**Namespaces**

Remember to put everything into namespaces if you want to use it.

Example two classes:

B1.h

#pragma once

#include "../Simple\_window.h" // get access to our window library

#include "../Graph.h" // get access to our graphics library facilities

namespace Ex14

{

class B1

{

public:

B1();

~B1();

virtual void vf();

void f();

};

}

B1.cpp

#include "B1.h"

namespace Ex14

{

B1::B1()

{

}

B1::~B1()

{

}

void B1::vf()

{

cout << "B1::vf()" << endl;

}

void B1::f()

{

cout << "B1::f()" << endl;

}

}

And class D1 which derivates form B1

D1.h

#pragma once

// "Programming -- Principles and Practice Using C++" by Bjarne Stroustrup

#include "../Simple\_window.h" // get access to our window library

#include "../Graph.h" // get access to our graphics library facilities

#include "B1.h"

class D1 : B1// Error can't see the B1 class

{

public:

void vf();

void f();

};

D1.cpp

#include "D1.h"

void D1::vf()

{

}

void D1::f()

{

}

Solution:

Put the D1 class into Ex14 namespace

D1.h

#pragma once

// "Programming -- Principles and Practice Using C++" by Bjarne Stroustrup

#include "../Simple\_window.h" // get access to our window library

#include "../Graph.h" // get access to our graphics library facilities

#include "B1.h"

namespace Ex14

{

class D1 : B1// Okay

{

public:

};

}

D1.cpp more convenient solution.

You don’t have to add “Ex14::” prefix for each function

#include "D1.h"

namespace Ex14

{

void D1::vf()

{

}

void D1::f()

{

}

}

D1.cpp alternative

#include "D1.h"

void Ex14::D1::vf()

{

}

void Ex14::D1::f()

{

}