PROGRAMAÇÃO ORIENTADA PELOS OBJECTOS

Aula 4

Dependências entre classes e interfaces

Facturas

Requisitos

- Escreva um programa que permita imprimir uma factura, com base em dados lidos de um ficheiro
  - Uma factura descreve os valores cobrados por determinados produtos, cada um dos quais com determinadas quantidades
  - Uma factura é composta por linhas
  - Cada linha tem informação sobre um produto, incluindo a sua descrição, preço unitário, quantidade e preço total
  - A factura apresenta um endereço no início e o total a pagar, no fim
- Neste exemplo, vamos ignorar impostos, números fiscais e outras complicações

Exemplo de ficheiro de dados


```
Pastelaria Doces com Bué Açúcar
3
Pastel de Nata
1,2 3
Croissant com chocolate
1,5 4
Sorbet de limão em prosecco
3,2 2
```

Exemplo de factura


```
Pastelaria Doces com Bué Açúcar
descrição; preço unitário; quantidade; total
Pastel de Nata; 1,20; 3; 3,60
Croissant com chocolate; 1,50; 4; 6,00
Sorbet de limão em prosecco; 3,20; 2; 6,40
16.0
```

Interface de topo

- Ler dados de consumo
- Imprimir factura

Entidades

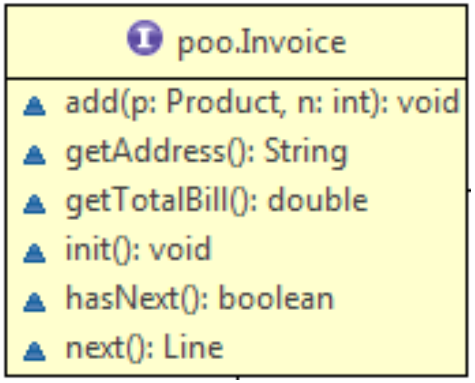
- Factura
  - Escrever factura
    - Devolver endereço
    - Escrever linhas detalhadas
    - Devolver total a pagar
- Linha de factura
  - Devolver produto, quantidade e total
- Produto
  - Devolver descrição do produto e preço unitário

Como especificar estas entidades?

- Directamente no código fonte?
  - Especificação das interfaces e das classes que as implementam
  - Comentários no código
    - Usando o Javadoc, quando programamos em Java
- Através de uma notação diagramática
  - Uma das mais populares é o UML, um standard usado na indústria para o desenho de software orientado a objectos
    - Nesta cadeira, usaremos diagramas de classes, que são apenas um dos mais de 10 tipos de diagramas do UML

Invoice (Interface em Java e em UML)

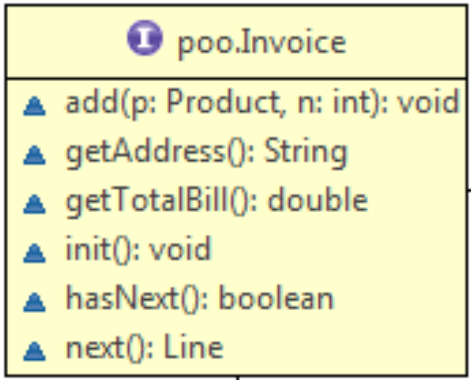

```
public interface Invoice {
  /**
   * Adiciona mais uma linha à factura, com <code>n</code>
   * produtos do tipo <code>p</code>.
   * @param p - Produto adicionado nesta linha
   * @param n - Quantidade de items adicionados nesta linha.
   */
   void add(Product p, int n);
  /**
   * Devolve o endereço do emissor da factura
   * @return - o endereço do emissor da factura
   */
   String getAddress();
   /**
    * Calcula o total da factura
    * @return - o total da factura
    */
   double getTotalBill();
   // Continua...
```



Invoice (Interface em Java e em UML)

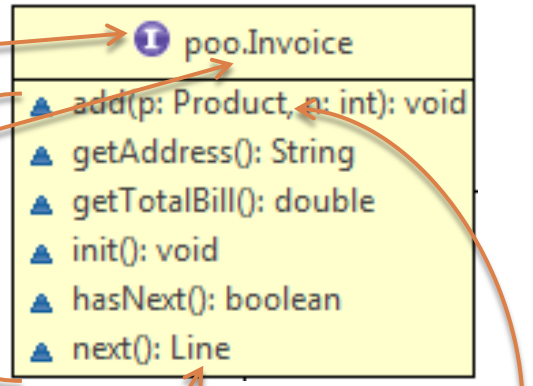

```
public interface Invoice {
    // ... continuação
    /**
     * inicia o iterador de linhas da factura
     */
    void init();

    /**
     * Verifica se existe mais alguma linha na factura
     * @return - true, se existe, false, caso contrário
     */
    boolean hasNext();
    /**
     * Devolve a próxima linha na factura
     * @return - a próxima linha da factura
     */
    Line next();
}
```



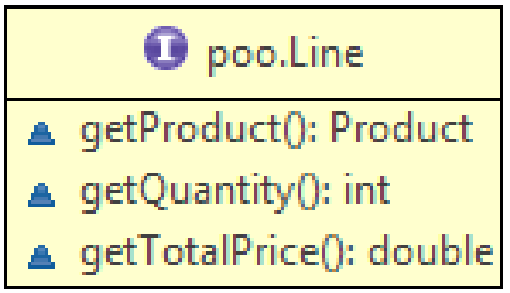
Especificação diagramática de uma interface

- Icon simbolizando a interface
- Nome da interface
- Operações definidas na interface
- Lista de argumentos
- Tipo de retorno das operações



Line (Interface em Java e em UML)

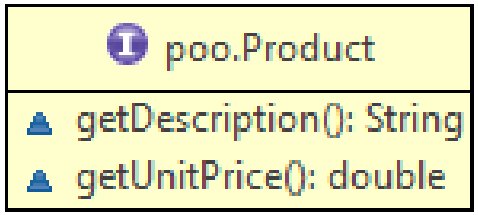

```
public interface Line {
   /**
    * Devolve o produto
    * @return - o produto
    */
   Product getProduct();
   /**
    * Devolve a quantidade de produtos
    * @return - número de unidades nessa linha
    */
   int getQuantity();
   /**
    * Devolve o preço total dos produtos nesta linha
    * @return - o preço total dos produtos
    */
   double getTotalPrice();
}
```



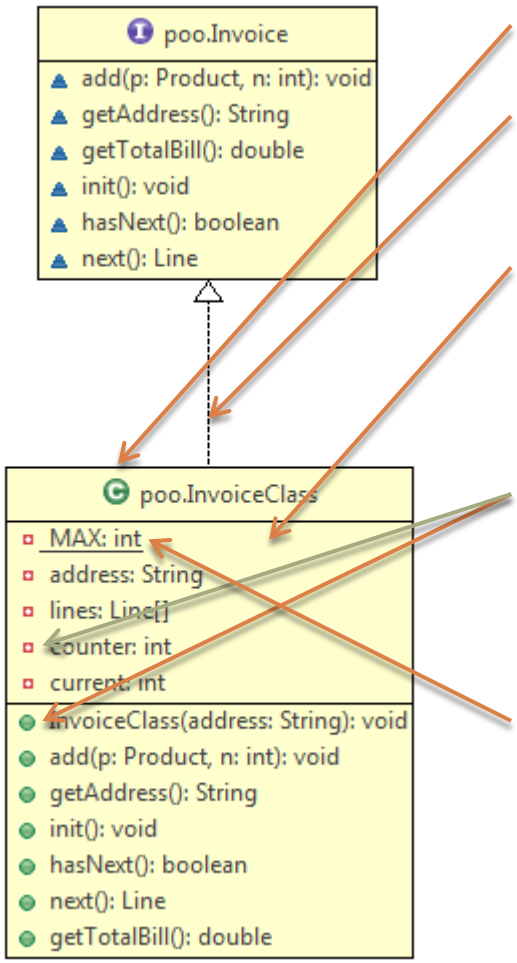
Product (Interface em Java e em UML)


```
public interface Product {
   /**
    * Devolve a descrição do produto.
    * @return - a descrição do produto.
    */
   String getDescription();

   /**
    * Devolve o preço unitário do produto.
    * @return - Preço unitário do produto.
    */
   double getUnitPrice();
}
```



Classes que implementam interfaces

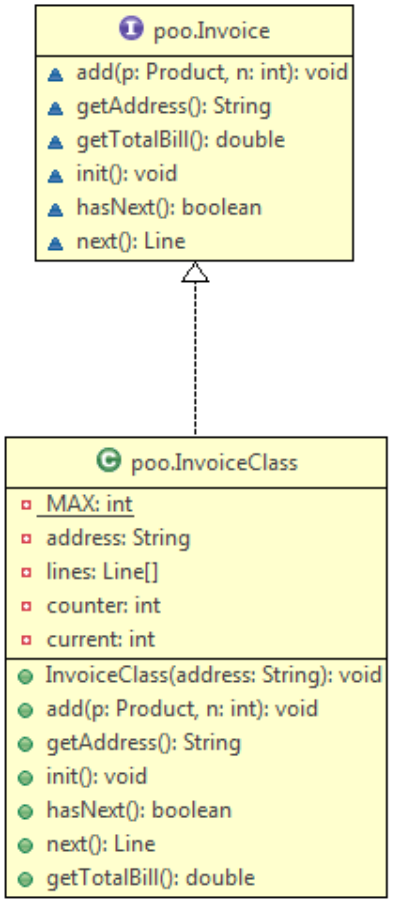


- Classe decorada com um icon diferente
- Relação implements representada com uma seta fechada e linha tracejada
- Repare no compartimento extra, para a declaração de variáveis e constantes na classe
- Icons diferenciados para mostrar visibilidade pública e privada, na classe
- Sublinhado indica que este membro é de classe (ou seja, declarado como static) e não de instância

Classes que implementam interfaces

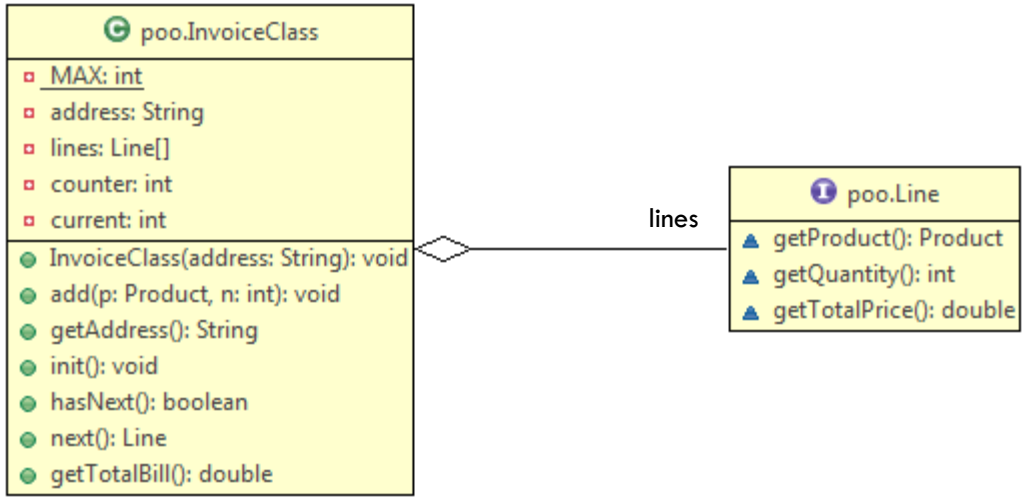

```
public class InvoiceClass implements Invoice {
  private static final int MAX = 10;
  private String address;
  private Line[] lines;
  private int counter;
  private int current;

  public InvoiceClass(String address) {...}
  public void add(Product p, int n) {...}
  public String getAddress() {...}
  public double getTotalBill() {...}
  public void init() {...}
  public boolean hasNext() {...}
  public Line next() {...}
}
```



Agregação

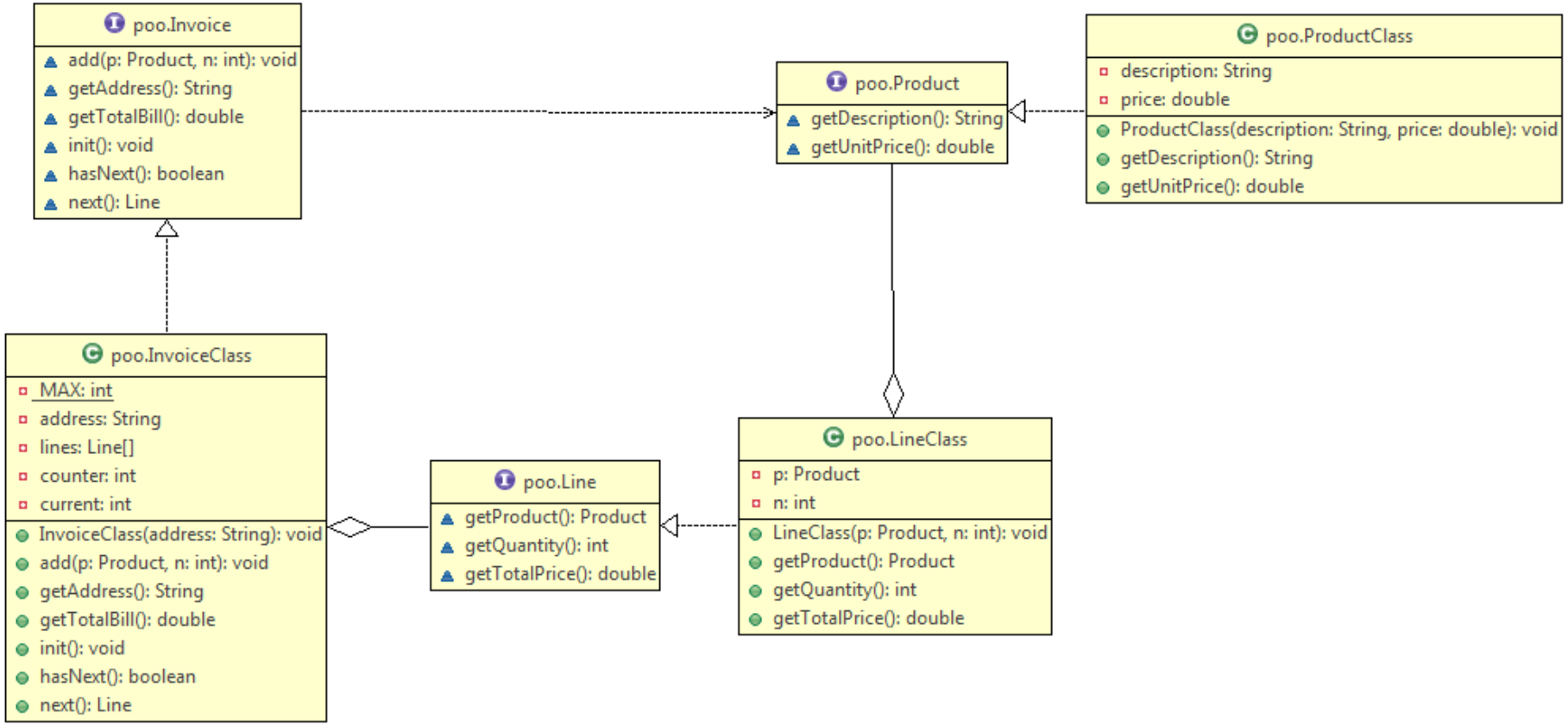
- Usa-se quando um elemento contém uma colecção de outros elementos de um determinado tipo
- Losango do lado do elemento "contentor"
- Nome da variável, do lado do elemento "contido"
- É redundante mostrar simultaneamente a variável e a respectiva agregação
  - Pode-se omitir a variável lines de InvoiceClass, desde que ela fique representada junto a Line
  - Pode-se omitir a representação da agregação no diagrama (desaconselhado)



Dependência

- Usado quando uma interface, ou classe, depende de outra para o seu comportamento
  - Neste caso, Invoice usa o tipo Product, na operação add

Diagrama de classes e interfaces



Demo do plug-in para o Eclipse Amateras UML

Este plug-in, instalado nos laboratórios, permite gerar os diagramas a partir do código e vice-versa, dentro do próprio projecto do Eclipse.