Relacionamento entre objetos

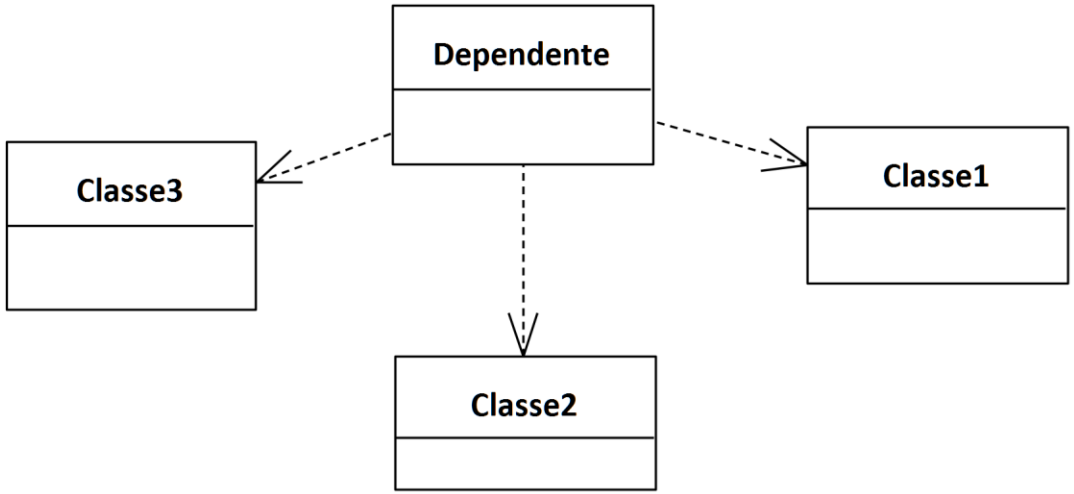
FURB
UNIVERSIDADE DE BLUMENAU

Bibliografia

- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML - guia do usuário**. Rio de Janeiro: Elsevier, 2005.
- BRAUDE, E. **Projeto de Software**. – Da programação à arquitetura. Porto Alegre: Bookman, 2005.
- GÉNOVA, G.; CASTILLO, C. R. D.; LLORENS, J. Mapping UML Associations into Java. **Journal Of Object Technology**, Zurich, v. 2, n. 5, p. 135-162, September-October 2003.
- MARTIN, R. C. **UML for Java Programmers**. New Jersey: Prentice-Hall, 2002.
- **Aggregation versus Composition, UML diagram, Java Example**. Site: http://www.apwebco.com/aggregation/AggregationComposition.html
- http://www.jugalpanchal.com/2013/07/differences-between-aggregation-and.html

Dependência

- Indica que objetos de duas classes se relacionam temporariamente para atender uma funcionalidade



- Define que se a classe na extremidade da seta mudar, isso afetará a classe dependente.

Dependência - Exemplo

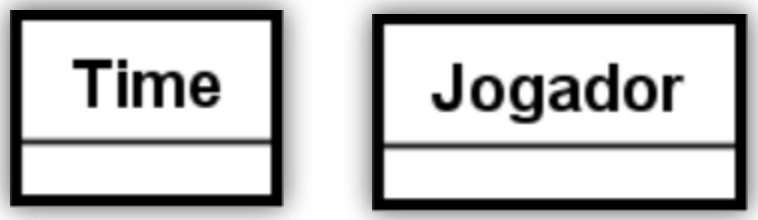
- No programa, a dependência pode ocorrer através de:
  - parâmetro de método
  - retorno de método
  - Variável local

```
public class Dependente {

    public void metodo1(Classe1 c) {
        ...
    }

    public Classe2 metodo2() {
        ...
    }

    public void metodo3() {
        Classe3 c;
        c = new Classe3();
    }

}
```

Especializações de associação

Especializações de associação

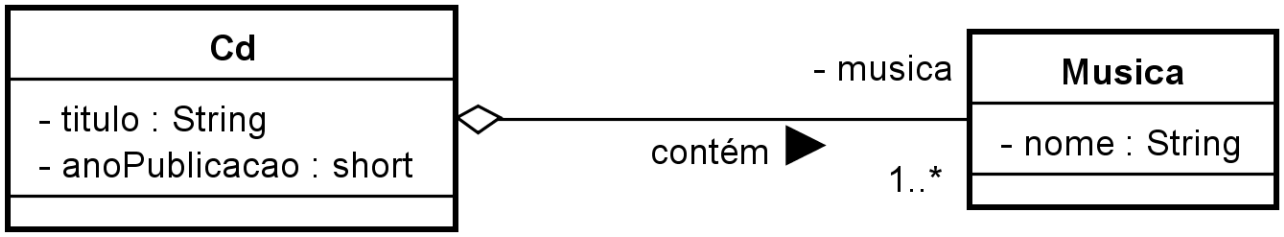
- São associações em que uma das classes faz papel de "todo" enquanto que a outra classe faz papel de "parte"
  - Chamado também de relacionamento todo/parte;
- Exemplo:



- As associações todo/parte podem ser classificadas em:
  - Agregação
  - Composição

Agregação

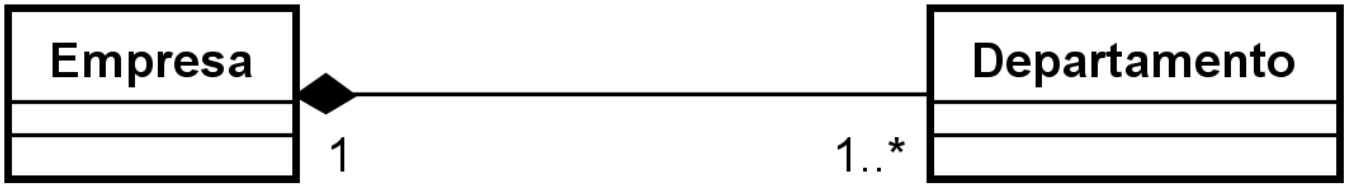
- A parte pode existir sem o todo
- Um objeto “parte” pode estar agregado a outros objetos “todo” (o objeto “parte” pode ser compartilhado com outros objetos)
- Exemplo:



- Na UML, é desenhado com um losango próximo à classe que faz o papel de “todo” – adorno “agregação”

Composição

- Similar à agregação, porém na composição o objeto que faz papel de “todo” é responsável por criar e destruir suas partes
  - Na composição, quando o objeto “todo” é destruído, suas partes também são
- Os objetos que fazem papel de parte não podem pertencer a mais de um todo
- Exemplo:



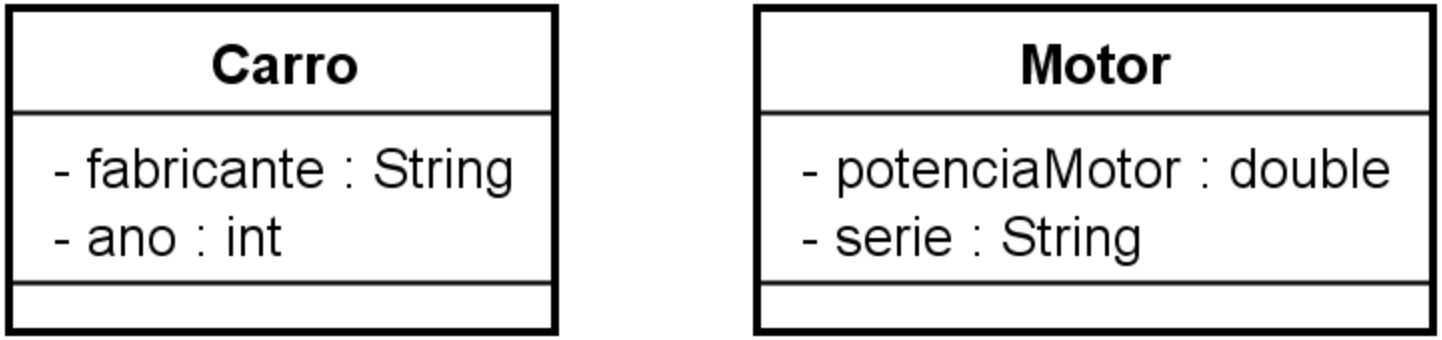
Composição

- Conforme (BRAUDE, 2005), na composição nenhum outro objeto pode referenciar o objeto composto. “O objeto composto só existe no escopo do objeto proprietário”.
- Na composição, as partes devem ser criadas a partir do objeto “todo”
- A variável que mantém o objeto *parte* deve ser privada
- Não pode existir *getter* ou *setter* para acessar o objeto *parte*
- Alguns autores denominam a composição com uma forma mais forte de agregação

Agregação X Composição

- Em Java, como a destruição é automática, a definição se utiliza agregação ou composição resume-se ao seguinte:
  - A parte é acessível a outros objetos além do *todo*?
    - Se sim: é agregação,
    - Se não: é composição.

Exemplo


Carro
- fabricante : String
- ano : int
Motor
- potenciaMotor : double
- serie : String

Exemplo de agregação

```
public class Carro {
    private String fabricante;
    private int ano;
    private Motor motor;

    public Carro(String fabricante, int ano, Motor motor) {
        this.fabricante = fabricante;
        this.ano = ano;
        this.motor = motor;
    }

    public String getFabricante() {
        return fabricante;
    }

    public int getAno() {
        return ano;
    }

    public Motor getMotor() {
        return motor;
    }

    public setMotor(Motor m) {
        this.motor = m;
    }
}
```

```
Motor m = new Motor(1.3, "344543533");
meuCarro = new Carro();
meuCarro.setMotor(m);
meuCarro = null;
```

OU

```
Motor m = new Motor(1.3, "344543533");
meuCarro = new Carro("Fiat", 2010, m);
meuCarro = null;
```

Exemplo de composição

```
public class Carro {
    private String fabricante;
    private int ano;
    private Motor motor;

    public Carro(String fabricante, int ano, double potencia, String serie) {
        this.fabricante = fabricante;
        this.ano = ano;
        this.motor = new Motor(potencia, serie);
    }

    public String getFabricante() {
        return fabricante;
    }

    public int getAno() {
        return ano;
    }

    public double getPotenciaMotor() {
        return motor.getPotencia();
    }

    public String getSerieMotor() {
        return motor.getSerie();
    }

}
```