Composição / Agregação / Associação

Programação Orientada a Objetos

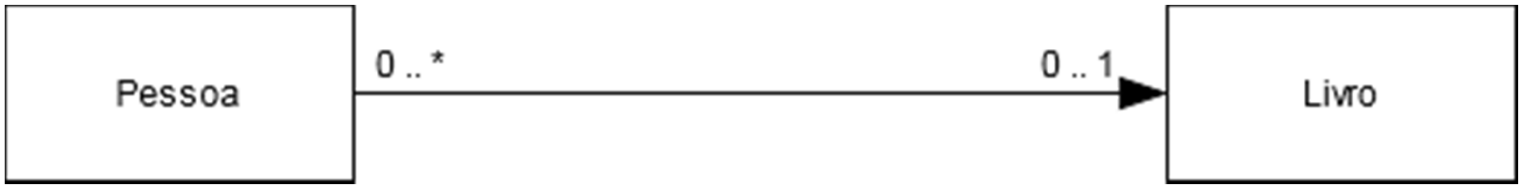
Aula 06

Prof. Diógenes Furlan

Associação

- Representam vínculos de relacionamentos
- Objetos relacionados não necessitam ser do mesmo tipo
- Vida independente
- Associações binárias e ternárias

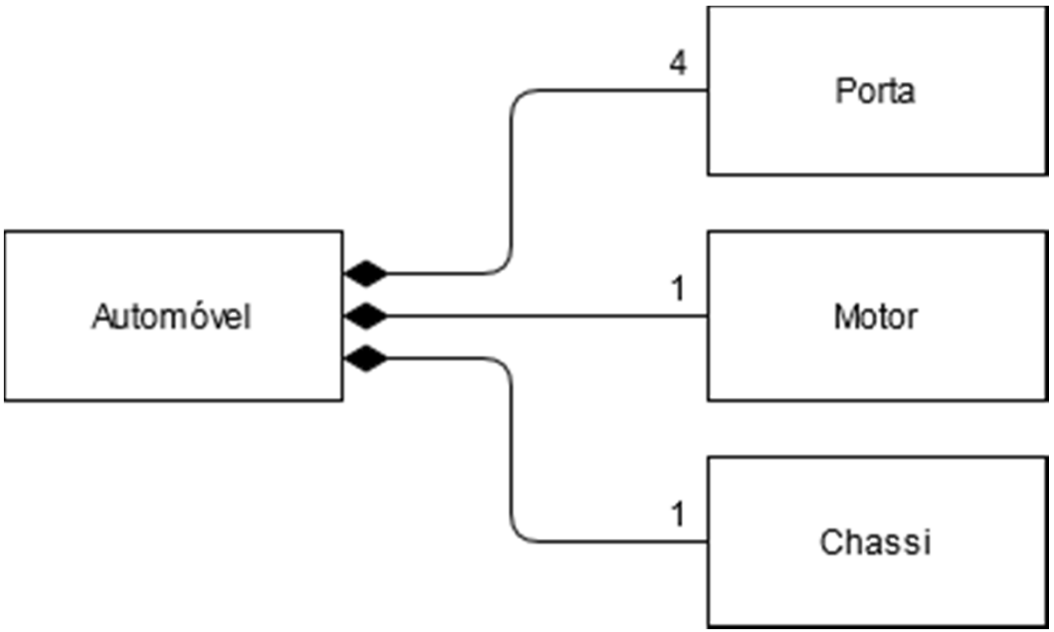
Associação



Composição

- Um objeto pode é composto por partes (objetos componentes)
  - Associação todo / partes
- O objeto composto NÃO existe sem suas partes (vidas dependentes)
  - Construtor do objeto composto instancia os objetos componentes
- Objetos componentes NÃO podem fazer parte de outros objetos compostos

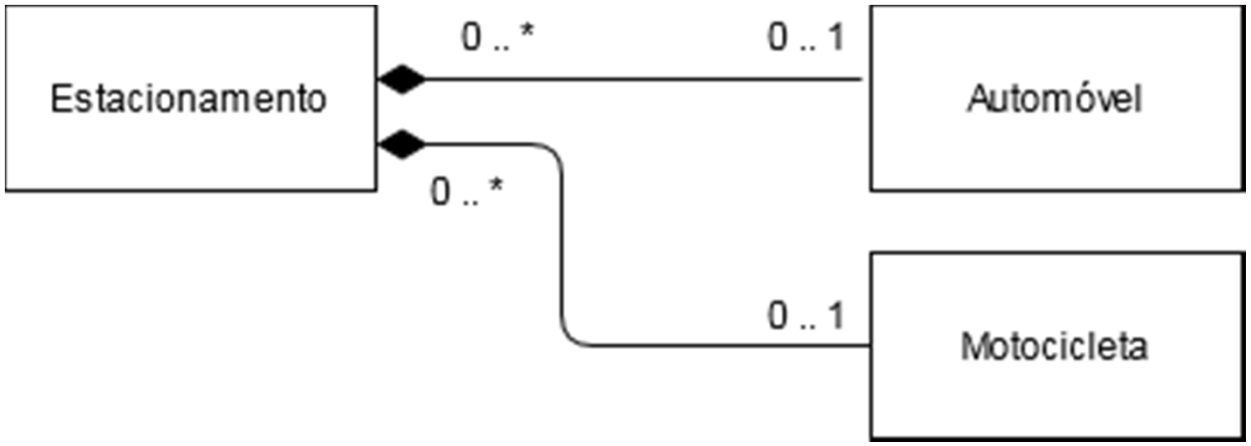
Composição



Agregação

- Associação todo / parte mais fraca
  - Listas de elementos, conjuntos
- Membros de uma agregação são de um mesmo tipo (ou categoria de tipos)
- Objeto agregado pode existir sem suas partes
- Objetos componentes podem fazer parte de diversas agregações

Agregação



EXEMPLO 1 – COMPOSIÇÃO

Exemplo

- Objetos podem ser compostos por outros objetos.

```
class Data {
   int dia;
   int mes;
   int ano;
};
```

```
class Pessoa {
   int codigo;
   string nome;
   Data dtNasc;
};
```

Exemplo

classe Data

```
void le() {
   cin >> this->dia;
   cin >> this->mes;
   cin >> this->ano;
}

void mostra() {
   printf("%02d/%02d/%04d",
           dia, mes, ano);
}
```

classe Pessoa

```
void le(int c, string n, Data d) {
   cin >> this->codigo;
   cin >> this->nome;
   dtNasc.le();
}

void mostra() {
   printf("\nPessoa:");
   printf("\ncodigo: %d", codigo);
   printf("\nnome: %s", nome);
   dtNasc.mostra();
}
```

Não funciona

classe Data

```
Data(int d, int m, int a) {
   this->dia = d;
   this->mes = m;
   this->ano = a;
}
```

classe Pessoa

```
Pessoa(int c, string n, int d,
int m, int a) {
   this->codigo = c;
   this->nome = n;
   this->dtNasc.dia = d;
   this->dtNasc.mes = m;
   this->dtNasc.ano = a;
}
```

```
Pessoa(int c, string n, int d,
int m, int a) {
   this->codigo = c;
   this->nome = n;
   this->dtNasc = Data(d,m,a);
}
```

```
Pessoa(int c, string n, Data d)
{
   this->codigo = c;
   this->nome = n;
   this->dtNasc = d;
}
```

Funciona

classe Pessoa

```
Pessoa(int c, string n, int d,
int m, int a)
: dtNasc(d,m,a) {
   this->codigo = c;
   this->nome = n;
}
```

EXEMPLO 2 – COMPOSIÇÃO

Exemplo Composição

```
class Somador {
    double valor;
public:
    Somador() { valor=0; }
    Somador(double v) { valor=v; }
    void   soma(double v) { valor += v; }
    double getValor() { return valor; }
};
```

Exemplo Composição

```
class Media {
    Somador valor;
    Somador qtd;
public:
    Media(): valor(), qtd() {}
    void adiciona(double v) {
        valor.soma(v);
        qtd.soma(1);
    }
    double getMedia() {
        return valor.getValor() / qtd.getValor();
    }
    void mostra() {
        printf("\ntotal: %lf", valor);
        printf("\nquantidade: %lf", qtd);
        printf("\nmedia: %lf", getMedia());
    }
};
```

Exemplo Composição

```
int main() {
    Media m1;
    double valor=0.0;

    while(valor>=0) {
        printf("\nDigite um valor positivo (negativo encerra): ");
        scanf("%lf", &valor);
        if(valor>=0)
            m1.adiciona(valor);
    }

    m1.mostra();
}
```

EXERCÍCIOS

Exercício 1 – Aula passada

- Crie uma classe chamada Ponto.
  - com 2 atributos: X e Y
- Crie dois construtores para esta classe:
  - um, sem parâmetros, que inicializa o ponto com 0,0
  - outro com 2 parâmetros
- Crie dois métodos chamados distancia:
  - um que recebe dois valores reais: X e Y
  - um que recebe outro Ponto

Classe Ponto

```
class Ponto
{
   float x;
   float y;

   public:

   Ponto();
   Ponto(float a, float b);
   void mostra(void);
   void move(float dx, float dy);
   float distancia(float x, float y);
   float distancia(Ponto &outro);
};
```

Exercício 2 (a)

- Crie uma classe Reta que tem como atributos dois objetos da classe Ponto.
- Crie 2 construtores para esta classe:
  - um que inicializa os pontos a partir de 2 pontos passados por parâmetro
  - um que inicializa os pontos a partir de 4 coordenadas reais passadas por parâmetro

Entregar!!

Exercício 2 (b)

- Crie uma função que calcula a distancia dos 2 pontos da reta. Reutilize código da classe Ponto.
- Mostre a reta, mostrando seus pontos e sua distancia.

Entregar!!

Exercício 2 (c)

- Quando seu programa ficar pronto acrescente funções membros para esta reta, tais como:
  - coeficiente angular (inclinação) → a
  - coeficiente linear (corta eixo y) → b
- Faça uma rotina que mostra a reta na forma
  - y=ax + b
- Defina uma função membro para a classe reta que retorna o ponto de intercessão com outra reta
  - **ponto reta::intercessao(reta a);**.
  - Não se esqueça de tratar os casos degenerados, tais como retas paralelas e coincidentes, use por exemplo mensagens de erro.

Entregar!!

Exercício 3

Crie uma classe Triângulo composta por três objetos de tipo Ponto.

Crie rotinas para:

- calcular os 3 lados do triangulo
- classificar o triangulo (equilátero, isósceles, escaleno)
- calcular a altura do triangulo