Composição, Agregação e Associação

- Diferença entre eles
  - https://pt.stackoverflow.com/questions/86715/qual-a-diferença-entre-associação-agregação-e-composição-em-oop
- Implementações em C++
  - Capítulo 10 do learn cpp
  - https://www.learncpp.com/

Associação

- A associação entre dois objetos ocorre quando eles são completamente independentes entre si mas eventualmente estão relacionados.
- Ela pode ser considerada uma relação de muitos para muitos.
- Não há propriedade nem dependência entre eles.
- Relação Aluno - Professor
  - Um aluno pode ter vários professores e um professor pode ter vários alunos.
  - Um não depende do outro para existir.

Composição

- A composição é uma agregação que possui dependência entre os objetos.
- Se o objeto principal for destruído, os objetos que o compõe não podem existir mais.
- Normalmente é a classe(Todo) que instancia os objetos que guarda(Parte).
- Uma nota fiscal tem vários itens. Todo(Nota) contém Partes(Item).

```
Toda vez que temos composição,
significa que a parte não existe sem o todo.
```

Agregação

- Não existe a relação de posse.
- Os objetos não membros não são criados pela instância, normalmente são recebidos.
- Não existe relação de N x N, normalmente é 1 x N.
- Um professor pode pertencer a vários departamentos, mas só pode estar alocado em 1.

```
Toda vez que temos agregação, significa que a parte
pode ser compartilhada entre vários objetos.
```

Resumindo

Nome	Hierarquia	Posse
Associação	Não	O Todo e Parte têm tempo de vida próprios.
Composição	Sim	Todo morre -> Parte morte.
Agregação	Sim	Todo morre -> Parte vive.

```
Essas relações não dependem da multiplicidade.
Posso tem 1 x N, mas também 1 x 1.
```

Composição 1 x 1

Isso é composição.

```
class Pos{
    int x;
    int y;
};

class Creature{
    string name;
    Pos location;
};
```

Composição 1 x N

Isso ainda é composição.

```
class Mapa{
    vector<Pos*> buracos;

    void addBuraco(int x, int y){
        buracos.push_back(new Pos(x, y));
    }

    ~Mapa(){
        for(auto * buraco : buracos)
            delete buraco;
    }
}
```

Agregação 1 x N

Se o buraco for permanecer, mesmo depois de destruir o mapa.

```
class Mapa{
    vector<Pos*> buracos;

    void addBuraco(Pos * pos){
        buracos.push_back(pos);
    }

    ~Mapa(){
    }
}
```

Agregação 1 x 1

O carro(Todo) tem um motor(Parte), mas não a posse dele.

```
class Motor {
    int potencia;
};
class Carro{
    Motor * motor;
    Carro(Motor * motor){
        this->motor = motor;
    }
    ~Carro(){
    }
};
int main(){
    Motor motor;
    Carro carro(&motor);
}
```

Associação

- Em c++ você pode fazer a costura com um só ponto de inserção da relação.
- https://www.learncpp.com/cpp-tutorial/10-4-association/

```
class Medico; //para se livrar da dependência cíclica
class Paciente{
private:
    string name;
    vector<Medico *> medicos;
    void addMedico(Medico *med){
        medicos.push_back(med);
    }
public:
    friend class Medico;
    //nao consigo mostrar o nome dos pacientes aqui
    string toString();
};
```

```
class Medico{
private:
    string name;
    vector<Paciente> * pacientes;
public:
    void addPaciente(Paciente * pac){
        pacientes.push_back(pac);
        pac->addMedico(this);
    }
    string toString(){
        cout << name;
        for(auto * pac : pacientes)
            cout << pac->name;
    }
};

void Paciente::toString(){
    cout << name;
    for(auto * med : medicos)
        cout << med->name;
}
```

Associação com múltipla entrada

```
class Medico;
class Paciente{
    vector<Medico *> medicos;
public:
    void addMedico(Medico *med){
        if(med nao existe em medicos){
            medicos.push_back(med);
            med->addPaciente(this);
        }
    }
};
class Medico{
    vector<Paciente> * pacientes;
public:
    void addPaciente(Paciente * pac){
        if(pac nao existe em pacientes){
            pacientes.push_back(pac);
            pac->addMedico(this);
        }
    }
};
```