); // ???
```

Type Conversion(2)

- Down Casting

```
class Base {
};
class Derived : public Base {
};

Derived d;

Base * pB = &d;
Derived * pD = (Derived *) pB;
```

```
class Base {
protected:
    int baseVal=10;
    void BasePrint() {}
};
class Derived : public Base {
public:
    int a=10;
    void print() {}
};
Derived * pD = new Derived;
Base * pB = pD;
pD->print();
pB->print(); // ???
((Derived *)pB)->print();
```

Inheritance - Constructor

- Constructor

```
class Base {
public:
    Base(){  cout << "base constructor\n"; }
};
class Derived : public Base {
public:
    Derived() {
        cout << "derived constructor\n";
    }
};
Derived d;                    What happens here?
```

```
class Base {
public:
    Base(int a){  cout << "base constructor\n"; }
};
class Derived : public Base {
public:
    Derived() {
        cout << "derived constructor\n";
    }
};
Derived d; // compile error
```

Implicit case:
When a Derived instance is created,
call the default constructor of a Base class

Explicit case:
When a Derived instance is created,
call the specified constructor of a Base class

Inheritance - Destructor

- Destructor

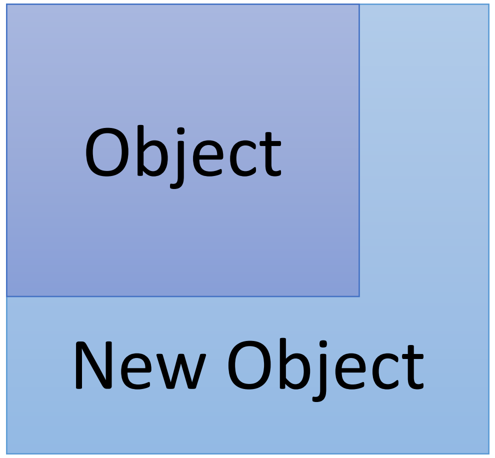
```
class Base {
public:
    ~Base(){ … }
};
class Derived : public Base {
public:
    ~Derived() { … }
};

Base* pB = new Derived ();
delete pB;                What happens here?
```

Order of Calls

- Order of constructor call, destructor call

```
class Base {
public:
    Base(){  cout << "1. base constructor\n"; }
    ~Base(){  cout << "2. base destructor\n"; }
};
class Derived : public Base {
    Derived() { cout << "3. derived constructor\n";   }
    ~Derived() { cout << "4. derived destructor\n";   }
};
Derived *pD = new Derived;
cout << "5. instance created.\n";
delete pD;
```

Test it !



Multiple Inheritance

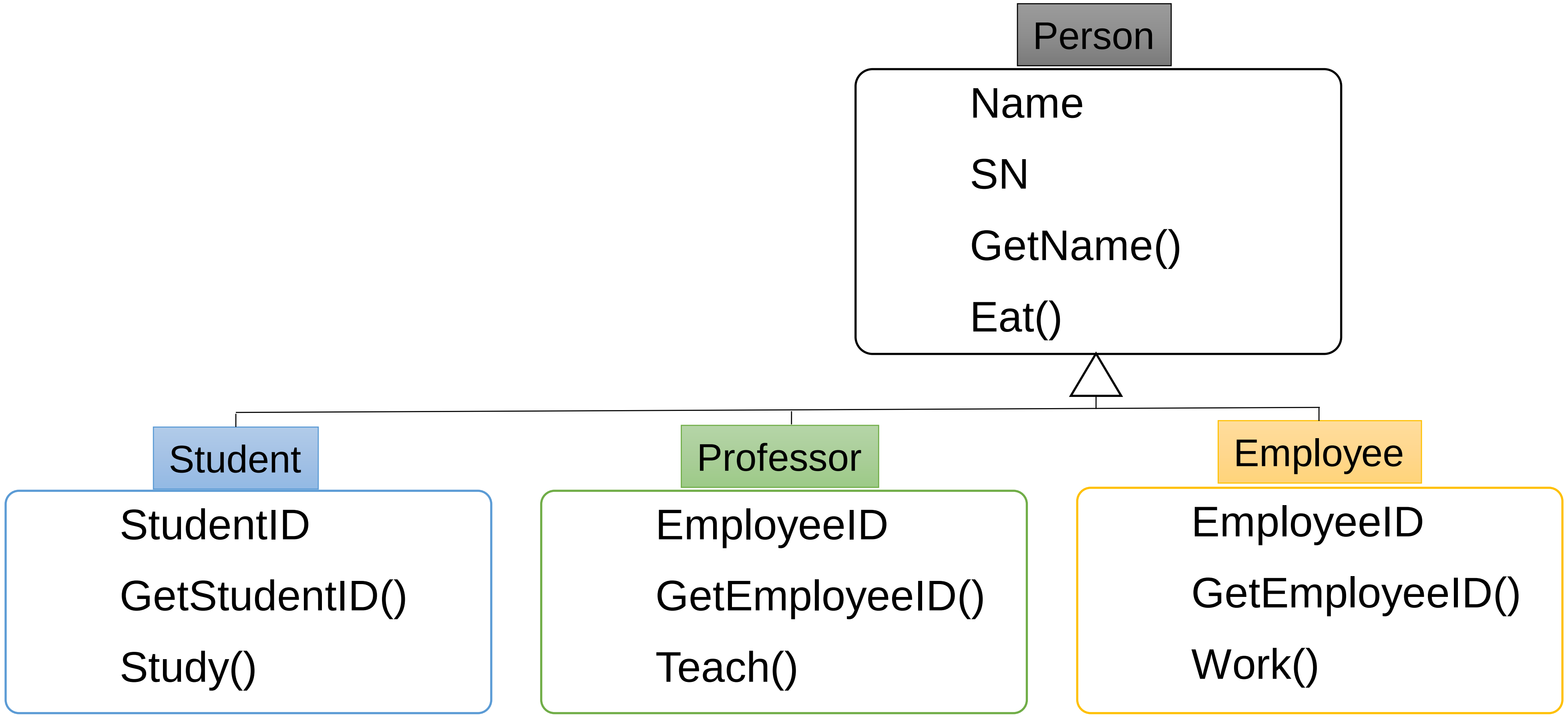
- What if we want to inheritance from more than 2 base classes

```
class Base1 {
    …
};
class Base2 {
    …
};
class Derived : private Base1, public Base2 {
    …
};
Derived d;
```

```
class Derived, public Base2 {
private:
    protected/public variable/method in Base1
public:
    protected/public variable/method in Base2

    Derived() : constructor of Base1, constructor of
Base2{
        Constructor of Derived
     }
    …
};
Derived d;
```

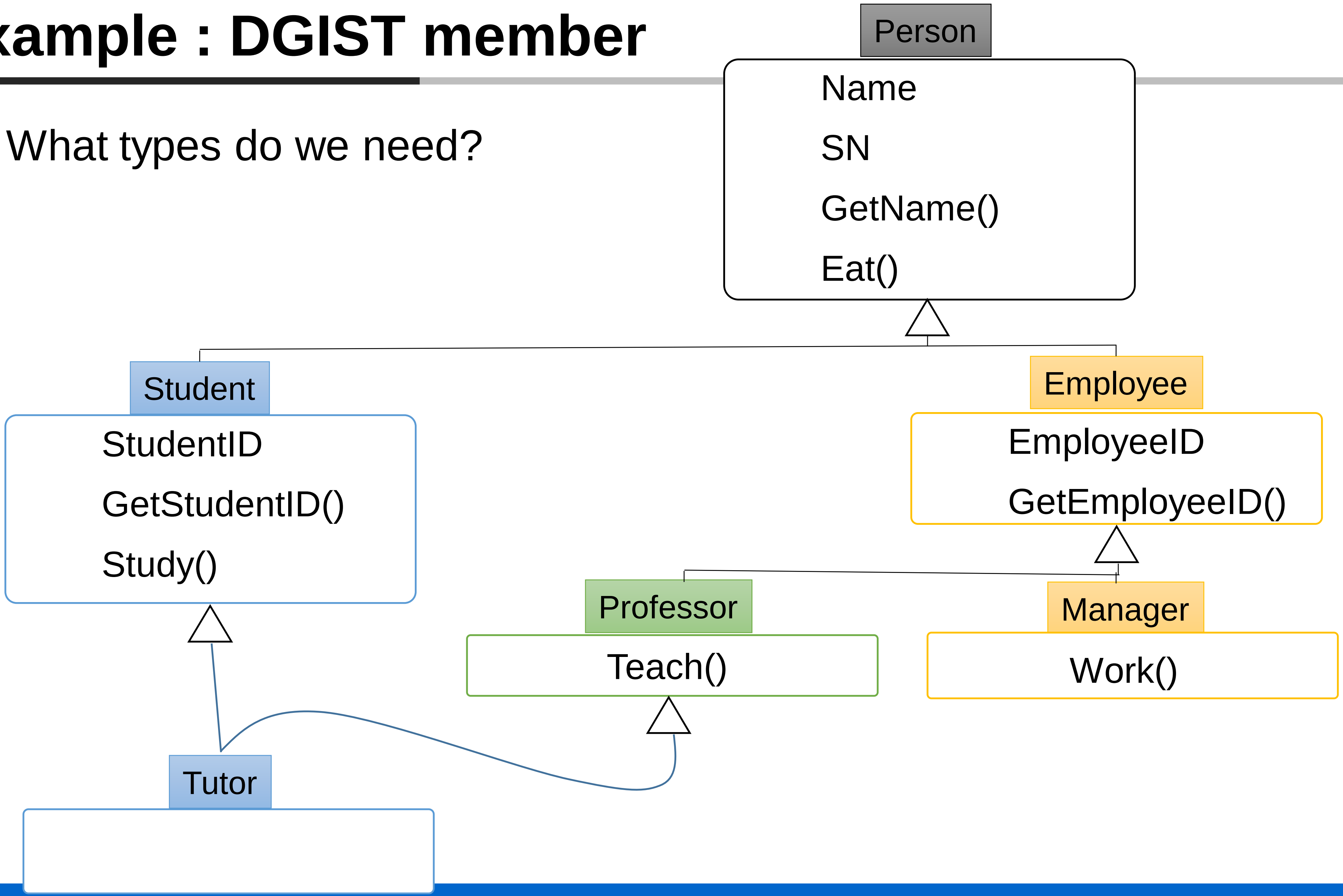
Example : DGIST member

- What types do we need?



Example : DGIST member

- What types do we need?



References

- Learn c++
  - https://www.learncpp.com/
  - Chapter : 11 1-5, 7

Q

ANY QUESTIONS?