CSE 1203

Object Oriented Programming [C++]

Chapter 4:
Advanced Topics-3: UML

UML: Unified Modeling Language

What is UML?

The Unified Modeling Language (UML) was created to forge a common, semantically and syntactically rich visual modeling language for the architecture, design, and implementation of complex software systems both structurally and behaviorally.

What is class diagram

Class Diagram is one of the important UML diagrams for software development which shows the object classes in the system and the associations between these classes.

The class diagram is a static type structure diagram that describes the structure of a system by showing the system's classes, each class's attributes and operations, and also the relationship among the objects.

Benefits of class diagrams

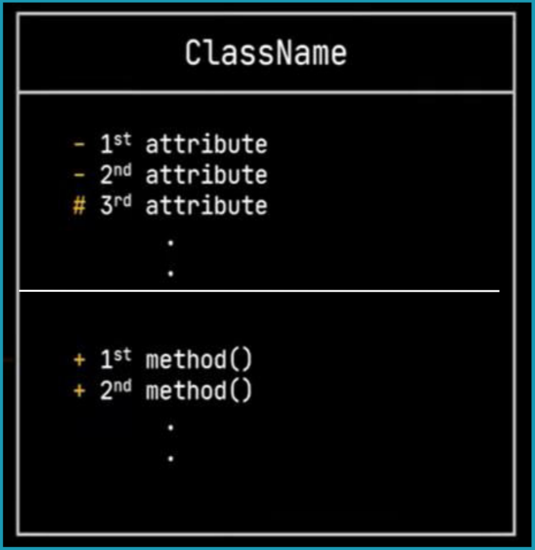
- Class diagrams offer a number of benefits for any organization. Use UML class diagrams to:
- Illustrate data models for information systems, no matter how simple or complex.
- Better understand the general overview of the schematics of an application.
- Visually express any specific needs of a system and disseminate that information throughout the business.
- Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure.
- Provide an implementation-independent description of types used in a system that are later passed between its components.

UML: Unified Modeling Language

Basic components of a class diagram

The standard class diagram is composed of three sections:

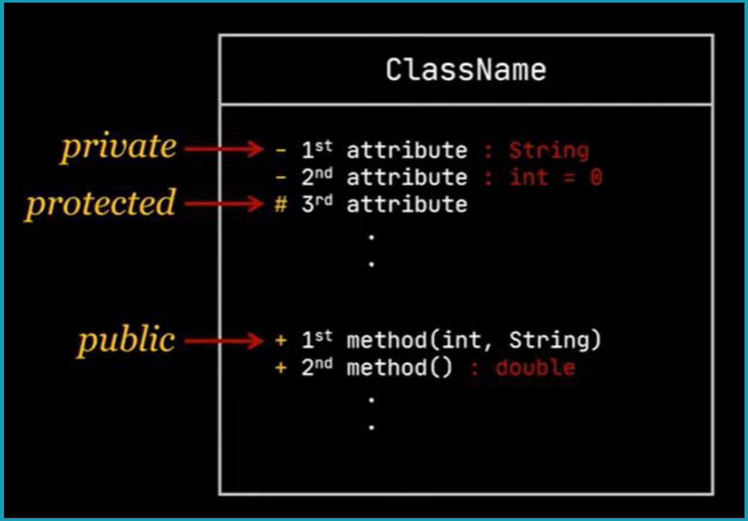
- **Upper section**: Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.
- **Middle section**: Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.
- **Bottom section**: Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.



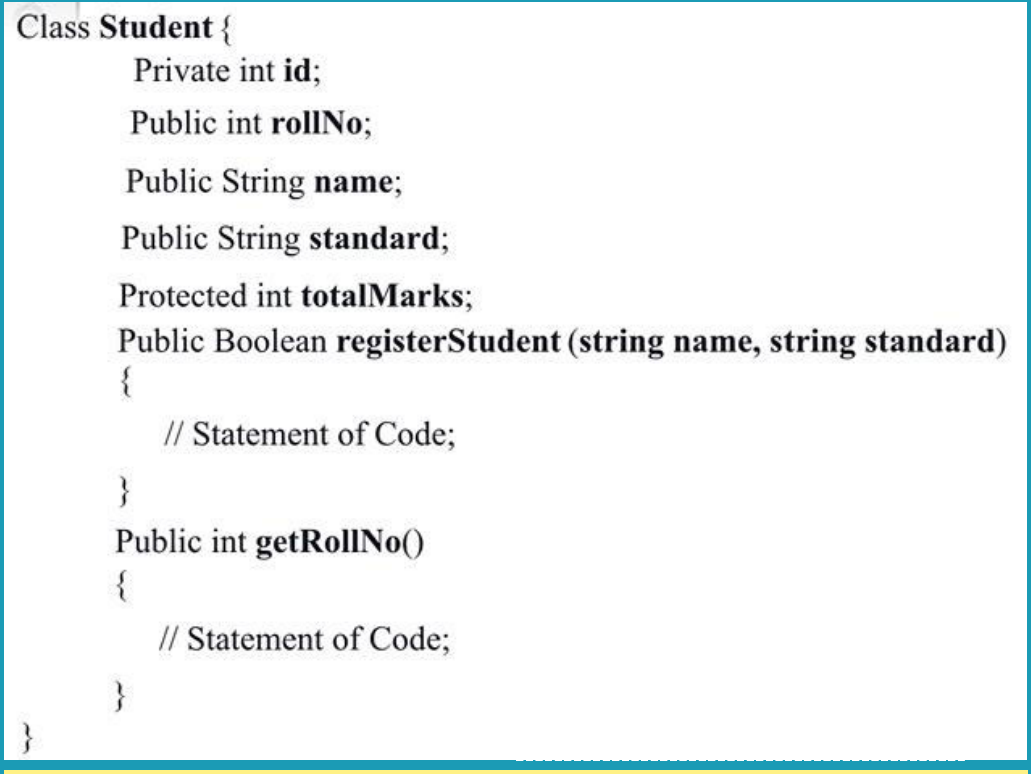
Member access modifiers

All classes have different access levels depending on the access modifier (visibility). Here are the access levels with their corresponding symbols:

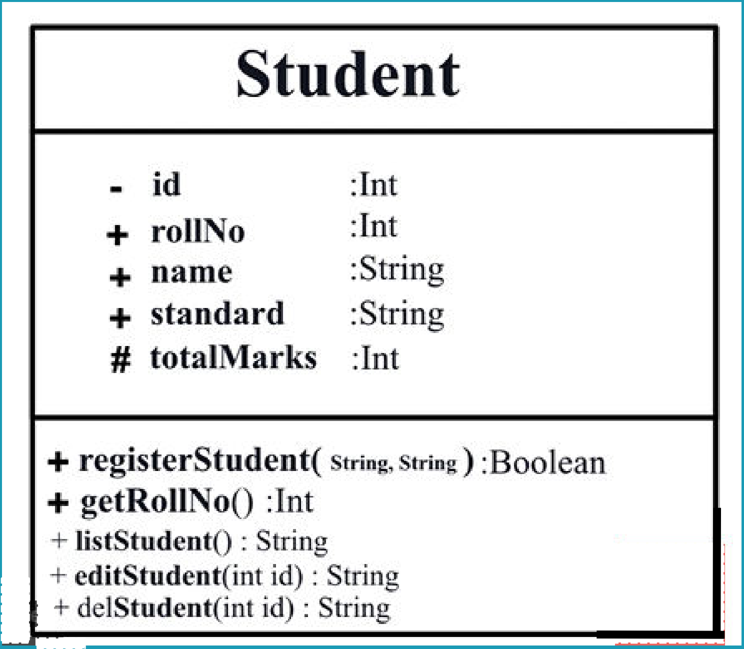
Public (+)
Private (-)
Protected (#)
Package (~)
Derived (/)
Static (underlined)



UML: Unified Modeling Language



```
Class Student {
         Private int id;
         Public int rollNo;
         Public String name;
         Public String standard;
         Protected int totalMarks;
         Public Boolean registerStudent (string name, string standard)
         {
             // Statement of Code;
         }
         Public int getRollNo()
         {
             // Statement of Code;
         }
}
```



```
                 Student
-  id            :Int
+  rollNo        :Int
+  name          :String
+  standard      :String
#  totalMarks    :Int

+ registerStudent( String, String ) :Boolean
+ getRollNo() :Int
+ listStudent() : String
+ editStudent(int id) : String
+ delStudent(int id) : String
```

The above program can be written form UML class diagram

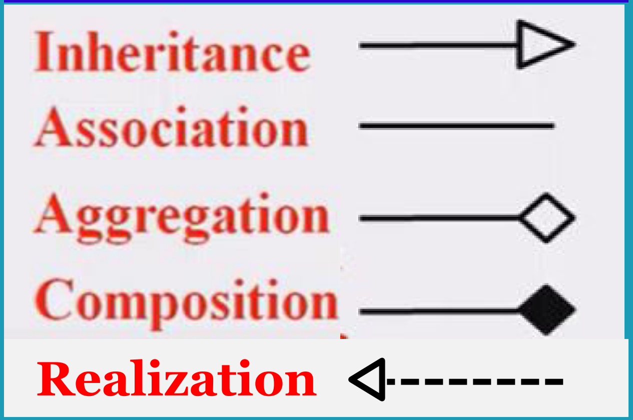
The class diagram contains all the classes that need to define while developing the system.

UML: Unified Modeling Language

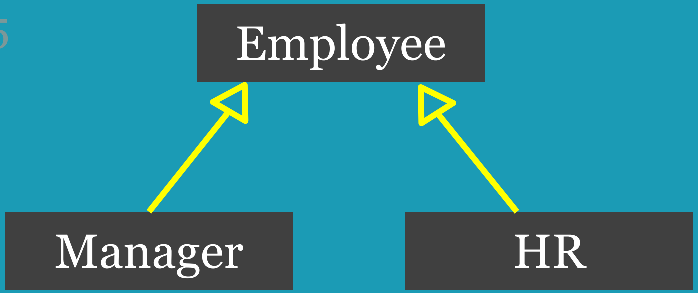
Relationship Types

1. Generalization
2. Association
   i. Aggregation
   ii. Composition
3. Realization

Relationship Types Symbol



1



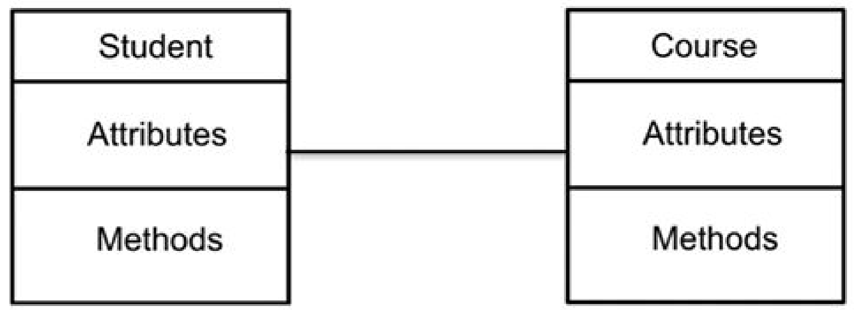
Example[Generalization]: here Manager & HR are also Employee so Manager & HR are general version of Employee. Attributes in Employee are available in Manager & HR in addition to their own attributes**. It is also referred to as inheritance**

UML: Association

What is Association?

When two classes communicate or passes information to each other . Draw a straight line between them.

Example of Association



Here Student and Course communicate to each other like one student has sever courses and one course has several students

Other Examples of Association

House and Person

Student and Membership

Student and Teacher

Airplane and Passenger

2

Exmaple:

Lets take an example of Doctor and Patient.

Multiple patients can associate with a single doctor and single patient can associate with multiple doctors, but there is no ownership between the objects and both have their own lifecycle. Both can create and delete independently.

UML: Unified Modeling Language

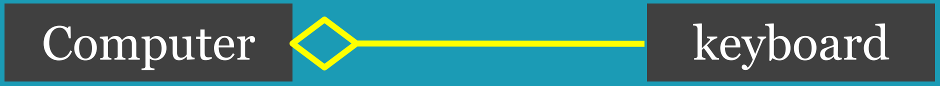
Types of Association

1. Aggregation
2. Composition

3

What is Aggregation?

Aggregation implies **a relationship where the child can exist independently of the parent**.

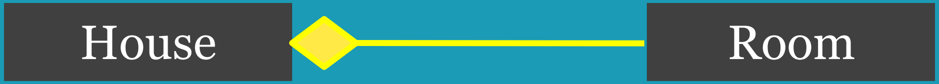


Here keyboard is a part of Computer but if Computer damage/doesn’t exits still Keyboard can exits and it can be a part of other computer

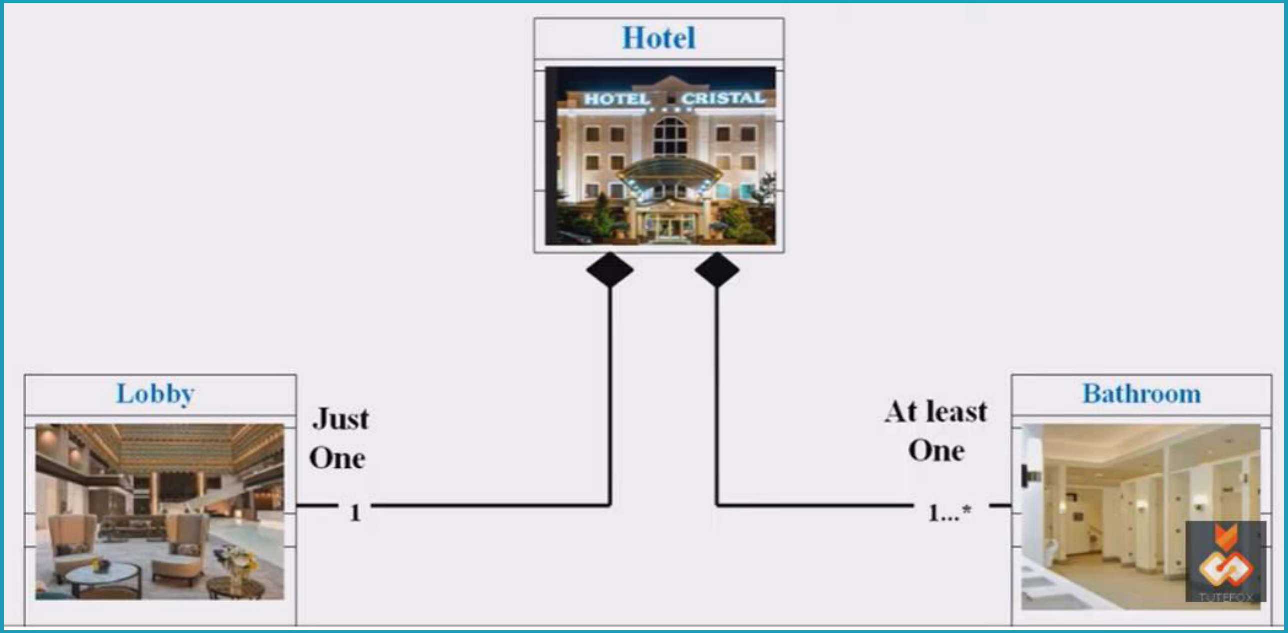
UML: Composition

What is Composition?

Composition implies a relationship where the child cannot exist independent of the parent.



Example: House (parent) and Room (child). Rooms don't exist separate to a House.



More Examples
* Person & Leg
* Whatsapp & WhGroup

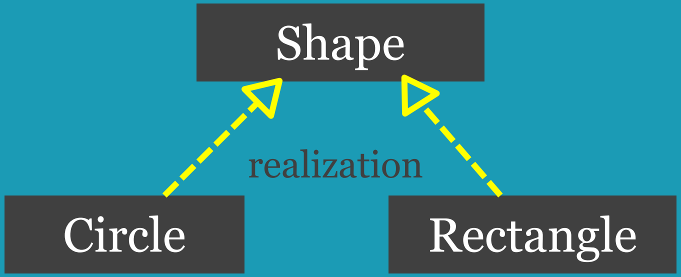
4

Example: If Hotel doesn't exist then Lobby and Bathroom wouldn't be exist

UML: Realization

What is Realization?

In UML modeling, a realization relationship is a relationship between two model elements, in which one model element (the client) realizes the behavior that the other model element (the supplier) specifies. Several clients can realize the behavior of a single supplier.

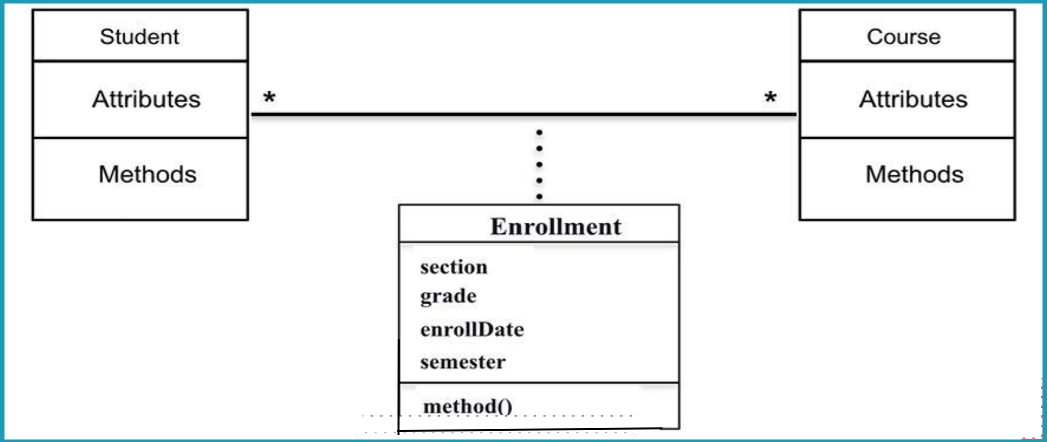


Example: shape is abstract, all implementations are to be done in child classes

UML: Unified Modeling Language

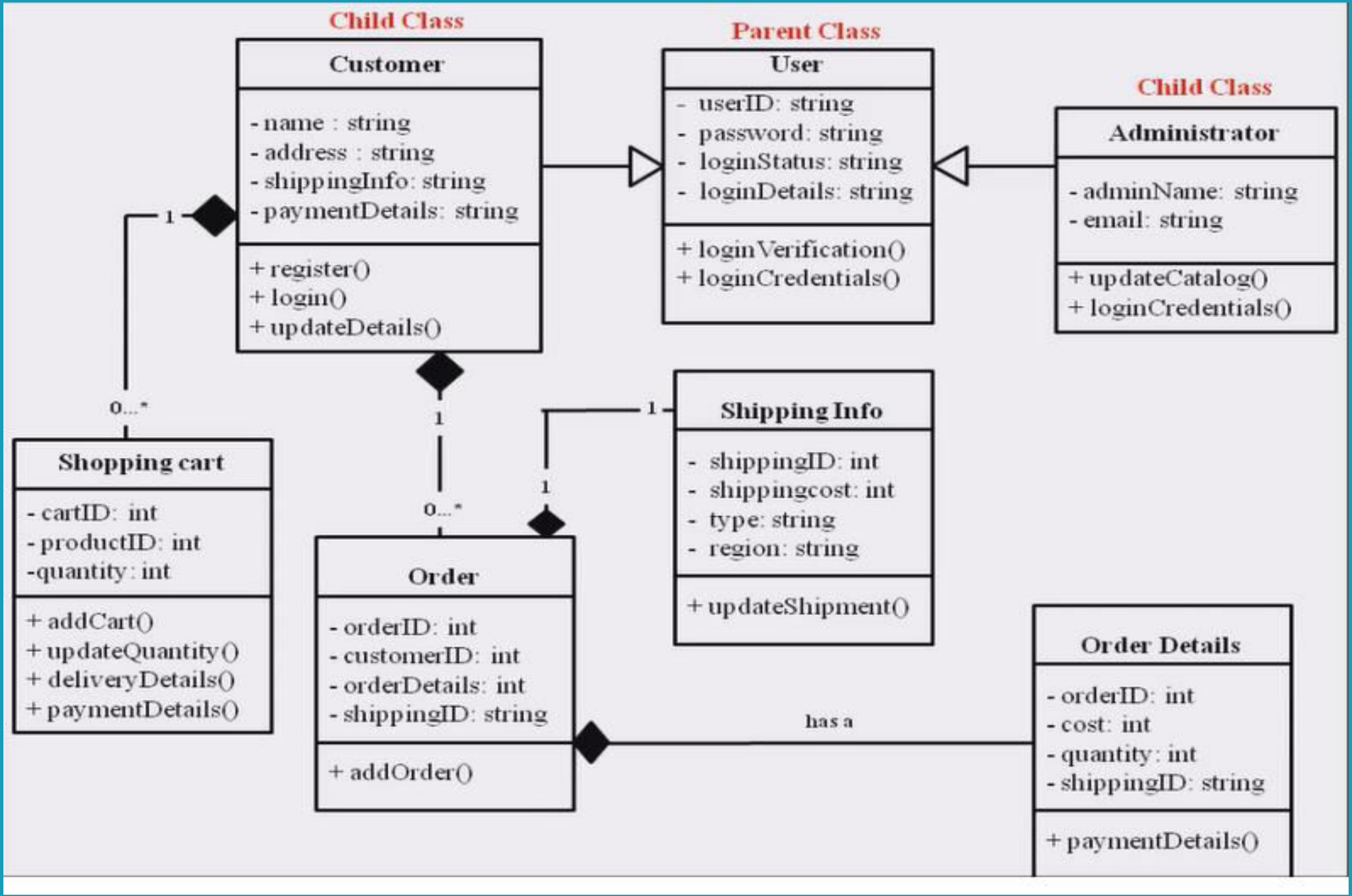
What is Association class?

n UML diagrams, an association class is **a class that is part of an association relationship between two other classes**. You can attach an association class to an association relationship to provide additional information about the relationship



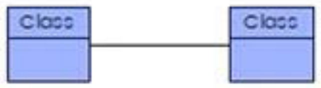
Here Enrollement class keeps additional information when association between Student and Course become established

UML: Online Shopping Example

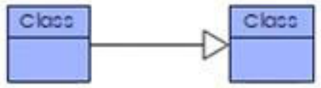


UML: Unified Modeling Language

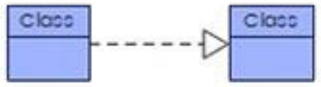
Relationship between classes



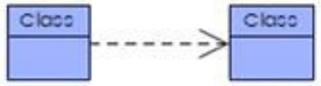
Bidirectional Association – A bilateral association is represented by a straight line connecting two classes. It simply demonstrates that the classes are aware of their relationship with each other.



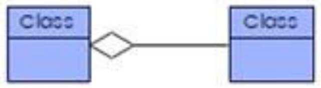
Inheritance – Indicate a "child-parent" relationship between classes. The child class is a specialized, sub-class of the parent.



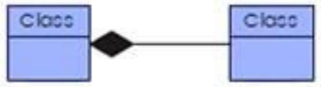
Realization – One class implements the behavior specified by another class.



Dependency – As the name suggests, one class depends on another. A dashed arrow shows this.



Aggregation – This represents a unilateral relationship between classes. One class is part of, or subordinate to, another. In this instance, the child and parent classes can exist independently.



Composition – It is a form of aggregation where one class is dependent on another. One class is a part of the other. In this instance, the child classes and parent classes cannot exist independently.

UML: Unified Modeling Language

C++ program for Aggregation

```
//UML Aggregation
//If Company class is deleted then
//Employee class is still on
#include <iostream>
#include<bits/stdc++.h>
#include<string>

using namespace std;

class Employee{
  string ename;
  public:
      Employee(string s){
       ename=s;
      }
      string getName(){
       return ename;
      }
      ~Employee(){
       cout<<"Employee is closed"<<endl;
     }
};
```

```
class Company{
 string cname;
 Employee* emp; //Aggregation
 public:
     Company(string s, Employee* e){
       cname=s;
       emp=e;
     }
     void getData(){
      cout<<"Company Name: "<<cname<<endl;
      cout<<"Employee Name: "<<emp->getName()<<endl;
     }
     ~Company(){
       cout<<"Company is closed"<<endl;
     }
};
int main(){
  Employee e("Zaman");
  cout<<"Name="<<e.getName()<<endl;
  {
   Company c("HP",&e);
   c.getData();
   //Company object c is closed but
   //Employee object e is still on
  }

}
```

Here Company is parent (whole) and Employee is child (part) as inside Company a data member emp of Employee is declared . When Company object is destroyed then Employee object is NOT destroyed

UML: Unified Modeling Language

C++ program for Decomposition

```
//UML Decomposition
#include <iostream>
#include<string>
using namespace std;

class Birthday{
 int day;
 int month;
 int year;
 public:
     Birthday(int d=0,int m=0,int y=0){
      day=d;
      month=m;
      year=y;
      cout<<"Birthday Constructor is called"<<endl;
     }
     void setDate(int d,int m,int y){
      day=d;
      month=m;
      year=y;
     }
     void Display(){
      cout<<"Date:
"<<day<<"/"<<month<<"/"<<year<<endl;
     }
     ~Birthday(){
       cout<<"Birthday destructor is called"<<endl;
     }
};
```

```
class Person{
    string name;
    Birthday obj;
    public:
     Person(string s,int d=0,int m=0,int y=0){
      name=s;
      obj.setDate(d,m,y);
      cout<<"Person Constructor is called"<<endl;
     }
     void Display(){
      cout<<"Name: "<<name<<endl;
      obj.Display();
     }
    ~Person(){
       cout<<"Person destructor is called"<<endl;
     }
};

int main(){
 Person p("Ali",3,10,2000);
 p.Display();
 //Nothing declare about Birhtday class directly
 //But when Person object destroyed Birthday class
 //also destroyed
}
```

Here Person is parent (whole) and Birthday is child (part) as inside Person a data member obj of Birthday is declared . When Person object is destroyed then Birthday object is also is destroyed

THANK YOU