**COMPOSITION**

**Example**

#include <iostream>

using namespace std;

#include <string>

**class student**

**{**

**int rollNumber;**

**string Name;**

**public:**

**student(int =0, string="Default");**

**void AddDetails(int, string);**

**void PrintDetails();**

**};**

**student::student(int rN, string N)**

**{**

**rollNumber=rN;**

**Name=N;**

**}**

**void student::PrintDetails()**

**{**

**cout<<rollNumber<<" "<<Name<<endl;**

**}**

**void student::AddDetails(int rN, string N)**

**{**

**rollNumber=rN;**

**Name=N;**

**}**

**class course**

**{**

**private:**

**string Name;**

**int creditHours;**

**string InstructorName;**

**student list;**

**public:**

**course( string="", int =0, string ="", int = -1, string ="None");**

**void PrintDetails();**

**void RegisterStudent();**

**};**

**course::course(string CName, int CH, string IN,int rN, string sName):list(rN, sName)**

**{**

**Name=CName;**

**creditHours=CH;**

**InstructorName=IN;}**

**void course::RegisterStudent()**

**{**

**string N;**

**int rN;**

**cout<<"\n Enter Student Name ";**

**cin>>N;**

**cout<<"\n Enter Roll Number ";**

**cin>>rN;**

**list.AddDetails(rN, N); }**

**void course:: PrintDetails()**

**{**

**cout<<Name<<endl;**

**cout<<creditHours<<endl;**

**cout<<InstructorName<<endl;**

**cout<<"\nStudent Details .. \n";**

**list.PrintDetails();**

**}**

**What is the output?**

int main()

{

course Programming("Programming", 3 , "Tooba");

Programming.PrintDetails();

return 0;

}

Composition of Array of objects in a different class

**#include <iostream>**

**using namespace std;**

**#include <string>**

**class student**

**{**

**int rollNumber;**

**string Name;**

**public:**

**student(int =0, string="Default");**

**void AddDetails(int, string);**

**void PrintDetails();**

**};**

**student::student(int rN, string N)**

**{**

**rollNumber=rN;**

**Name=N;**

**}**

**void student::PrintDetails()**

**{**

**cout<<rollNumber<<" "<<Name<<endl;**

**}**

**void student::AddDetails(int rN, string N)**

**{**

**rollNumber=rN;**

**Name=N;**

**}**

**class course**

**{**

**private:**

**string Name;**

**int creditHours;**

**string InstructorName;**

**student list[35];**

**int currentStudents;**

**public:**

**course( string="", int =0, string ="");**

**void PrintDetails();**

**void RegisterStudent();**

**};**

**//constructor of container class**

**course::course(string CName, int CH, string IN)**

**{**

**Name=CName;**

**creditHours=CH;**

**InstructorName=IN;**

**currentStudents=0;}**

**void course::RegisterStudent()**

**{**

**string N;**

**int rN;**

**cout<<"\n Enter Student Name ";**

**cin>>N;**

**cout<<"\n Enter Roll Number ";**

**cin>>rN;**

**list[currentStudents].AddDetails(rN, N);**

**currentStudents++;}**

**void course:: PrintDetails()**

**{**

**cout<<Name<<endl;**

**cout<<creditHours<<endl;**

**cout<<InstructorName<<endl;**

**for (int i=0; i< currentStudents ; i++)**

**list[i].PrintDetails();}**

**Question – 1** What is the output of following codes?

|  |  |
| --- | --- |
| **int main()**  **{**  **course Programming("Programming", 3 , "Tooba");**  **Programming.RegisterStudent();**  **Programming.RegisterStudent();**  **Programming.RegisterStudent();**  **Programming.PrintDetails();**  **return 0;}** | **int main()**  **{**  **course Programming("Programming", 3 , "Tooba");**  **Programming.PrintDetails();**  **return 0;}** |

**Question - 2**

Write a member function for class Course to sort the list of students by roll number

Composition of dynamic Array of objects in a different class

class course

{

private:

string Name;

int creditHours;

string InstructorName;

student \*list;

int currentStudents;

const int maxStudent;

public:

course( string="", int =0, string ="",int =1 );

void PrintDetails();

void RegisterStudent();

};

course::course(string CName, int CH, string IN,int Max):maxStudent(Max)

{

Name=CName;

creditHours=CH;

InstructorName=IN;

currentStudents=0;

list=new student[maxStudent];

}

void course::RegisterStudent()

{

string N;

int rN;

cout<<"\n Enter Student Name ";

cin>>N;

cout<<"\n Enter Roll Number ";

cin>>rN;

if(currentStudents<maxStudent)

{list[currentStudents].AddDetails(rN, N);

currentStudents++;

}

}

void course:: PrintDetails()

{

cout<<Name<<endl;

cout<<creditHours<<endl;

cout<<InstructorName<<endl;

for (int i=0; i< currentStudents; i++)

list[i].PrintDetails();

}

int main()

{

course Programming("Programming", 3 , "Tooba", 35);

Programming.RegisterStudent();

Programming.RegisterStudent();

Programming.RegisterStudent();

Programming.PrintDetails();

return 0;

}

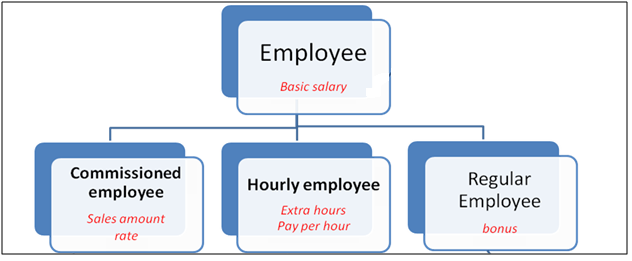
**INHERITANCE**

* Object Oriented Programming allows you to derive new classes from existing ones. This is called inheritance
* Suppose we want to define classes for circles, rectangles and triangles. These classes have many common features. In order to avoid redundancy, we may use inheritance.
* Inheritance is called “is-a” relationship
* Two Types of inheritance: Single Inheritance, and Multiple Inheritance

Base class

**Examples:**

**Single Inheritance**

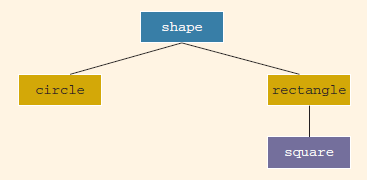


Derived class

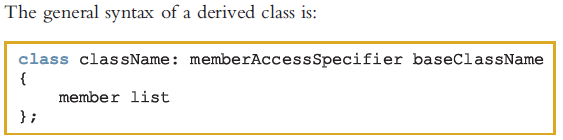
**Multiple Inheritance**



**Multiple Inheritance**

****

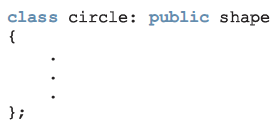
**C++ Syntax for derived class**

****



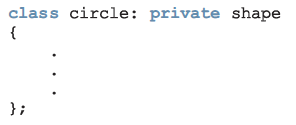
**Example**

**Public derivation**



Member access specifier tells whether features of base class are privately derived, protected or publicly derived.

**Private derivation**

****

**Protected derivation**

|  |
| --- |
| **class circle: protected shape**  {  .  .  .  }; |

**Example:**

**Rectangle**

class rectangleType

{

public:

void setDimension(double l, double w);

double getLength() const;

**Box**

double getWidth() const;

double area() const;

double perimeter() const;

void print() const;

rectangleType();

//Default constructor

rectangleType(double l, double w);

private:

double length;

double width;

};

void rectangleType::setDimension(double l, double w)

{

if (l >= 0)

length = l;

else

length = 0;

if (w >= 0)

width = w;

else

width = 0;

}

double rectangleType::getLength() const

{

return length;

}

double rectangleType::getWidth()const

{

return width;

}

double rectangleType::area() const

{

return length \* width;

}

double rectangleType::perimeter() const

{

return 2 \* (length + width);

}

void rectangleType::print() const

{

cout << "Length = " << length

<< "; Width = " << width;

}

rectangleType::rectangleType(double l, double w)

{

setDimension(l, w);

}

rectangleType::rectangleType()

{

length = 0;

width = 0;

}

class boxType: public rectangleType

{

public:

void setDimension(double l, double w, double h);

double getHeight() const;

double area() const;

double volume() const;

void print() const;

boxType();

//Default constructor

boxType(double l, double w, double h);

private:

double height;

};

void boxType::setDimension(double l, double w, double h)

{

rectangleType::setDimension(l, w);

if (h >= 0)

height = h;

else

height = 0;

}

double boxType::getHeight() const

{

return height;

}

double boxType::area() const

{

return 2 \* (getLength() \* getWidth()

+ getLength() \* height

+ getWidth() \* height);

}

double boxType::volume() const

{

return rectangleType::area() \* height;

}

void boxType::print() const

{

rectangleType::print();

cout << "; Height = " << height;

}

boxType::boxType()

{

height = 0.0;

}

boxType::boxType(double l, double w, double h)

: rectangleType(l, w)

{

if (h >= 0)

height = h;

else

height = 0;

}

#include <iostream>

#include <iomanip>

#include "rectangleType.h"

#include "boxType.h"

using namespace std;

int main()

{

rectangleType myRectangle1; //Line 1

rectangleType myRectangle2(8, 6); //Line 2

boxType myBox1; //Line 3

boxType myBox2(10, 7, 3); //Line 4

cout << fixed << showpoint << setprecision(2); //Line 5

cout << "Line 6: myRectangle1: "; //Line 6

myRectangle1.print(); //Line 7

cout << endl; //Line 8

cout << "Line 9: Area of myRectangle1: "

<< myRectangle1.area() << endl; //Line 9

cout << "Line 10: myRectangle2: "; //Line 10

myRectangle2.print(); //Line 11

cout << endl; //Line 12

cout << "Line 13: Area of myRectangle2: "

<< myRectangle2.area() << endl; //Line 13

cout << "Line 14: myBox1: "; //Line 14

myBox1.print(); //Line 15

cout << endl; //Line 16

cout << "Line 17: Surface Area of myBox1: "

<< myBox1.area() << endl; //Line 17

cout << "Line 18: Volume of myBox1: "

<< myBox1.volume() << endl; //Line 18

cout << "Line 19: myBox2: "; //Line 19

myBox2.print(); //Line 20

cout << endl; //Line 21

cout << "Line 22: Surface Area of myBox2: "

<< myBox2.area() << endl; //Line 22

cout << "Line 23: Volume of myBox2: "

<< myBox2.volume() << endl; //Line 23

return 0; //Line 24

}