

Association

A relationship between two classes

Association

- 01** **Objects and Classes**
- 02** **Relationships**
- 03** **Association**
- 04** **Types of Association**
- 05** **Examples**

Objects & Classes

Objects are some entities that have some features and behavior. Many objects get together to make a class. In a class many objects are present.


The Class can be said as the “**blueprint**” for the creation of objects. Similar to real life, objects in programming also have relationships.

Relationship Between Classes

Class relationships in C++ define the special relationships among different kinds of classes.

For example, there is a special relationship between a class named Vehicle and a class Car:

A Car is a type of Vehicle.



Relationship Between Classes

We have studied three types of relationship between classes :

Composition

Aggregation

Association

Association

Association in C++ is a relationship between two classes where one class uses the functionalities provided by the other class. In other words, an association represents the connection or link between two classes.

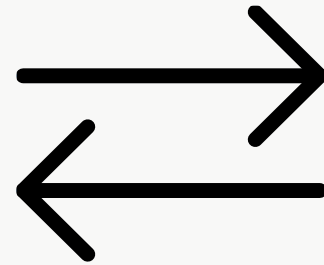
Association

Some points to remember when we talk about association relationship:

- A kind of connection between two objects.
- The relationship may be in one direction or in both directions that is either both are dependent on each other or anyone on the other one.

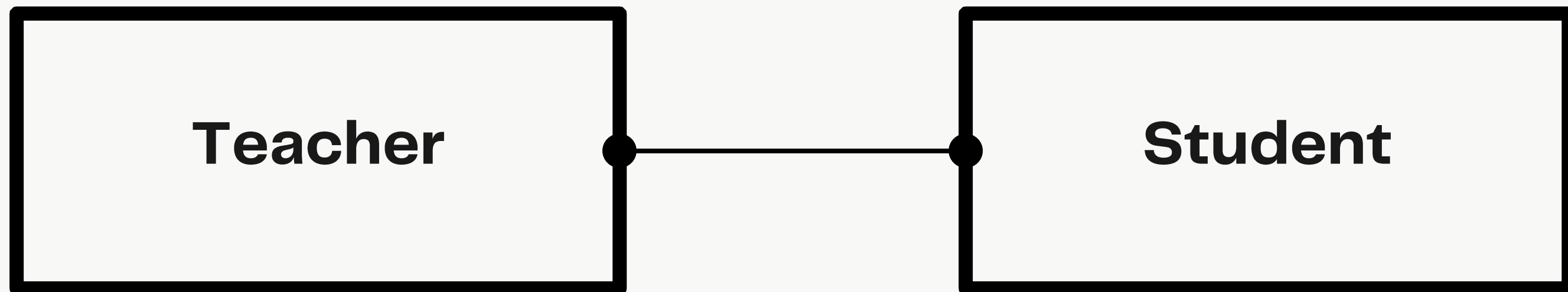
Example:

A Relationship between a teacher and a student.



Example:

The relationship of association will be shown in this way



Types of Association

We have studied three types of association between classes :

ONE-TO-ONE

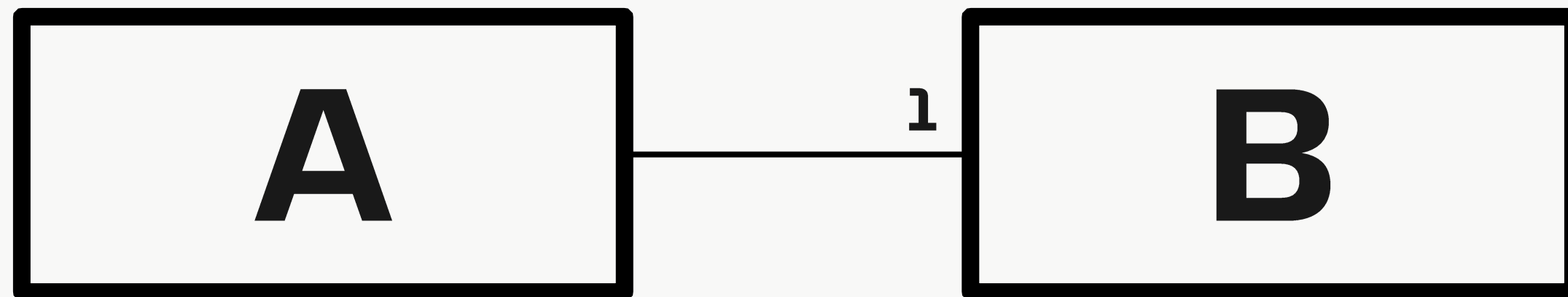
ONE-TO-MANY

MANY-TO-ONE

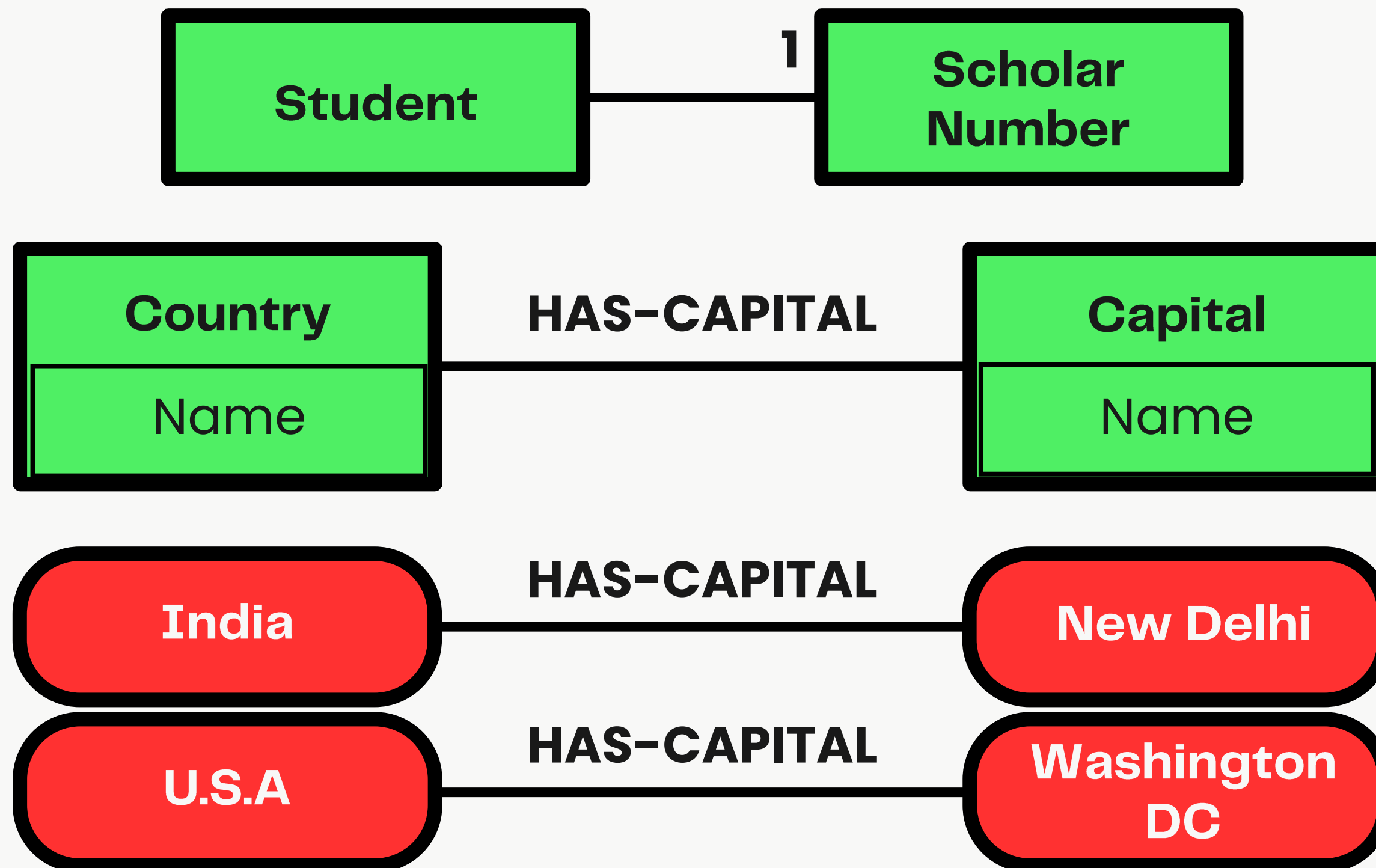
MANY-TO-MANY

1. One-to-One

- One object is associated with one or more object at a time.
- The symbol '1' is used at the end of the link which symbolises that there is only one object present in the association.



Example of One-to-one



```
class Professor;
```

```
class Student {
```

```
public:
```

```
    Student(int student_id, const string& name) : student_id(student_id), name(name), advisor(nullptr) {}
```

```
    void setAdvisor(Professor* professor);
```

```
private:
```

```
    int student_id;
```

```
    string name;
```

```
    Professor* advisor;
```

```
};
```

```
class Professor {
```

```
public:
```

```
    Professor(int professor_id, const string& name) : professor_id(professor_id), name(name) {}
```

```
private:
```

```
    int professor_id;
```

```
    string name;
```

```
};
```

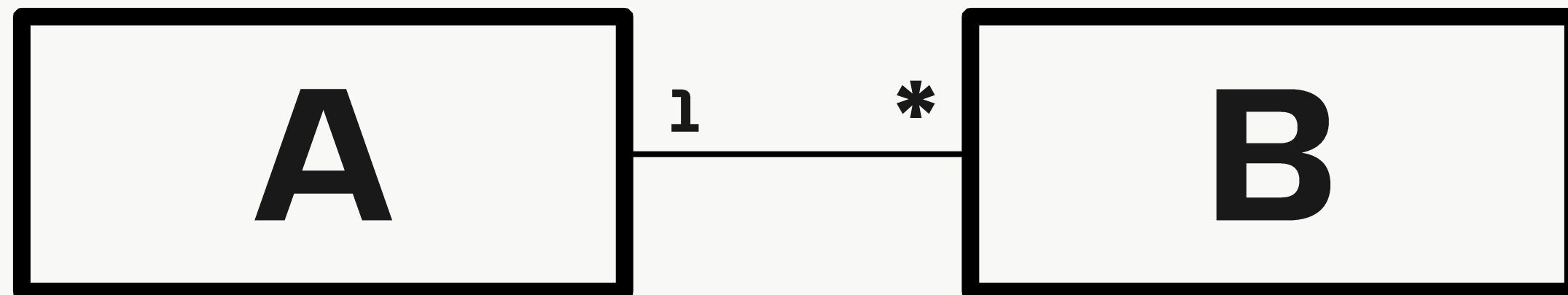
```
void Student::setAdvisor(Professor* professor) {
```

```
    advisor = professor;
```

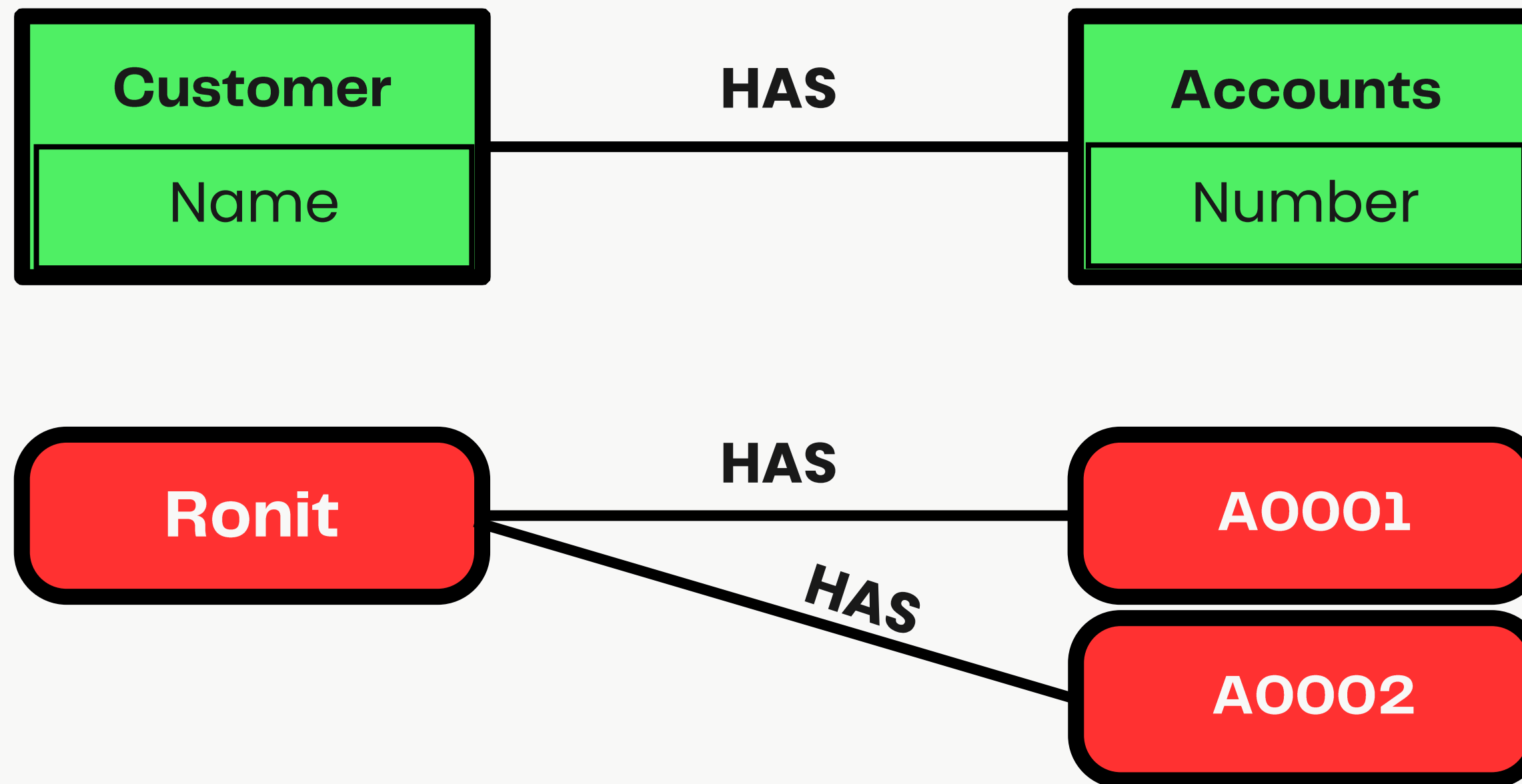
```
}
```

2. One-to-Many

- Only one object is associated with another object at a time.
- The symbol '1' and '*' is used at the end of the link in one-to-many association.



Example of One-to-Many



```
class Course {
public:
    Course(int course_id, const string& name) : course_id(course_id), name(name),
        enrolledStudents(nullptr),
        numEnrolled(0) {}

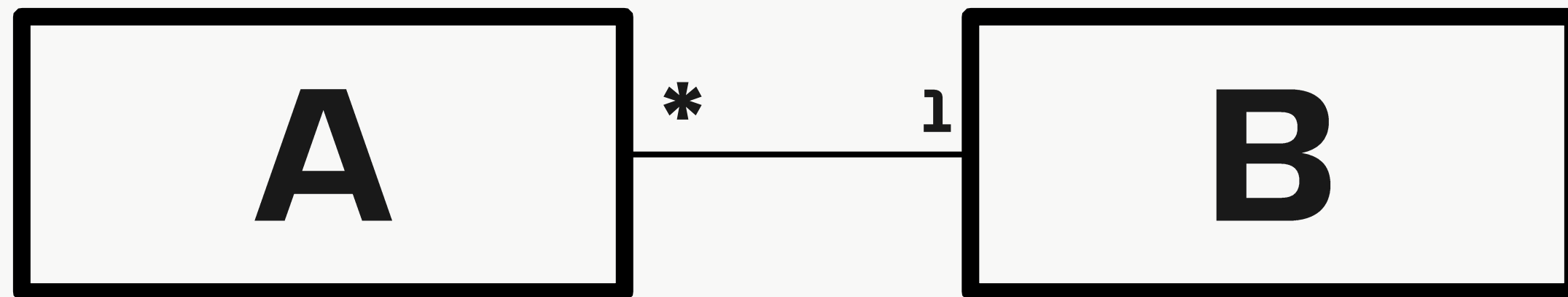
    void enrollStudent(Student* student);

private:
    int course_id;
    string name;
    Student** enrolledStudents;
    int numEnrolled;
};

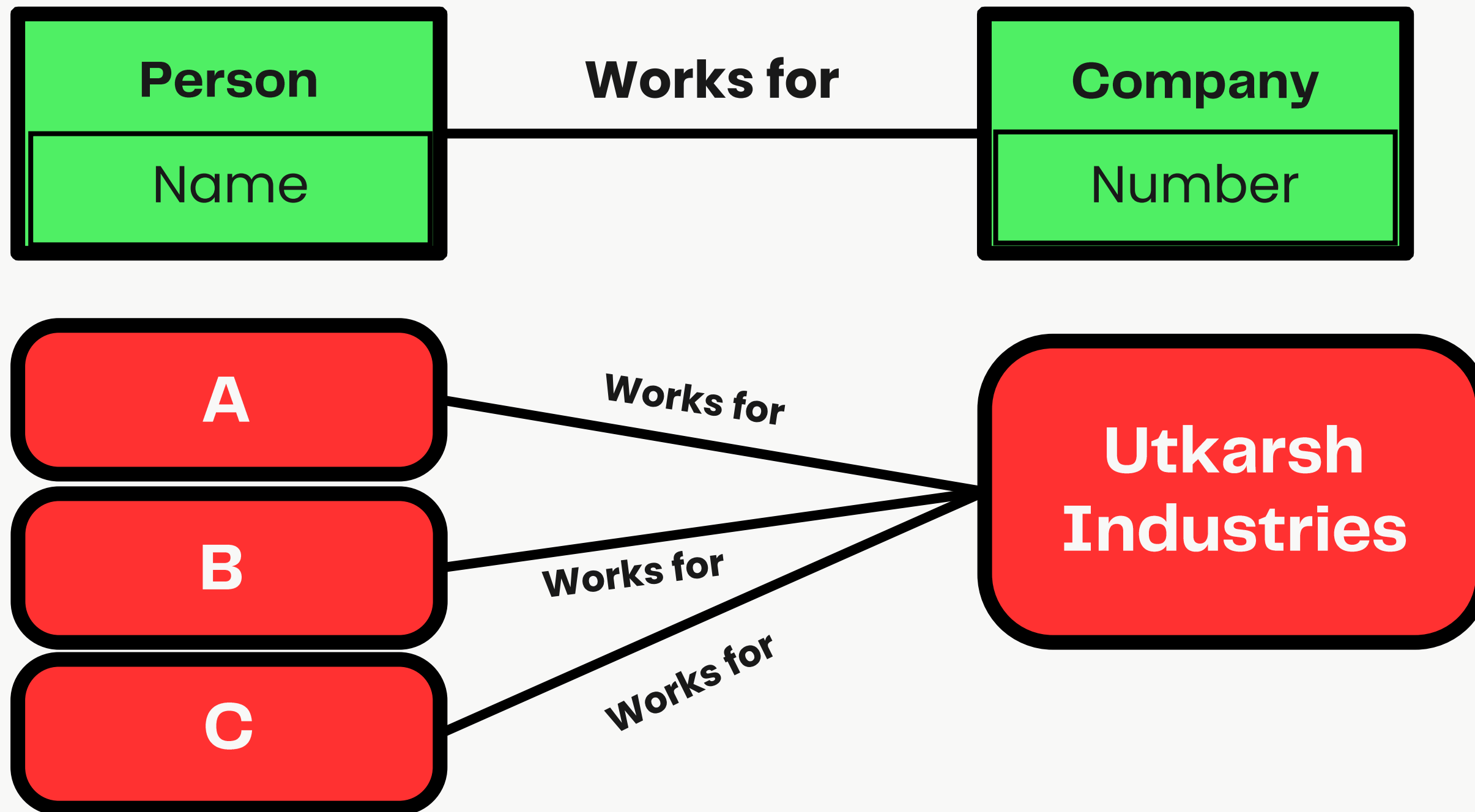
void Course::enrollStudent(Student* student) {
    const int maxStudents = 50;
    if (numEnrolled < maxStudents) {
        if (!enrolledStudents) {
            enrolledStudents = new Student*[maxStudents];
        }
        enrolledStudents[numEnrolled++] = student;
    }
}
```


3. Many-to-One

- Many objects are associated with one object at a time.
- The symbol '*' and '1' is used at the end of the link in many-to-one association.



Example of Many-to-One



```
class Professor;
```

```
class Student {
```

```
public:
```

```
    Student(int id, const string& name) : id(id), name(name), advisor(nullptr) {}
```

```
    void setAdvisor(Pprofessor* professor);
```

```
private:
```

```
    int id;
```

```
    string name;
```

```
    Pprofessor* advisor;
```

```
};
```

```
class Pprofessor {
```

```
public:
```

```
    Pprofessor(int id, const string& name) : id(id), name(name) {}
```

```
    void adviseStudent(Student* student);
```

```
private:
```

```
    int id;
```

```
    string name;
```

```
    vector<Student*> advisedStudents;
```

```
};
```

```
void Student::setAdvisor(Pprofessor* professor) {
```

```
    advisor = professor;
```

```
}
```

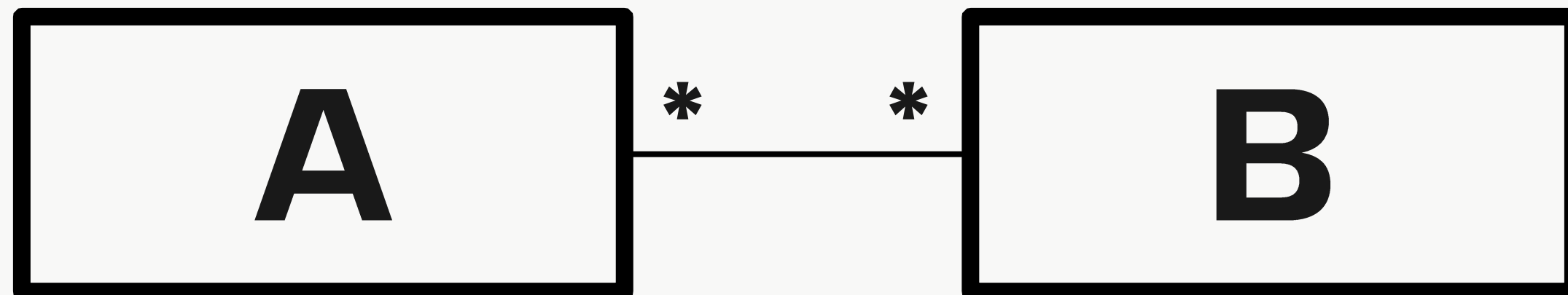
```
void Pprofessor::adviseStudent(Student* student) {
```

```
    advisedStudents.push_back(student);
```

```
}
```

4. Many-to-Many

- Many objects are associated with Many object at a time.
- The symbol '*' and '*' is used at the end of the link in many-to-many association.



Example of Many-to-One



```
class Course;
```

```
class Student {
```

```
public:
```

```
    Student(int id, const string& name) : id(id), name(name) {}
```

```
    void enrollInCourse(Course* course);
```

```
private:
```

```
    int id;
```

```
    string name;
```

```
    vector<Course*> enrolledCourses;
```

```
};
```

```
class Course {
```

```
public:
```

```
    Course(int id, const string& name) : id(id), name(name) {}
```

```
    void enrollStudent(Student* student);
```

```
private:
```

```
    int id;
```

```
    string name;
```

```
    vector<Student*> enrolledStudents;
```

```
};
```

```
void Student::enrollInCourse(Course* course) {
```

```
    enrolledCourses.push_back(course);
```

```
    course->enrollStudent(this);
```

```
}
```

```
void Course::enrollStudent(Student* student) {
```

```
    enrolledStudents.push_back(student);
```

```
}
```

Thankyou

Ved Jain
Murtaza Shahid