# Tutorial for Week 2 Session 1: Class and Inheritance

## Contents

1. Tutorial 2a: Class
2. Tutorial 2b: Multiple File Program
3. Tutorial 2c: Inheritance

## Tutorial w2s1a: Class

### Learning outcomes:

1. Create a class
2. Instantiate (create) objects out of a class
3. Implement function overloading
4. Access the member data of the objects
5. Invoke the methods of the objects

### Tasks:

1. Create a new file (CTRL-N).
2. Solve the following programming problem.
3. Create a class Box which has three data members: length, width, height;
4. Create two member functions (methods) to calculate area and volume
5. Create two constructors: default and parameterized
6. Create an object of Box in the main function
7. Access the member data and invoke the member functions
8. You can start from the below code:

#include <iostream>

using namespace std;

class Box{

// function members (methods)

public:

int getLength();

int getWidth();

int getHeight();

void setLength(int l);

void setWidth(int w);

void setHeight(int h);

int calculateArea();

int calculateVolume();

Box(); // default constructor

Box(int l, int w, int h); // constructor

// data members

protected:

int length;

int width;

int height;

};

// your code here

1. Complete the program.
2. Save the source file, compile and run it.

### Questions:

1. What new C++ programming keywords that you learned from this exercise?
2. What is scope operator? Why do we need a scope operator?
3. What are access modifiers (or specifiers)? How do they affect the access of the member data and functions?
4. Why do we need an access specifier?
5. What is member access operator?
6. What the difference between default constructor and normal ones?
7. What is the reason for implementing the member functions outside of the class body?
8. What is data or information hiding? Why is it useful?
9. What is encapsulation? How encapsulation is implemented in C++
10. What is function prototypes?

### Summary:

In this tutorial, you have learned how to create a class in C++. You created constructors, specify the access for the data and functions, declare the function prototype in the class and provide the implementation (the body) of the functions by using a scope operator. After your created an instance of the class (objects), you access the member data by calling the set and get methods and also invoked the methods to calculate the area and volume.

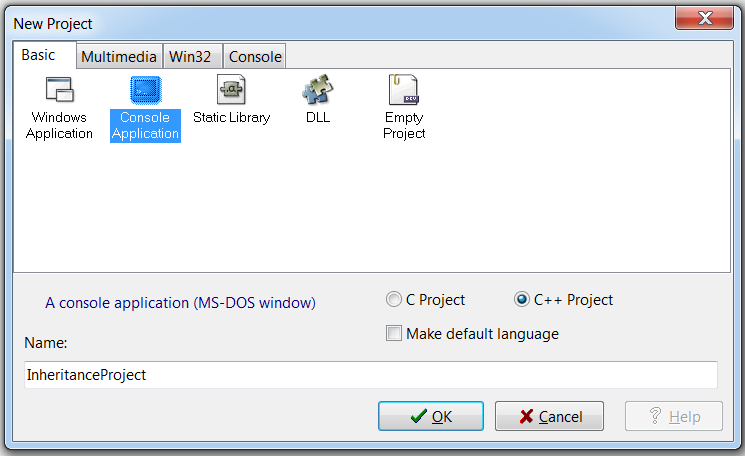
## Tutorial w2s1b: Multiple files Program

### Learning outcomes:

1. Create Project in Dev-C++
2. Separate the class declaration and class implementation in different files
3. Compile and run multiple file program

### Tasks:

1. Create a new Project in Dev-C++ (File, New, Project)



Since our program is console-program, we chose the "Console Application". Provide a name for your new project, for example "InheritanceProject", then when you click OK, you will be asked in which folder you want to save the project. Your project will automatically contains main.cpp file. Just delete this file.

1. Solve the following programming problems.
2. Insert a new file in the project (right click on the project, new file)
3. In this new file, copy the following code from tutorial 1a:

class Box{

// function members (methods)

public:

int getLength();

int getWidth();

int getHeight();

void setLength(int l);

void setWidth(int w);

void setHeight(int h);

int calculateArea();

int calculateVolume();

Box(); // default constructor

Box(int l, int w, int h); // constructor

//~Box(); // destructor

// data members

protected:

int length;

int width;

int height;

};

1. Save the file as "Box.h" (dot h means header file)
2. Insert another new file in the project
3. In this new file, copy the following code from tutorial 1a:

#include <iostream>

#include "Box.h"

using namespace std;

int Box::getLength(){

return length;

}

int Box::getWidth(){

return width;

}

int Box::getHeight(){

return height;

}

void Box::setLength(int l){

length = l;

}

void Box::setWidth(int w){

width = w;

}

void Box::setHeight(int h){

height = h;

}

Box::Box(){}

Box::Box(int l, int w, int h){

length = l;

width = w;

height = h;

}

int Box::calculateArea(){

int area = 2\*(width\*height+width\*length+height\*length);

return area;

}

int Box::calculateVolume(){

int vol = length\*width\*height;

return vol;

}

1. Save the file as "Box.cpp" (this is called an implementation file)
2. Insert another new file in the project
3. In this new file, copy the following code from tutorial 1a:

#include <iostream>

#include "Box.h"

using namespace std;

int main(){

// create an object of Box

Box mybox;

Box yourbox(3, 6, 10);

// set the width, length and height of mybox

mybox.setLength(5);

mybox.setWidth(8);

mybox.setHeight(15);

// get the dimension of mybox

cout << "The dimension (length, width, height) of mybox is: ("<< mybox.getLength() << ", " << mybox.getWidth() << ", " << mybox.getHeight() << ")" << endl;

// calculate the area of myBox

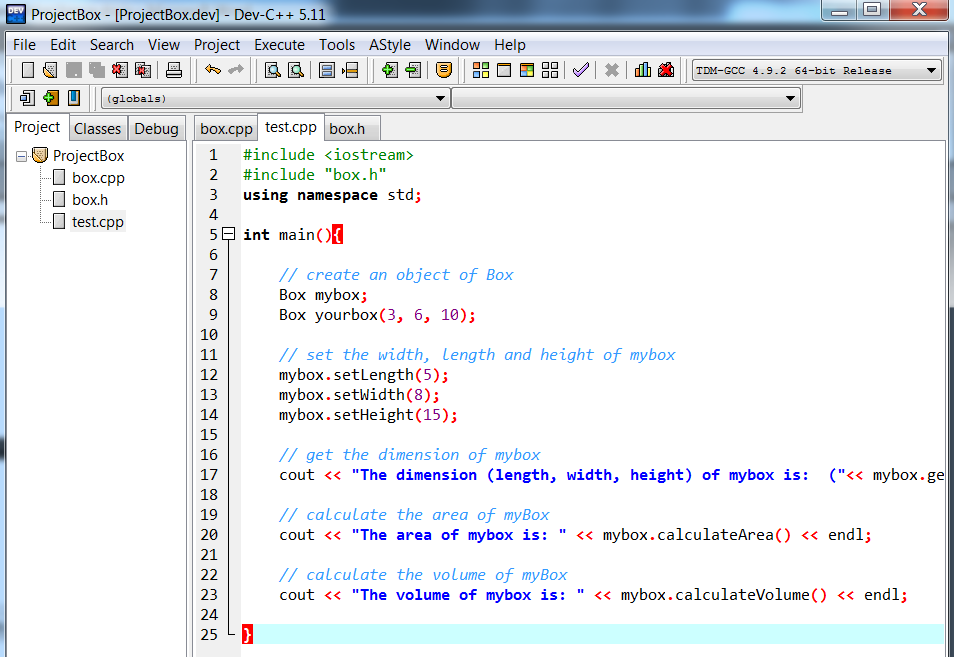
cout << "The area of mybox is: " << mybox.calculateArea() << endl;

// calculate the volume of myBox

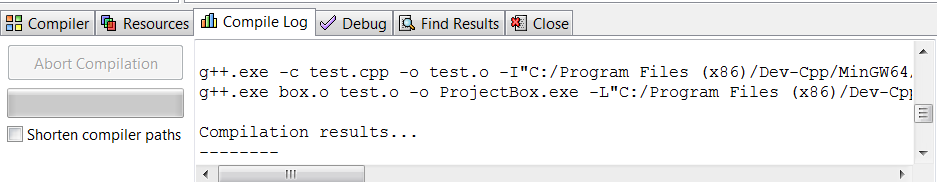
cout << "The volume of mybox is: " << mybox.calculateVolume() << endl;

}

1. Save the file as "Test.cpp" (this is called an implementation file)
2. Notice that you have split your program into three files: box.h (header file), box.cpp (implementation file) and test.cpp (the main function)



1. Save the project and source file, compile and run it. Notice that when you compiled, the compiler also "linked" these three files.



### Questions:

1. What is header file? What code should go into this file?
2. What is implementation file? What code should go into this file?
3. Why do we want to separate our program into header files and implementation files?
4. What is the purpose of compiling? What files are produced in the compilation?
5. What is the purpose of linking? What files are produced in the linking?

### Summary:

In this tutorial, you have learned how to separate a program into header and implementation files. You put the class definition (and other statement such as global variables, external variable, pre-processor such as define statement, etc) in the header file. The implementation of the class goes into the implementation file (.cpp). You need to include the header file (#include "file.h") in your implementation file, otherwise the compiler will generate errors. Having multiple files in a program can improve the structure and organization of the program. Just imagine if you have a lengthy program with all the code goes into one file.

## Tutorial w2s1c: Inheritance

### Learning outcomes:

1. Create class inheritance
2. Implement function overriding

### Tasks:

1. Create a new file (CTRL-N).
2. Solve the following programming problem.

1. Create a Shape class which has width and height data members

2. Create a member function to calculate area

3. Create a sub class Rectangle which inherits the Shape class

4. Override the function to calculate area

5. In the main function. create objects of a derived class

6. Access its member data and invoke its methods

1. You can start from the below code:

#include <iostream>

using namespace std;

// base class or super class

class Shape {

public:

int getWidth(){

return width;

}

int getHeight(){

return height;

}

void setWidth(int w){

width = w;

}

void setHeight(int h){

height = h;

}

int getArea(){

return width\*height;

}

private:

int width;

int height;

};

// Derived class or sub class

class Triangle: public Shape{

public:

int getArea(){

return 0.5\*getWidth()\*getHeight();

}

};

// your code here

1. Complete the program.
2. Save the source file, compile and run it.

### Questions:

1. What new C++ programming keywords that you learned from this exercise?
2. What is the syntax for inheritance?
3. What is function overloading (see Tutorial 2a)? Why function overloading is useful?
4. What is function overriding? Why function overriding is useful?
5. Explain the difference between overloading and overriding.

### Summary:

In this tutorial, you have learned how to implement inheritance in C++. You override the function in the base class by re-writing (redefining) the body of the function.

---end of Tutorial Week 2---