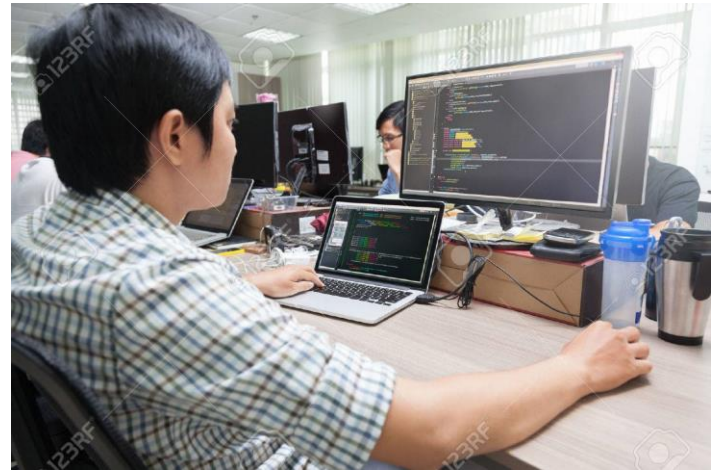


Two Types of Roles

Two Types of Roles



Computer Maker



Computer User

Two Types of Roles

Class Designer (Class Maker)

Those who design classes

Developer (Class User)

Those who use the classes (by making objects)

Two Types of Access Modifiers

(Think like a Class Designer)

`private`

`public`

Two Types of Access Modifiers



private

Your toothbrush
(Only you can use it)

public

Two Types of Access Modifiers



private

Your toothbrush
(Only you can use it)



public

Your toothpaste
(Anyone can use it)

Inheritance

Creating a class from another class(s)

$$X = 1 + 2 + 3 + 4$$

Inheritance

Creating a class from another class(s)

$$X = 1 + 2 + 3 + 4$$

$$Y = 1 + 2 + 3 + 4 + 5$$

Inheritance

Creating a class from another class(s)

$$X = 1 + 2 + 3 + 4$$

$$Y = \rightarrow X + 5$$

Inheritance

Creating a class from another class(s)

```
class Point
{
public:
    int x;
    int y;
};
```

Inheritance

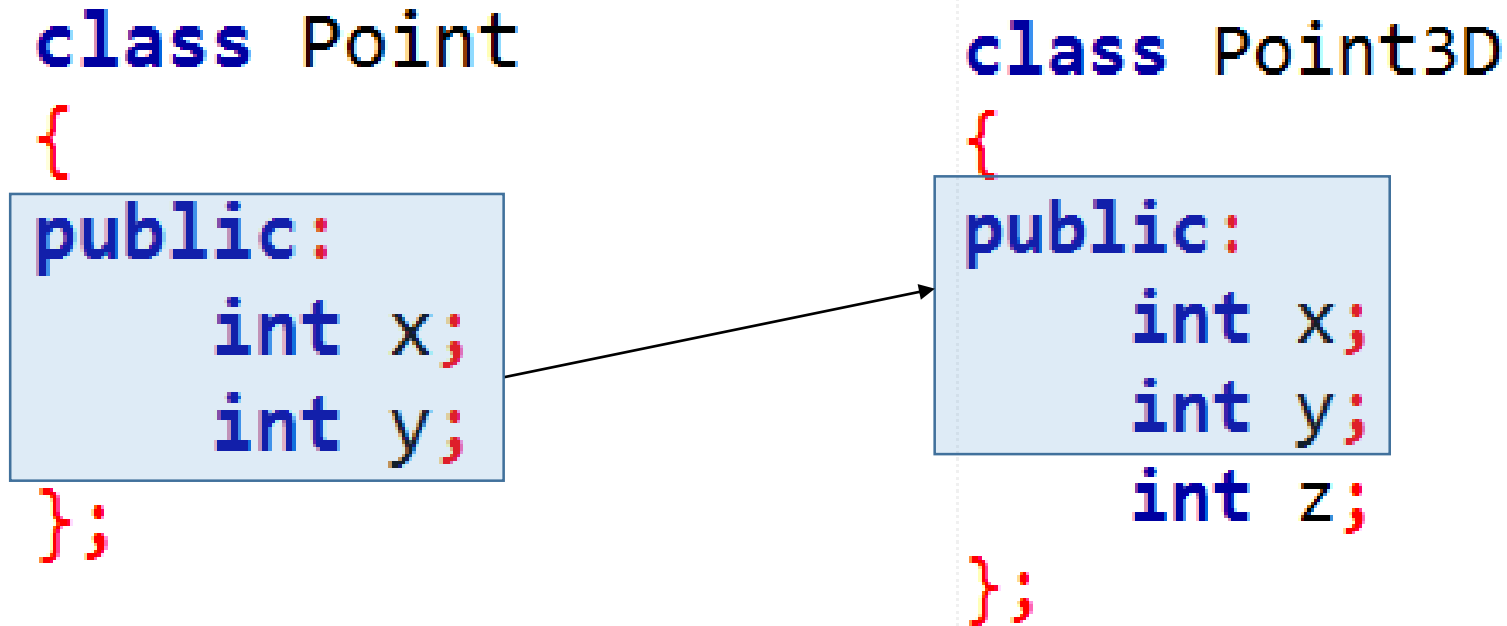
Creating a class from another class(s)

```
class Point
{
public:
    int x;
    int y;
};
```

```
class Point3D
{
public:
    int x;
    int y;
    int z;
};
```

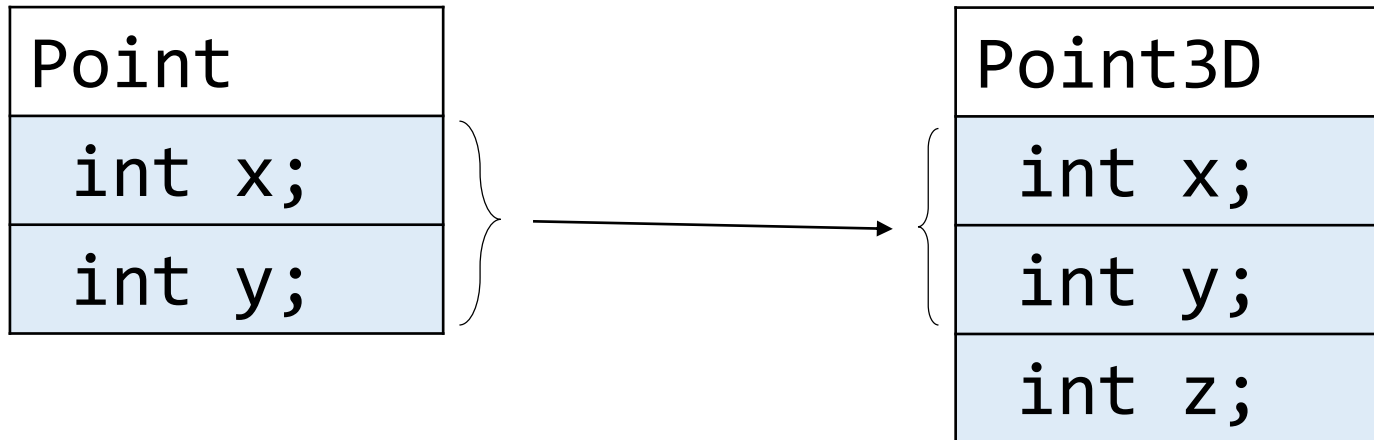
Inheritance

Creating a class from another class(s)

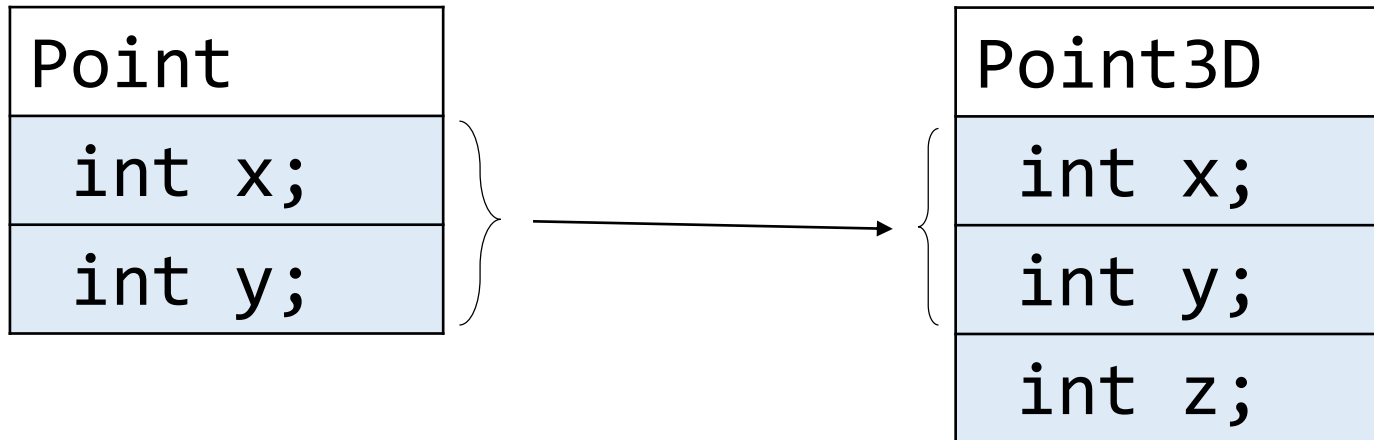


Inheritance

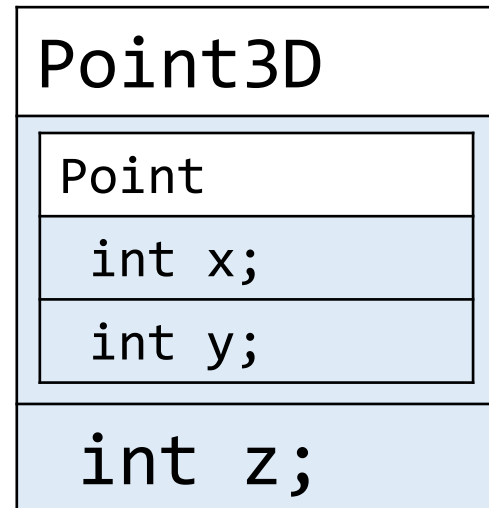
 Public member



Inheritance



We need something like this:



Inheritance Syntax

```
class Point
{
public:
    int x;
    int y;
};
```

Point
int x;
int y;

Inheritance Syntax

```
class Point
{
public:
    int x;
    int y;
};
```

Notice the mark

```
class Point3D :
{

};
```

Point
int x;
int y;

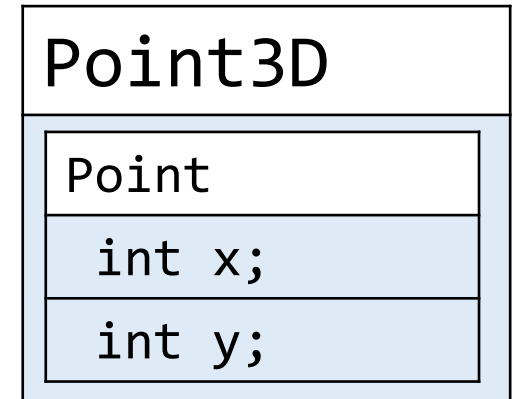
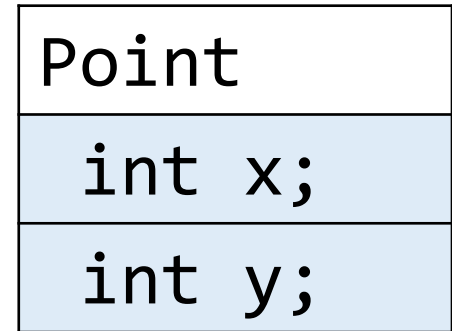
Point3D

Inheritance Syntax

```
class Point
{
public:
    int x;
    int y;
};
```

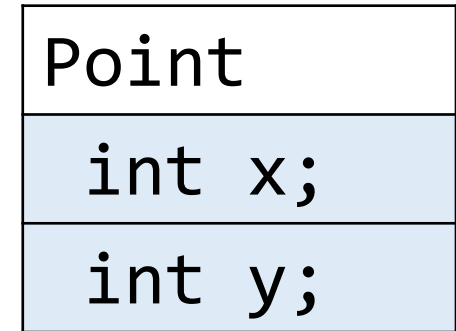
Not 'class Point'

```
class Point3D : public Point
{
};
```

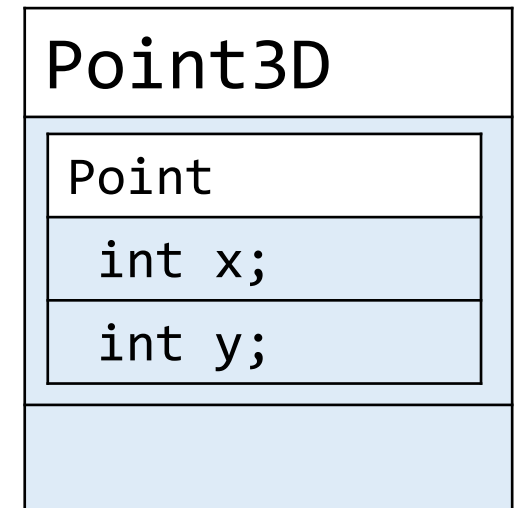


Inheritance Syntax

```
class Point
{
public:
    int x;
    int y;
};
```

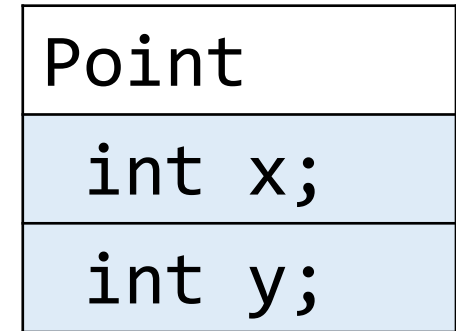


```
class Point3D : public Point
{
public:
};
```

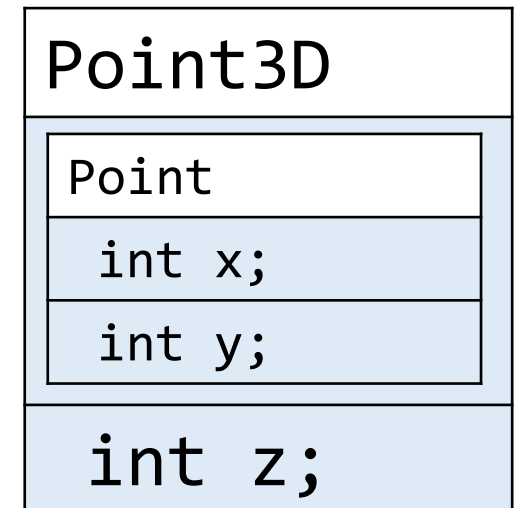


Inheritance Syntax

```
class Point
{
public:
    int x;
    int y;
};
```



```
class Point3D : public Point
{
public:
    int z;
};
```



Class designer's view



Inheritance Syntax

```
class Point
{
public:
    int x;
    int y;
};
```

Point
int x;
int y;

```
class Point3D : public Point
{
public:
    int z;
};
```

Point3D
int x;
int y;
int z;

Developer's view



Terminology

Derived class

Base class



```
class Point3D : public Point
```

Point3D
int x;
int y;
int z;

Point
int x;
int y;

A More Useful Example

An undergraduate student in MIST receives marks in Theory and Sessional individually. Design a Student class that will hold the respective marks. Write appropriate getter and setter functions (except constructor).

Student Class

 Private member

Student
<pre>int theory; int sessional;</pre>
<pre>void setMarks(...); int getTheory(); int getSessional();</pre>

Extending Student Class

Student

```
int theory;  
int sessional;  
  
void setMarks(...);  
int getTheory();  
int getSessional();
```

L4Student

```
...  
int thesis;  
  
...  
void setThesis(...);  
int getThesis();
```


Extending Student Class

```
class L4Student
```

Student
int theory; int sessional;
void setMarks(_); int getTheory(); int getSessional();

L4Student

Extending Student Class

```
class L4Student : public Student
```

Student
int theory; int sessional;
void setMarks(_); int getTheory(); int getSessional();

L4Student

Extending Student Class

```
class L4Student : public Student
```

Student

```
int theory;  
int sessional;
```

```
void setMarks(_);  
int getTheory();  
int getSessional();
```

L4Student

Student

```
int theory;  
int sessional;
```

```
void setMarks(_);  
int getTheory();  
int getSessional();
```

Extending Student Class

```
class L4Student : public Student
```

Student

```
int theory;  
int sessional;
```

```
void setMarks(_);  
int getTheory();  
int getSessional();
```

L4Student

Student

```
void setMarks(_);  
int getTheory();  
int getSessional();
```

Extending Student Class

```
class L4Student : public Student
{
private:
```

Student

```
int theory;
int sessional;
```

```
void setMarks(_);
int getTheory();
int getSessional();
```

L4Student

Student

```
void setMarks(_);
int getTheory();
int getSessional();
```

Extending Student Class

```
class L4Student : public Student
{
private:
    int thesis;
```

Student
int theory; int sessional;
void setMarks(_); int getTheory(); int getSessional();

L4Student			
int thesis;			
<table><tr><th>Student</th></tr><tr><td></td></tr><tr><td>void setMarks(_); int getTheory(); int getSessional();</td></tr></table>	Student		void setMarks(_); int getTheory(); int getSessional();
Student			
void setMarks(_); int getTheory(); int getSessional();			

Extending Student Class

```
class L4Student : public Student
{
private:
    int thesis;
public:
    void setThesis(int t)
    {
        thesis = t;
    }
}
```

Student
int theory; int sessional;
void setMarks(_); int getTheory(); int getSessional();

L4Student			
int thesis;			
<table><tr><th>Student</th></tr><tr><td></td></tr><tr><td>void setMarks(_); int getTheory(); int getSessional();</td></tr></table> void setThesis(_)	Student		void setMarks(_); int getTheory(); int getSessional();
Student			
void setMarks(_); int getTheory(); int getSessional();			

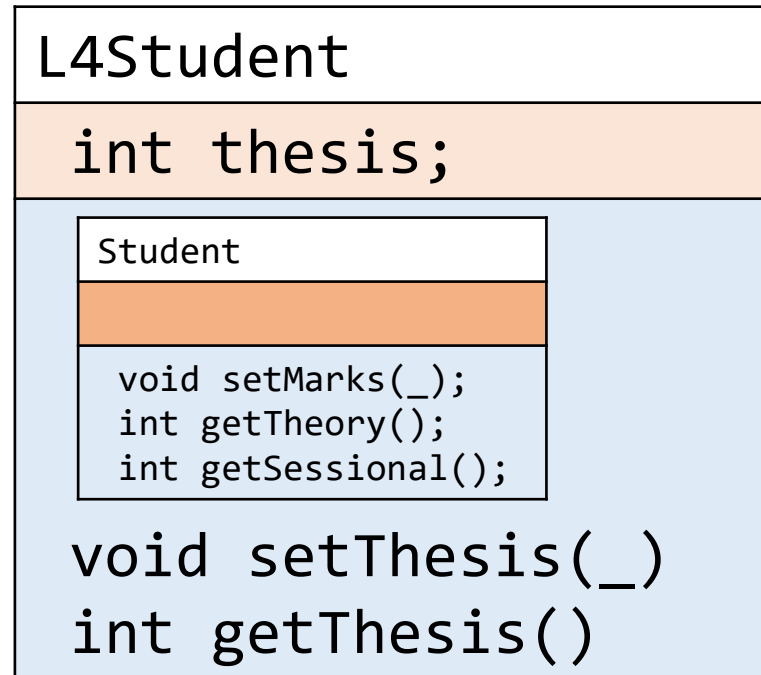
Extending Student Class

```
class L4Student : public Student
{
private:
    int thesis;
public:
    void setThesis(int t)
    {
        thesis = t;
    }
    int getThesis()
    {
        return thesis;
    }
};
```

Student
int theory; int sessional;
void setMarks(_); int getTheory(); int getSessional();

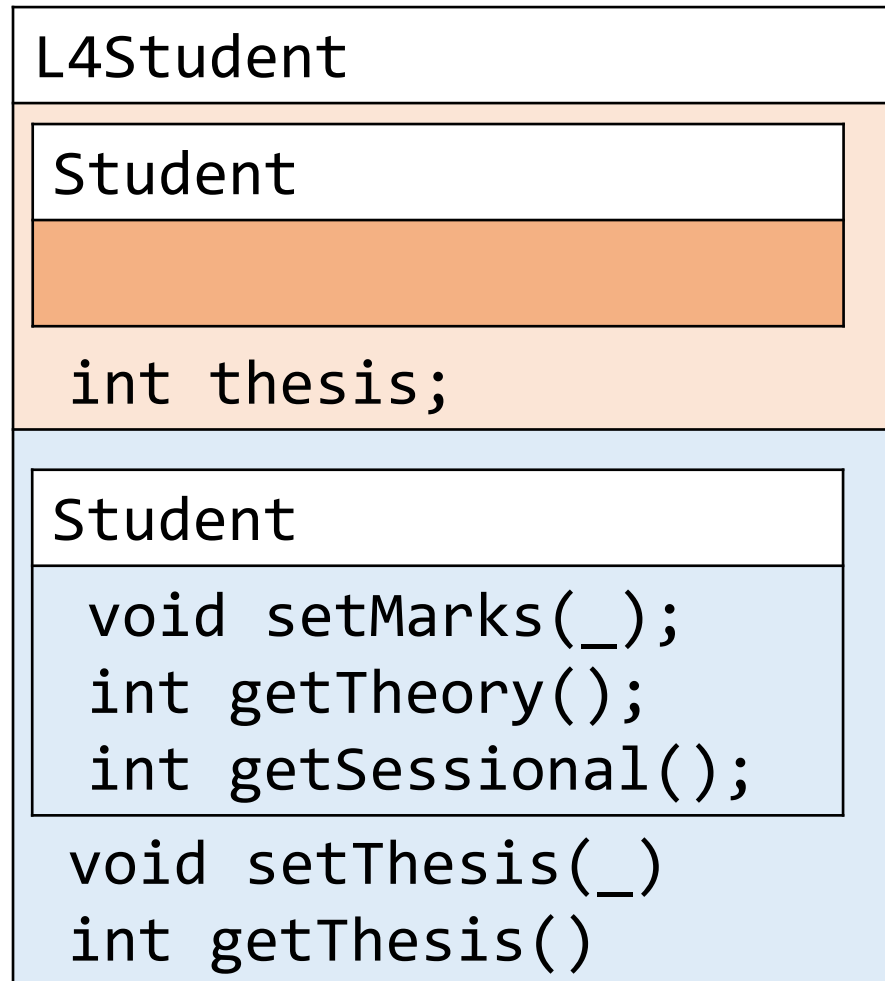
L4Student			
int thesis;			
<table><tr><th>Student</th></tr><tr><td></td></tr><tr><td>void setMarks(_); int getTheory(); int getSessional();</td></tr></table> void setThesis(_) int getThesis()	Student		void setMarks(_); int getTheory(); int getSessional();
Student			
void setMarks(_); int getTheory(); int getSessional();			

L4Student Class



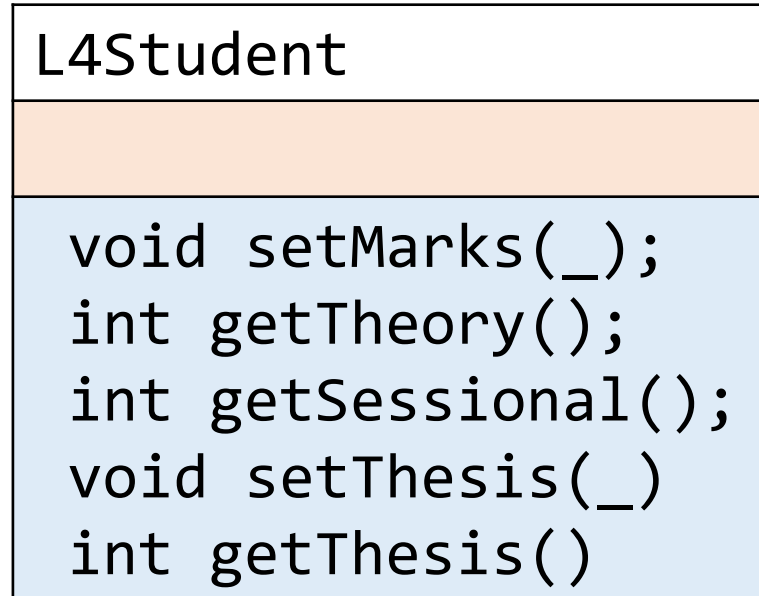
Class designer's view

L4Student Class



*Class designer's view
(Expanded)*

L4Student Class



Developer's view

Notice that

L4Student is a Student

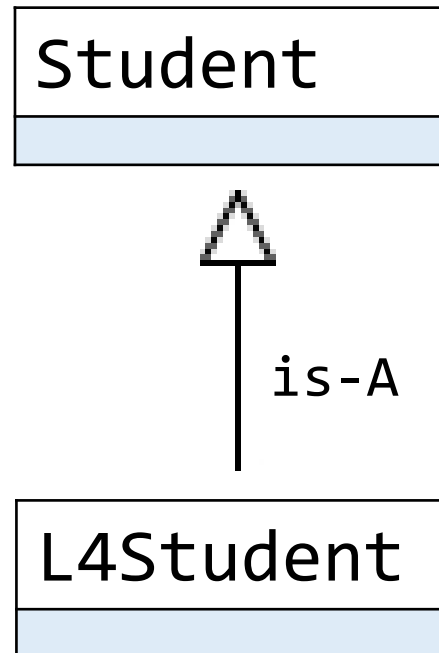
Point3D is a Point

Triangle is a Shape

SalaryAccount is an Account

Car is a Vehicle

is-A Notation (Applicable for public inheritance)

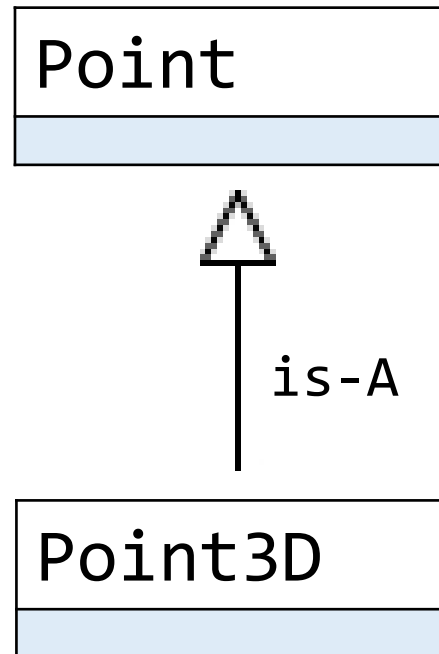


L4Student is-A Student

Derived-Class is-A Base-Class

The direction of the arrow may look counter-intuitive, but it's accurate.

is-A Notation (Applicable for public inheritance)



Point3D is-A Point

Derived-Class is-A Base-Class

The direction of the arrow may look counter-intuitive, but it's accurate.

Private Inheritance

```
class L4Student : private Student
```

Student
int theory; int sessional;
void setMarks(_); int getTheory(); int getSessional();

L4Student

Private Inheritance

```
class L4Student : private Student
```

Student

```
int theory;  
int sessional;
```

```
void setMarks(_);  
int getTheory();  
int getSessional();
```

L4Student

Student

```
void setMarks(_);  
int getTheory();  
int getSessional();
```


Private Inheritance

```
class L4Student : private Student
```

Student

```
int theory;  
int sessional;
```

```
void setMarks(_);  
int getTheory();  
int getSessional();
```

L4Student

Student

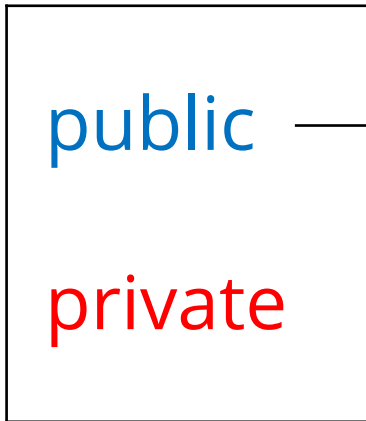
Student

```
void setMarks(_);  
int getTheory();  
int getSessional();
```

Access Specifier Summary

public Inheritance

Base Class



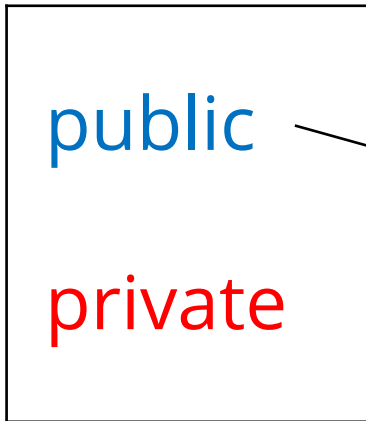
Derived Class



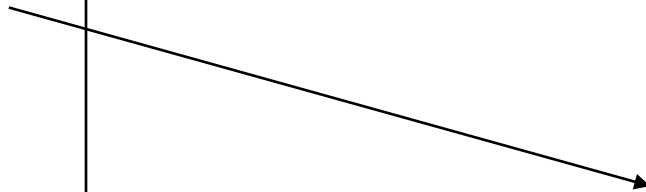
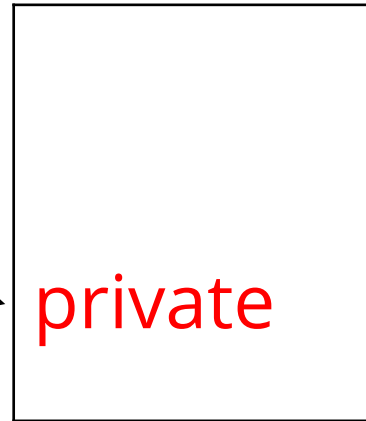
Access Specifier Summary

private Inheritance

Base Class

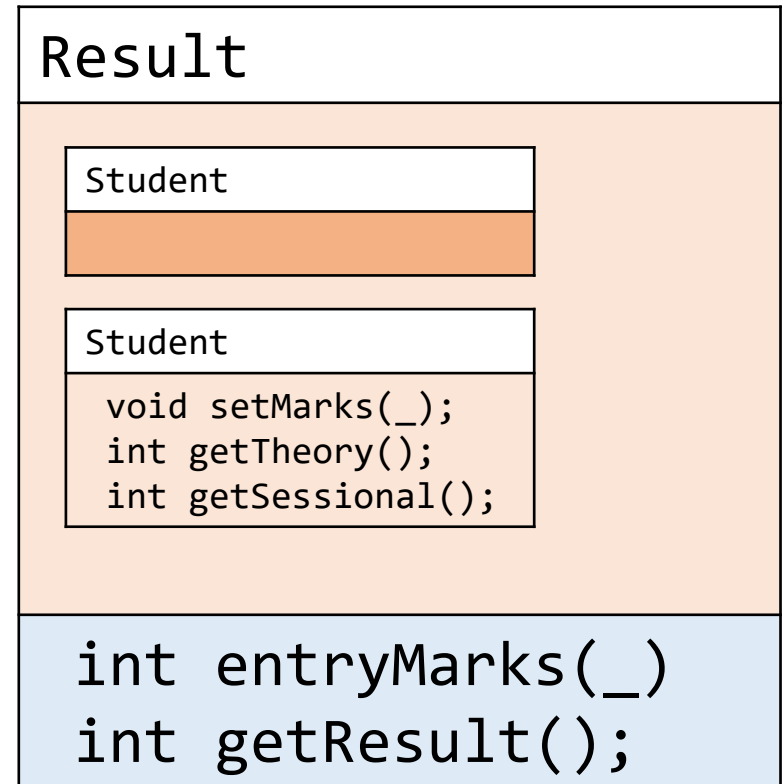
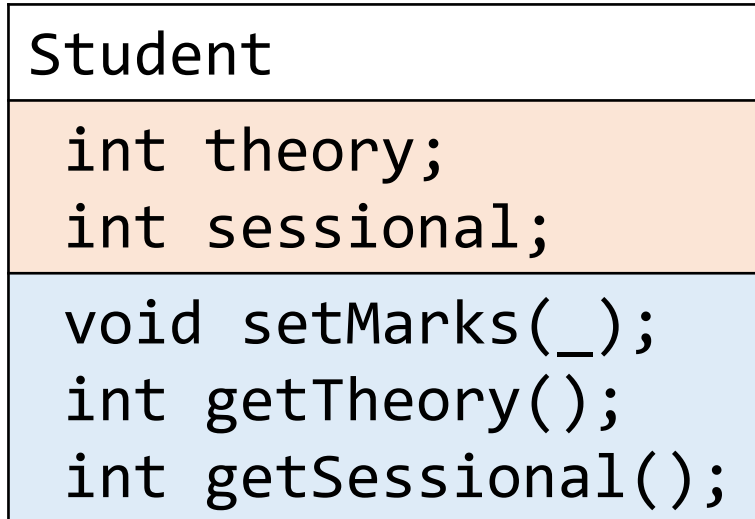


Derived Class



A better usage of Private Inheritance

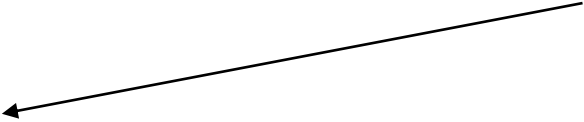
```
class result : private Student
```



A better usage of Private Inheritance

```
class Result : private Student
{
public:
    int entryMarks(int a, int b)
    {
        setMarks(a, b);
    }
    int publishResult()
    {
        int total = getTheory() + getSessional();
        return total;
    }
};
```

*This is called
Delegate Function,
Because it 'delegates'
Its duty to another
Function.*



Note

Result is not a Student! So it's not a *is-a* relation.

It's a *has-a* relation.

Result *has-a* Student.

More Examples of has-a

Car has-an Engine

Body has-an Organ

School has-a Classroom

Course has-a ClassTest