Parte 6 – Diagramas de Classes em UML

Fernando Barros, Karima Castro, Luís Cordeiro,
Marília Curado, Nuno Pimenta

Os conteúdos desta apresentação baseiam-se nos materiais produzidos por António José Mendes para a unidade curricular de Programação Orientada a Objetos.

Quaisquer erros introduzidos são da inteira responsabilidade dos autores.

UML

- O desenvolvimento de programas orientados aos objetos assenta na construção de um *modelo*
- O modelo é uma *abstração* dos aspetos essenciais do problema
- UML (*Unified Modeling Language*) inclui um conjunto de ferramentas criadas para representar esses modelos
- Nesta disciplina vamos utilizar apenas alguns aspetos básicos, ficando para outras disciplinas uma abordagem mais aprofundada

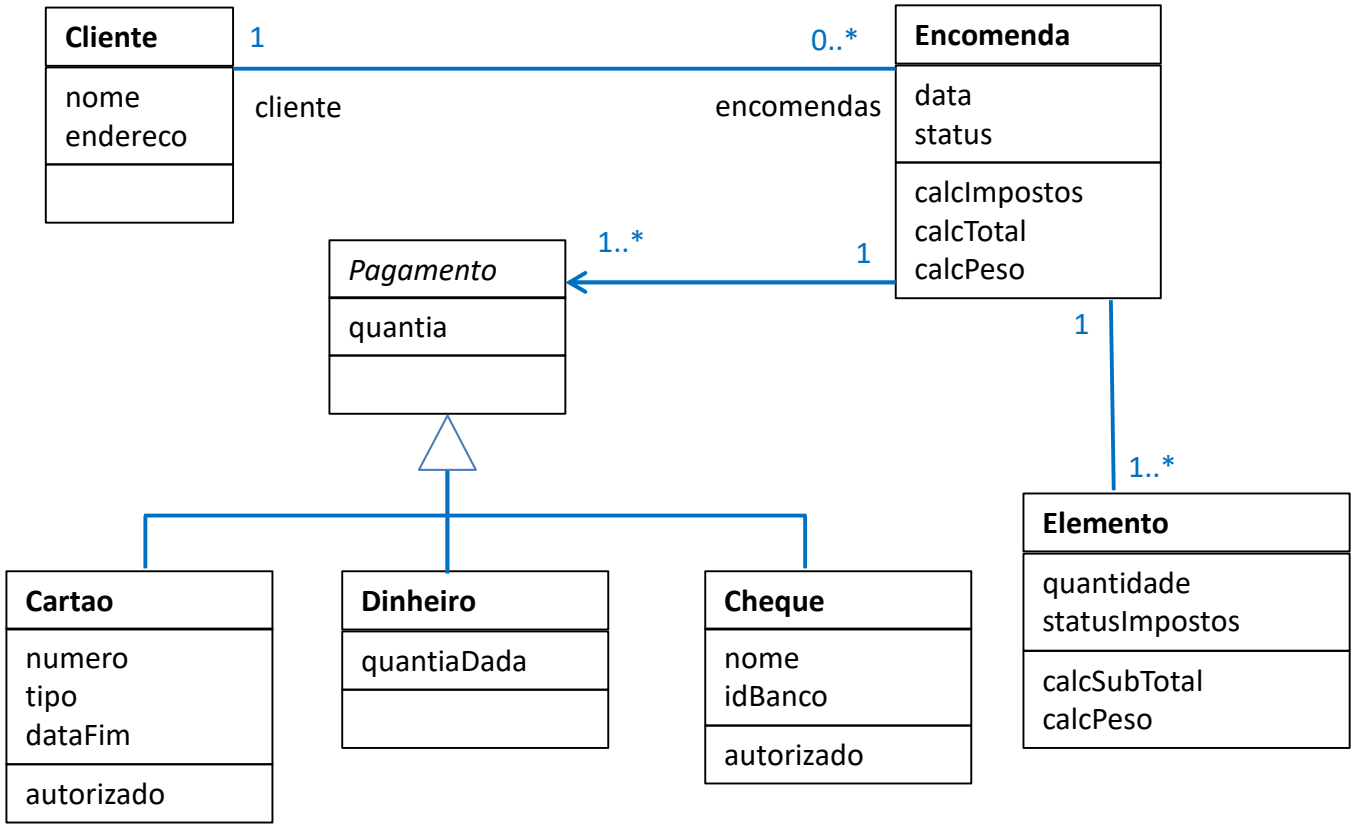
UML

- Em UML os modelos de software são representados por um conjunto de diagramas
- Estes são criados e utilizados durante a fase de análise e, posteriormente, como ferramenta de comunicação com os programadores
- Os modelos são constituídos por objetos (com os seus atributos e comportamentos) que interagem entre si através do envio de mensagens

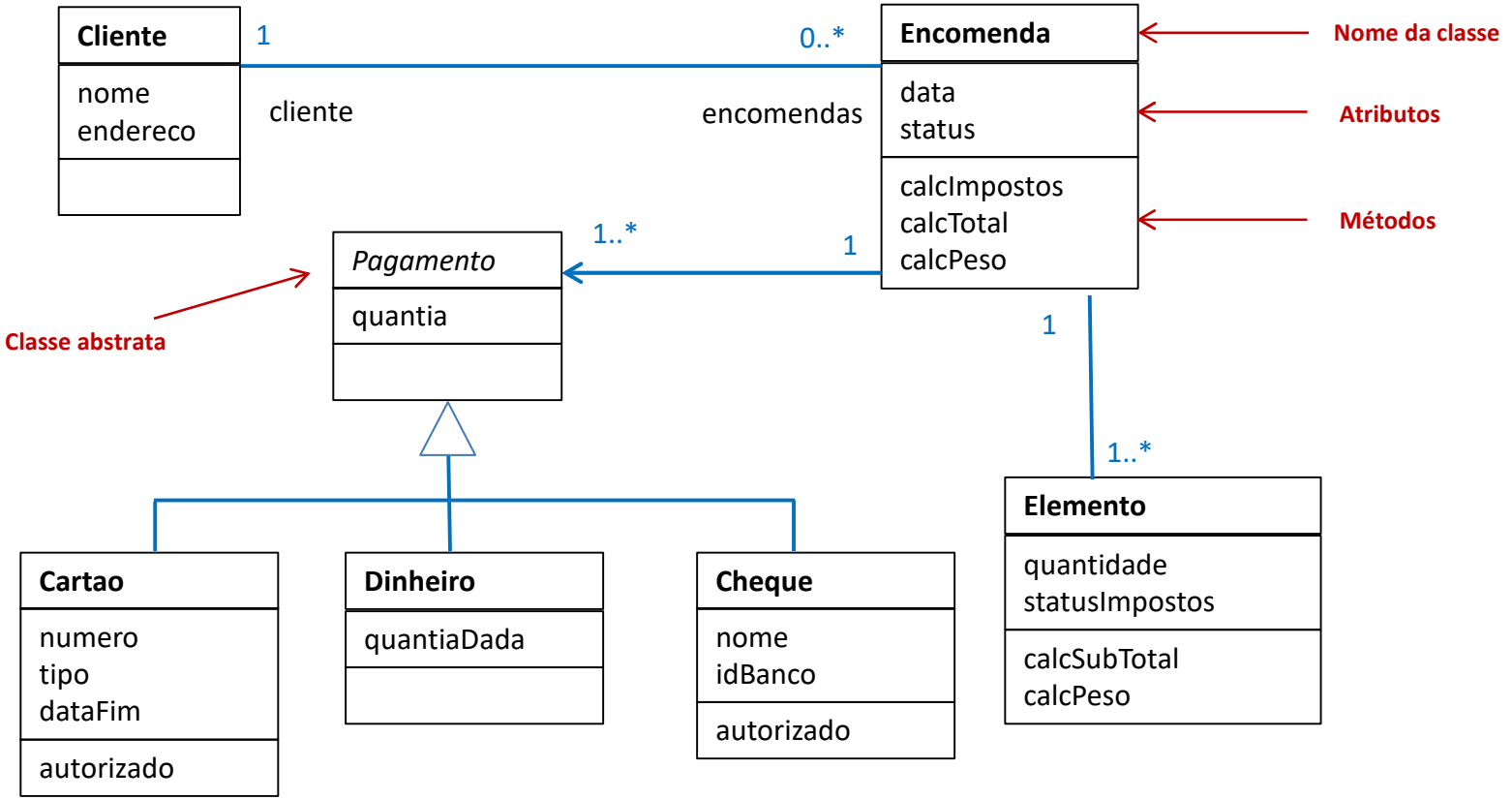
UML – Diagramas de classes

- Um diagrama de classes dá uma panorâmica geral do sistema, mostrando as suas classes e as relações entre elas
- Os diagramas de classes são estáticos, ou seja mostram o que interage, mas não descrevem a interação

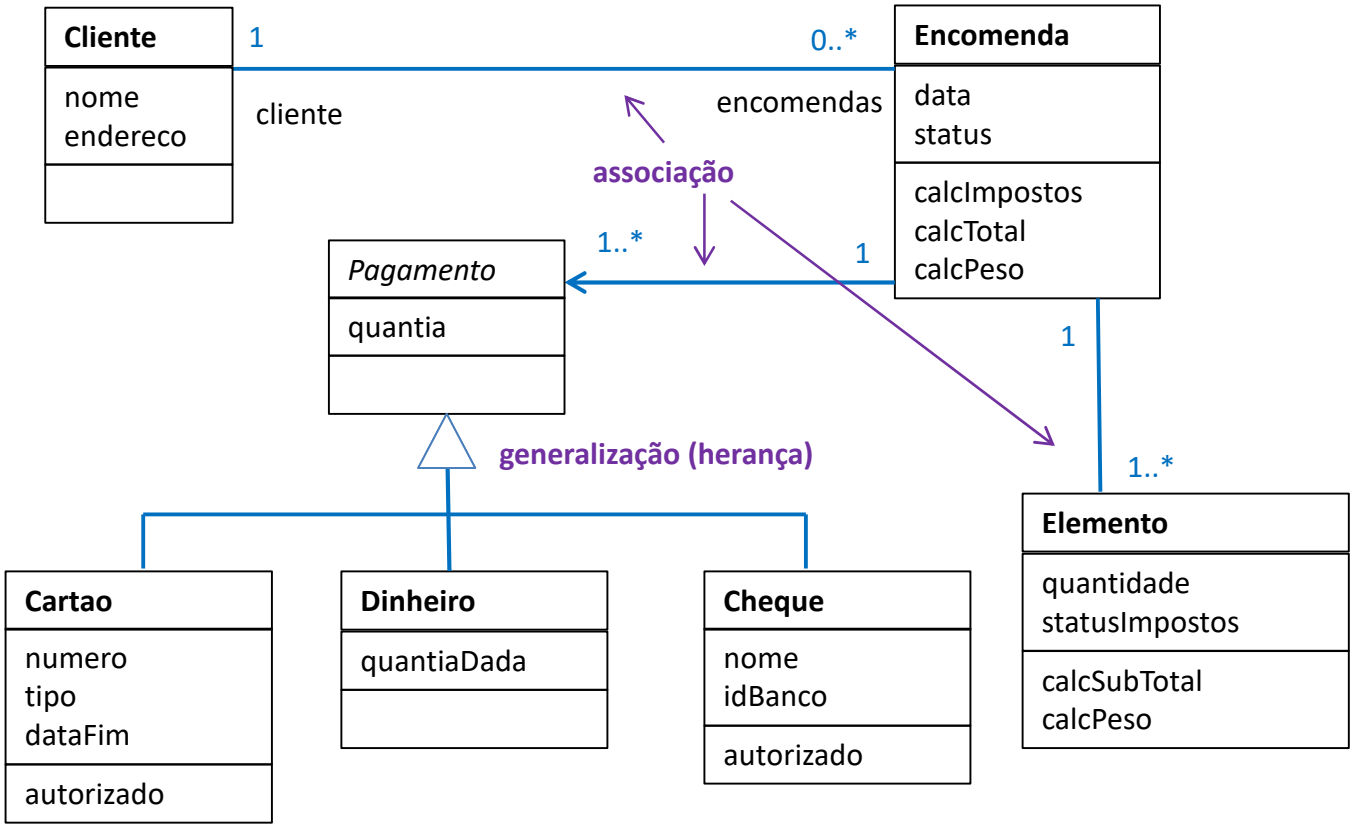
UML – Diagrama de classes simplificado



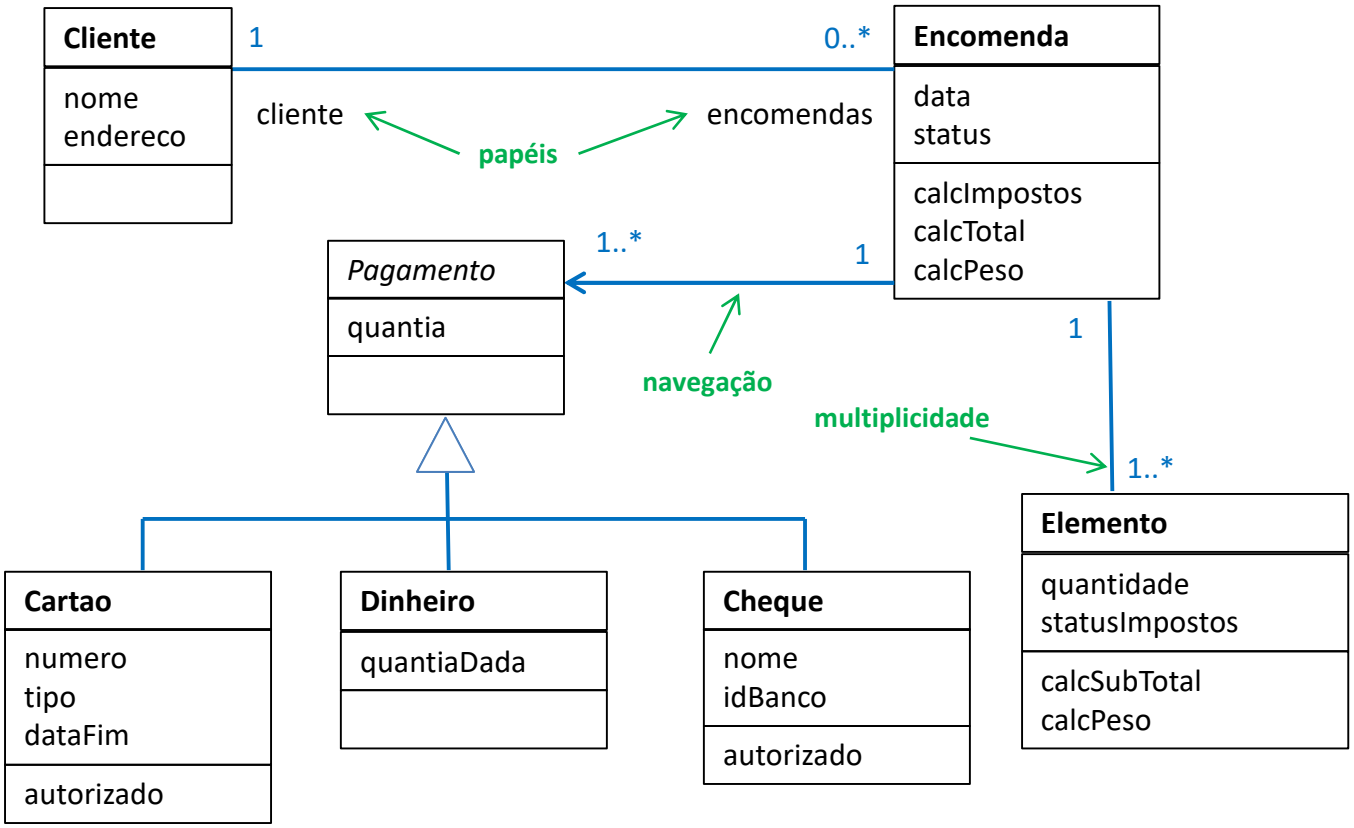
UML – Diagrama de classes simplificado



UML – Diagrama de classes simplificado



UML – Diagrama de classes simplificado



UML – Diagramas de classes

- Uma classe é representada por um retângulo dividido em 3 partes:
  - nome da classe
  - atributos
  - comportamentos/métodos
- No caso de classes abstratas o *nome* aparece em itálico
- Atributos ou métodos estáticos aparecem sublinhados
- O nome da interface é precedido da indicação de «interface»
- A implementação de uma interface por uma classe é representada como a relação de herança, em que o tipo de linha é tracejado

UML – Diagramas de classes

- Para cada atributo deve ser indicado
  - Nível de proteção
  - Nome
  - Tipo

  Exemplo: - nomePessoa: String

- Para cada método deve ser indicado
  - Nível de proteção
  - Nome
  - Tipo dos parâmetros
  - Tipo devolvido

  Exemplo: + aplicaImposto(float, int): float

- Os níveis de proteção são representados da seguinte forma
  - Private: “-”
  - Package (omissão): “~”
  - Protected: “#”
  - Public: “+”

UML – Diagramas de classes

- As relações entre classes são representadas por linhas:
  - **Associação** – há uma associação entre 2 classes se uma instância de uma classe precisa de conhecer a outra para desempenhar o seu papel. Num diagrama, uma associação é uma ligação entre duas classes
  - **Generalização** – uma associação em que há uma relação de herança, indicando que uma das classes é superclasse da outra

UML – Diagramas de classes

- A cardinalidade de um extremo de uma associação é o número de possíveis instâncias da classe nesse extremo associadas com uma única instância da classe no outro extremo
  - A cardinalidade é indicada por números ou por intervalos.

UML – Diagramas de classes

- Algumas cardinalidades comuns

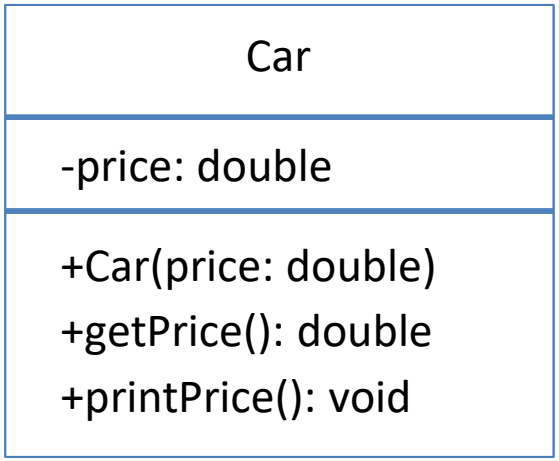
Cardinalidade	Significado
0..1	0 ou 1 instância (n..m significa n a m instâncias)
0..* ou *	Qualquer número de instâncias (mesmo zero)
1	Exactamente uma instância
1..*	Pelo menos uma instância

UML – Diagramas de classes

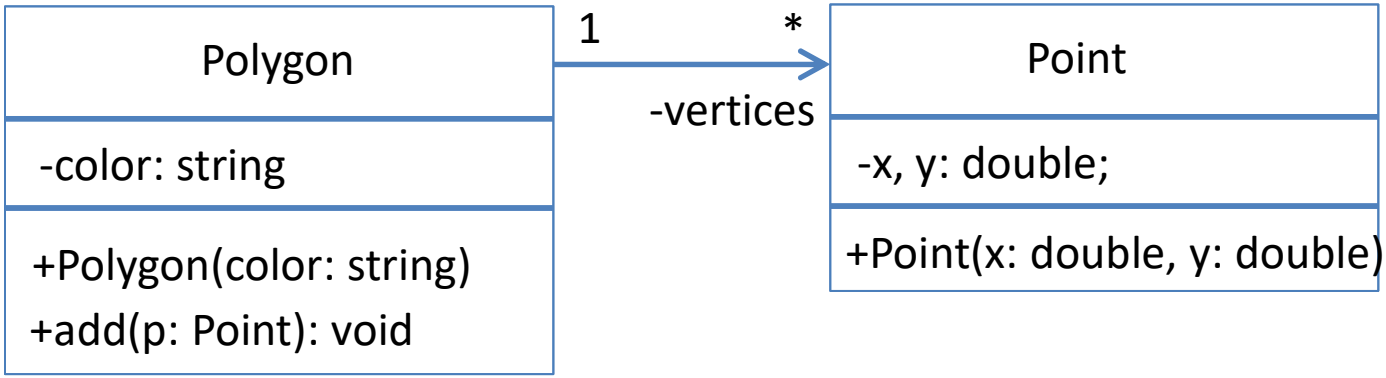
- Os diagramas de classe incluem classes, associações e cardinalidade
- Navegabilidade e papéis são opcionais para aumentar a clareza do diagrama

UML- Implementação

```
class Car {
    private double price;
    public Car(double price){
        this.price = price;
    }
    public double getPrice(){
        return price;
    }
    public void printPrice() {
        System.out.printf("Car price: %f", price);
    }
}
```



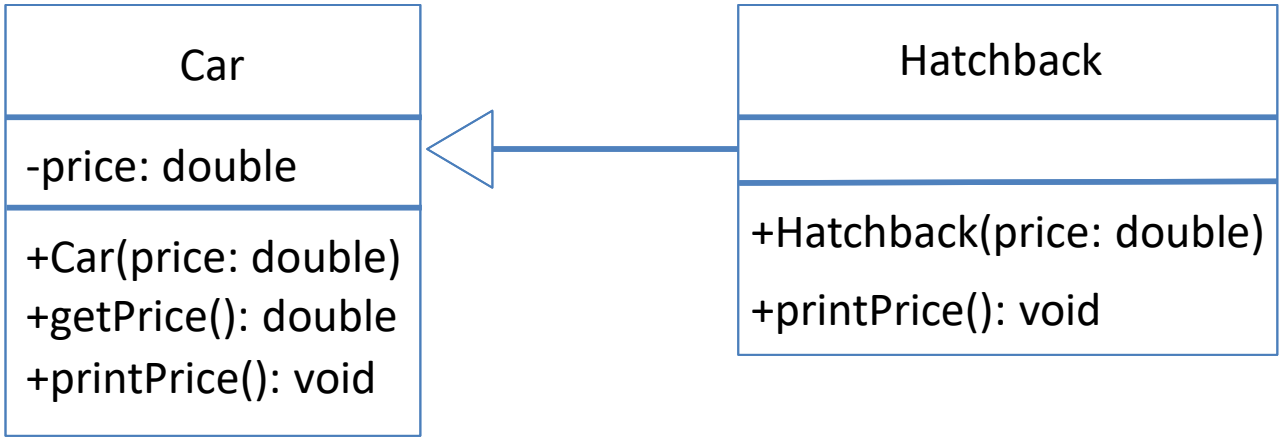
UML- Implementação



UML- Implementação

```
class Polygon {
    private String color;
    private ArrayList<Point> vertices;
    public Polygon(String color) {
        vertices = new ArrayList<>();
        this.color = color;
    }
    public void add(Point p) {
        vertices.add(p);
    }
}
class Point {
    private double x, y;
    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
}
```

UML- Implementação




```
class Hatchback extends Car {
    public Hatchback(double price) {
        super(price);
    }
    public void printPrice() {
        System.out.printf("Hatchback price: %f", getPrice());
    }
}
```

UML

- O UML inclui outros tipos de diagramas:
  - Use case diagrams
  - Class diagrams
  - Object diagrams
  - Sequence diagrams
  - Collaboration diagrams
  - Statechart diagrams
  - Activity diagrams
  - Component diagrams
  - Deployment diagrams

Referências

- https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/index.html

- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/