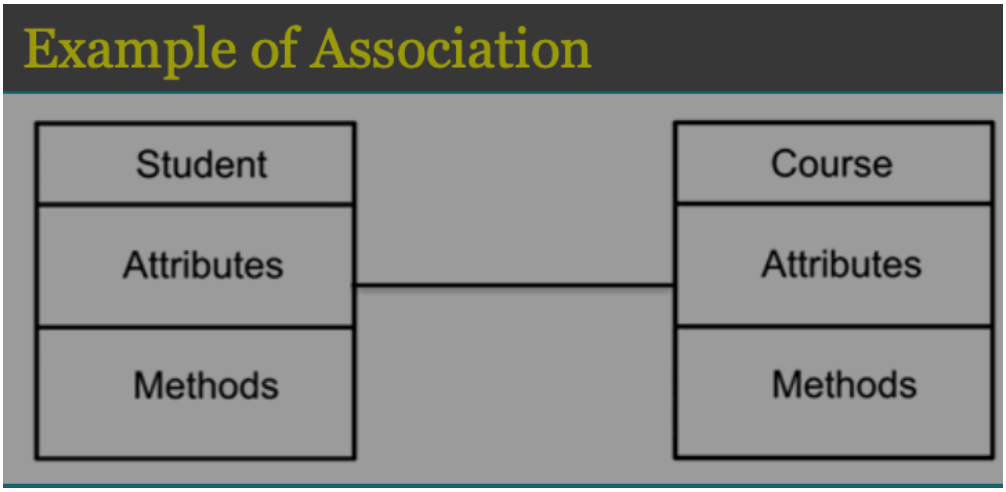
6 - UML

Association

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



Types of Association:

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

- **Composition** -> a relationship where the child cannot exist independently of the parent.
  - 

Aggregation

```
#include <iostream>
#include <string>

using namespace std;

class Employee{
    string ename;
    public :
```

```
        Employee(string s){
            ename = s;
        }
        string get_name(){
            return ename;
        }
        ~Employee(){
            cout << "Employee is closed.\n";
        }
};

class Company{
    string cname;
    Employee *emp;
    public :
        Company(string s, Employee* e){
            cname = s;
            emp = e;
        }
        void get_data(){
            cout << "Company name = " << cname << '\n';
            cout << "Employee name = " << emp->get_name() << '\n';
        }
        ~Company(){
            cout << "Company is closed.\n";
        }
};

int main(){
    Employee e("priashis");
    Company c("Apple", &e);
    c.get_data();
}
```

Output:

```
Company name = Apple
Employee name = priashis
Company is closed.
Employee is closed.
```

- Employee is a part of Company, but destroying Company does not destroy Employee.

Decomposition / Composition

```cpp
#include <iostream>
#include <string>

using namespace std;

class Employee{
    string ename;
    public :
        Employee(string s){
            ename = s;
        }
        string get_name(){
            return ename;
        }
        ~Employee(){
            cout << "Employee is closed.\n";
        }
};

class Company{
    string cname;
    Employee emp;
    public :
        Company(string s, string name) : emp(name){
            cname = s;
        }
        void get_data(){
            cout << "Company name = " << cname << '\n';
            cout << "Employee name = " << emp.get_name() << '\n';
        }
        ~Company(){
            cout << "Company is closed.\n";
        }
};

int main(){
    Company c("Apple", "priashis");
    c.get_data();
}
```

Output:

```
Company name = Apple
Employee name = priashis
```

```
Company is closed.
Employee is closed.
```

- Company contains an Employee object (composition). When the Company object is destroyed, the Employee object is automatically destroyed because its lifetime is tied to Company.

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.