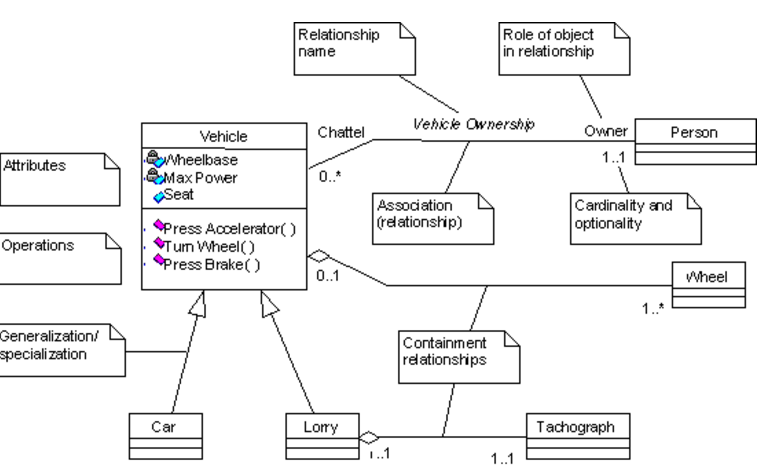
***UML Class Diagram Explained With C++ samples***

posted in C++ by Sadique Ali E

As you know a Class diagram is a diagram showing different classes in a system their attribute, operation and the relationship among different objects.



Even I have been using it for long time; I always had confusion each time when I use it. This post is meant for such people; also it will be helpful for beginners, here I will explain the class diagram with C++ class example.

**A class representation**

class Circle{

private:

double radius;

Point center;

public:

setRadius(double radius);

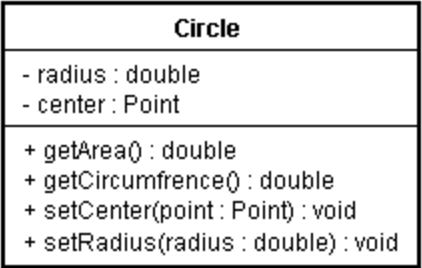
setCenter(Point center);

double getArea();

double getCircumfrence();

};

**Class diagram for the above class is shown below**



**Different visibility of the class can be represented as**

“+” Public

“-” Private

“#” Protected

**Different Parameter direction**

“in” The parameter is an input parameter.

“Inout” The parameter is capable of both input and output.

“Out” The parameter is an output parameter.

**Different type of members in a class**

1) Static members are represented as underlined.

2) Pure virtual functions are represented as italics.

**Class relationship**

In a system a class may be related to different classes,following are the different relation ship.

●Association (knows a)

●Dependency (uses a)

●Composition (has a)

●Aggregation (has a)

●Inheritance (is a)

●Class template

**Different Multiplicity in a relation**

“0..1” No instances, or one instance (optional, may)

“1” Exactly one instance

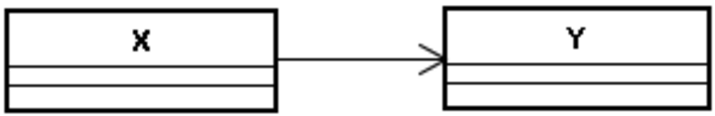
“0..\* or \*” Zero or more instances

“1..\*” One or more instances (at least one)

**Association**

One object is aware of another; it contains a pointer or reference to another object.

Representaion



C++ Example:

class X{

X(Y \*y) : y\_ptr(y) {}

void SetY(Y \*y) { y\_ptr = y; }

void f() { y\_ptr->Foo();}

...

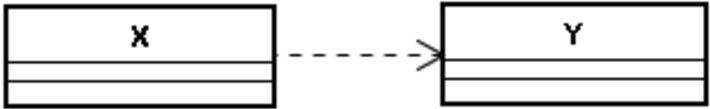
Y \*y\_ptr; // pointer

};

**Dependency**

One class depends on another if the independent class is a parameter variable or local variable of a method of the dependent class

Representaion



C++ Example:

class X {

...

void f1(Y y) {…; y.Foo(); }

void f2(Y \*y) {…; y->Foo(); }

void f3(Y &y) {…; y.Foo(); }

void f4() { Y y; y.Foo(); …}

void f5() {…; Y::StaticFoo(); }

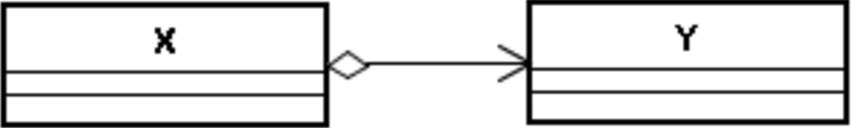
...

};

**Aggregation**

Aggregation can occur when a class is a collection or container of other classes, but where the contained classes do not have a strong life cycle dependency on the container—essentially, if the container is destroyed, its contents are not. You may have confusion between aggregation and association .Association differs from aggregation only in that it does not imply any containment.

Representaion



C++ Example:

Example 1

class Window{

public:

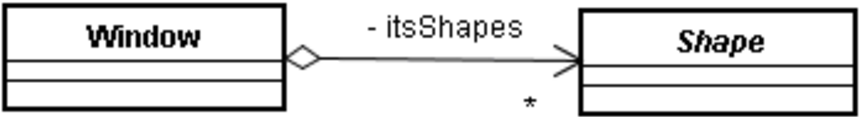
//...

private:

vector itsShapes;

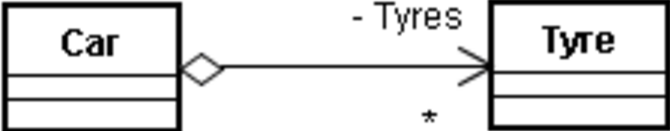
};

A window class contains a list of its shapes

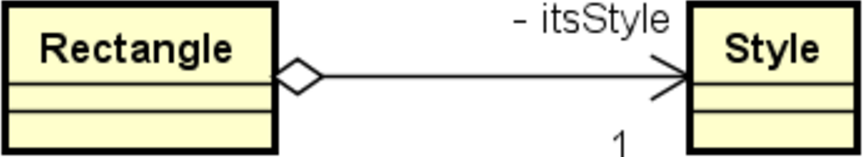
****

Example 2

A car has it’s tiers, and the scope of tyre doesn’t depend on a car since a tyre can be used for another car also



A Rectangle class has its style, which may be shared by other shapes also; life time of style doesn’t depend on Rectangle class.



**Composition**

Composition is the stronger form of aggregation. Composition can occur when a class is a collection or container of other classes, but where the contained classes have a strong life cycle dependency on the container—essentially, if the container is destroyed, its contents are also destroyed

Representation



C++ Example

class Circle{

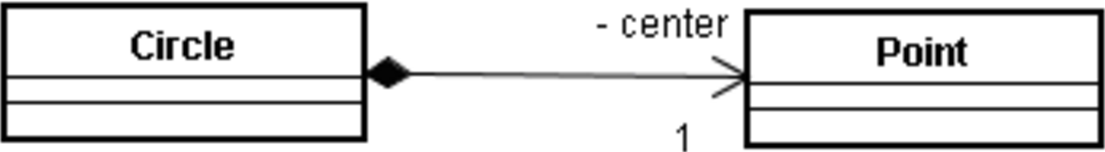
private:

...

Point center;

....

};



class X {

...

Y a; // 1; Composition

Y b[10]; // 0..10; Composition

};

class X {

X() { a = new Y[10]; }

~X(){ delete [] a; }

...

Y \*a; // 0..10; Composition

};

class X {

...

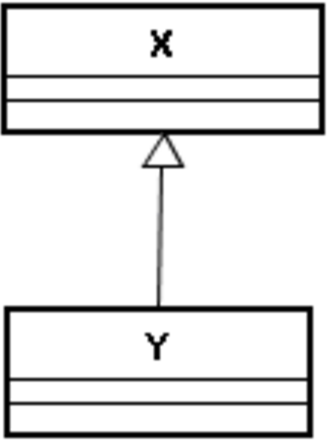
vector a;

};

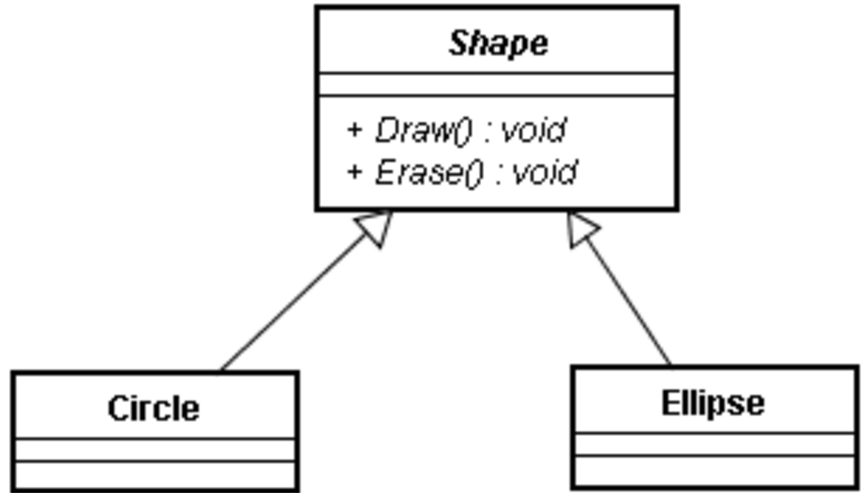
**Inheritance (Generalization)**

In Inheritance relationship a class is derived from another class. It is a “is a” relationship between two classes.

Representation



Here X and Y are normal classes.

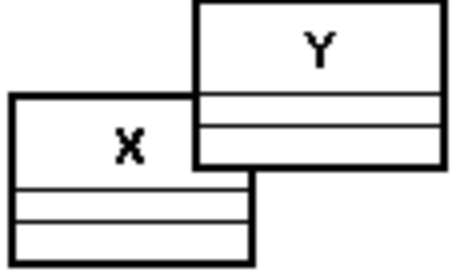


Here Shape is an abstract class that is why it is shown in Italics. Draw () and Erase() methods of Shape class is pure virtual function, so it is also shown as italics.

**Class Template**

Template class mean generic classes.Languages like C++, java, C# supports generic programming.

Representation



C++ represenatation

template

class X{

...

...

...

};

X Y

...

X a;

...