#ifindef, *define*, and *endif* with the closing statement *#endif* are a guard to multiple inclusions.

If a file header includes an *.h* file and another file header includes the same *.h* file, the compiler sees the class definition twice, and it complains about two classes with the same name (even though you are referring to the same class).

Another option:
Using *#pragma once* in the header of the class → **BUT** not all compilers will recognize this command.

MyClass.h (header → public interface)

```
//Name header: your name, date, etc.
//Brief comment on purpose of the class

#ifndef MYCLASS_H
#define MYCLASS_H

#include <...> //libraries: do you have a string var?
using...;

//global constants

class MyClass  //class definition
{

public:
	MyClass();  //default constructor → ALWAYS

	MyClass(any parameters); //other constructors

	//set functions will modify member variables
	void setFunction(any parameters);

	//get functions do not modify member variables
	returnType getFunction() const;

	//other functions
	//(may or may not be const)

	~MyClass();  //destructor → ALWAYS

private:
	//member variables
	//static variables
	//(may or may not have private functions)
};

#endif
```

Main.cpp

```
//header: your name,date, etc.
#include "MyClass.h"

#include <...> //libraries
using...;

//global constants

//function declarations (if any)

int main()
{

	cout << endl;
	system("Pause"); //for VS
	return 0;
}

//function definitions (if any)
```

MyClass.cpp

```
#include "MyClass.h"

//initialize static vars (if any)

MyClass::MyClass(){}  //default constructor
{  (may or may not initialize member variables)  }

MyClass::MyClass(parameters) //other constructors
{  (initialize member variables) }

//set functions
void MyClass::setFunction(any parameters)
{  (your code here...)  }

//get functions
returnType MyClass::accessorFunction() const
{  (your code here...)  }

MyClass::~MyClass()
{  (may or may not have code)  }
```