PROGRAMAÇÃO ORIENTADA A OBJETOS [POO]

Material 10 – POO_10

Prof. Mestre Marcos Roberto de Moraes [Maromo]

Interface e polimorfismo sem hierarquia de classes

Agenda

- Interfaces
- Implementação
- Exercícios práticos

Introdução

- Uma interface é um conjunto nomeado de comportamentos (e/ou elementos de dados constantes) para o qual um **implementador deve fornecer código**.
- Uma interface **especifica o comportamento** que a implementação oferece, mas não como ela é realizada.

Definição

```
public interface  interfaceName {
        return Type methodName(argumentList);
}
```

Exemplo:

```
public interface Imprimivel {
        void imprimirDocumento(String doc);
}
```

Sobre a declaração de uma interface

- Uma declaração de interface se parece com uma declaração de classe, exceto que você usa a palavra-chave **interface**.
- Você pode atribuir à interface o nome que desejar (sujeito às regras da linguagem), no entanto, por convenção, **os nomes de interface se parecem com os nomes de classe.**
- Os **métodos definidos em uma interface não têm nenhum corpo** de método.
- O **implementador da interface** é responsável por **fornecer o corpo** do método (assim como com os métodos abstratos).

Ainda sobre a declaração

- Você define as hierarquias de interfaces do mesmo modo que para as classes, exceto que uma única classe pode implementar tantas interfaces quanto for necessário. Se uma classe estender outra classe e implementar interfaces, estas serão listadas depois da classe estendida, como segue:

```
public class Secretaria extends Empregado  implements Imprimivel {
// Etc...
}
```

Interfaces de Marcador

- Afinal, uma interface não precisa ter nenhum corpo. Na verdade, a definição a seguir é perfeitamente aceitável:

```
public interface Imprimivel {
}
```

- Falando em termos gerais, essas interfaces são chamadas de interfaces de marcador, porque elas marcam uma classe como implementação da interface , mas não oferecem nenhum comportamento explícito.

Implementando interfaces

- Para usar uma interface, basta implementá-la, o que significa simplesmente fornecer um corpo de método, o qual, por sua vez, fornecerá o comportamento que cumpre o contrato da interface. Você faz isso com a palavra-chave implements:

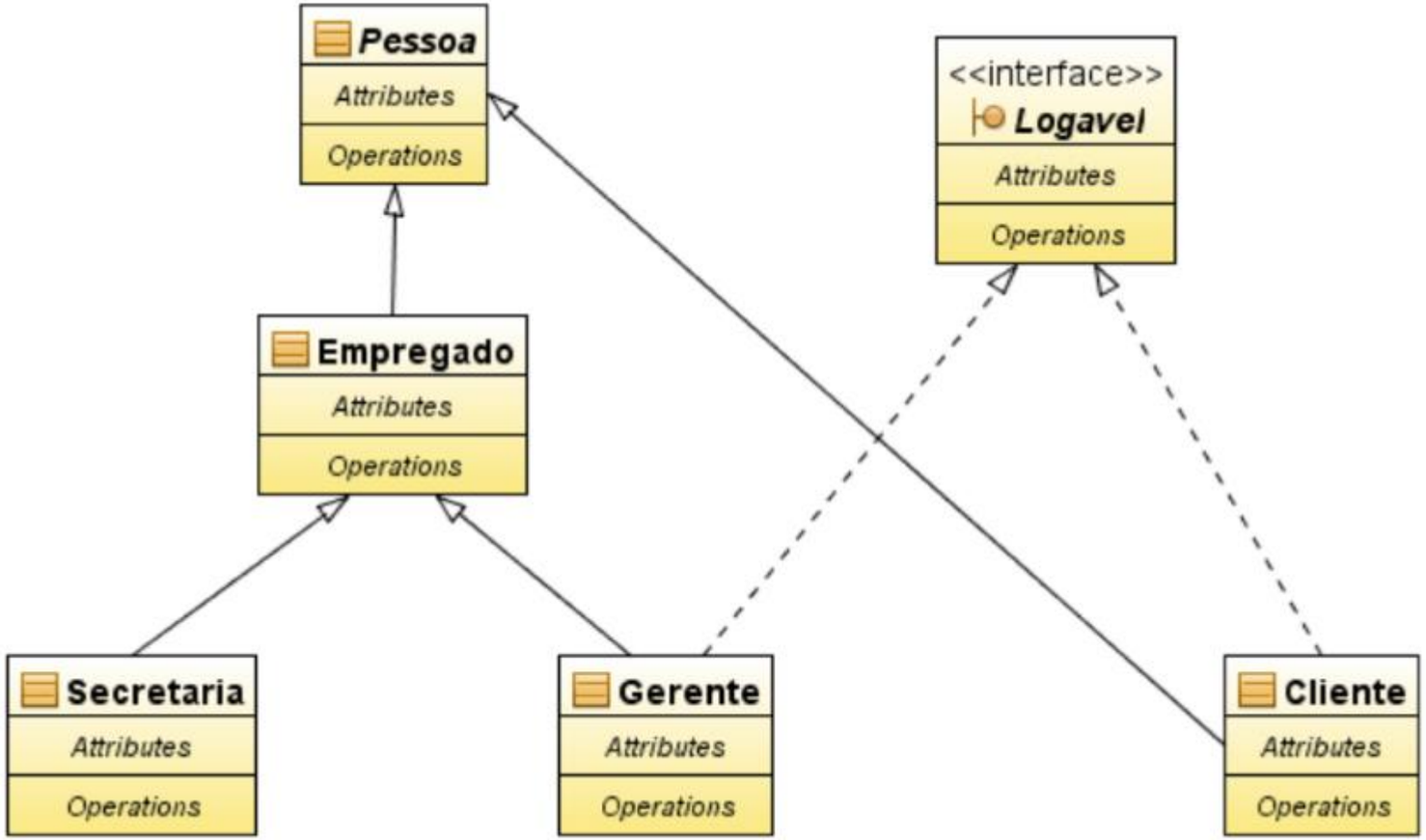
```
public class Secretaria extends Empregado  implements Imprimivel {
            public void imprimirDocumento(String doc){
        System.out.println(doc.ToString());
    }
}
```

- Ao implementar a interface, você fornece o comportamento dos métodos na interface. É necessário implementar os métodos com assinatura que correspondam aos presentes na interface, com a adição do modificador de acesso public.

Quando usar

- Em geral, uma interface é utilizada quando classes díspares(isto é, não relacionadas) precisam compartilhar métodos e constantes comuns. Isso permite que objetos de classes não relacionadas sejam processadas polimorficamente – objetos de classes que implementam a mesma interface podem responder às mesmas chamadas de métodos. Os programadores podem criar uma interface que descreve a funcionalidade desejada e então implementar em quaisquer classes que requerem essa funcionalidade. (DEITEL, 2005, 354)

Exemplo: Diagrama de Classes



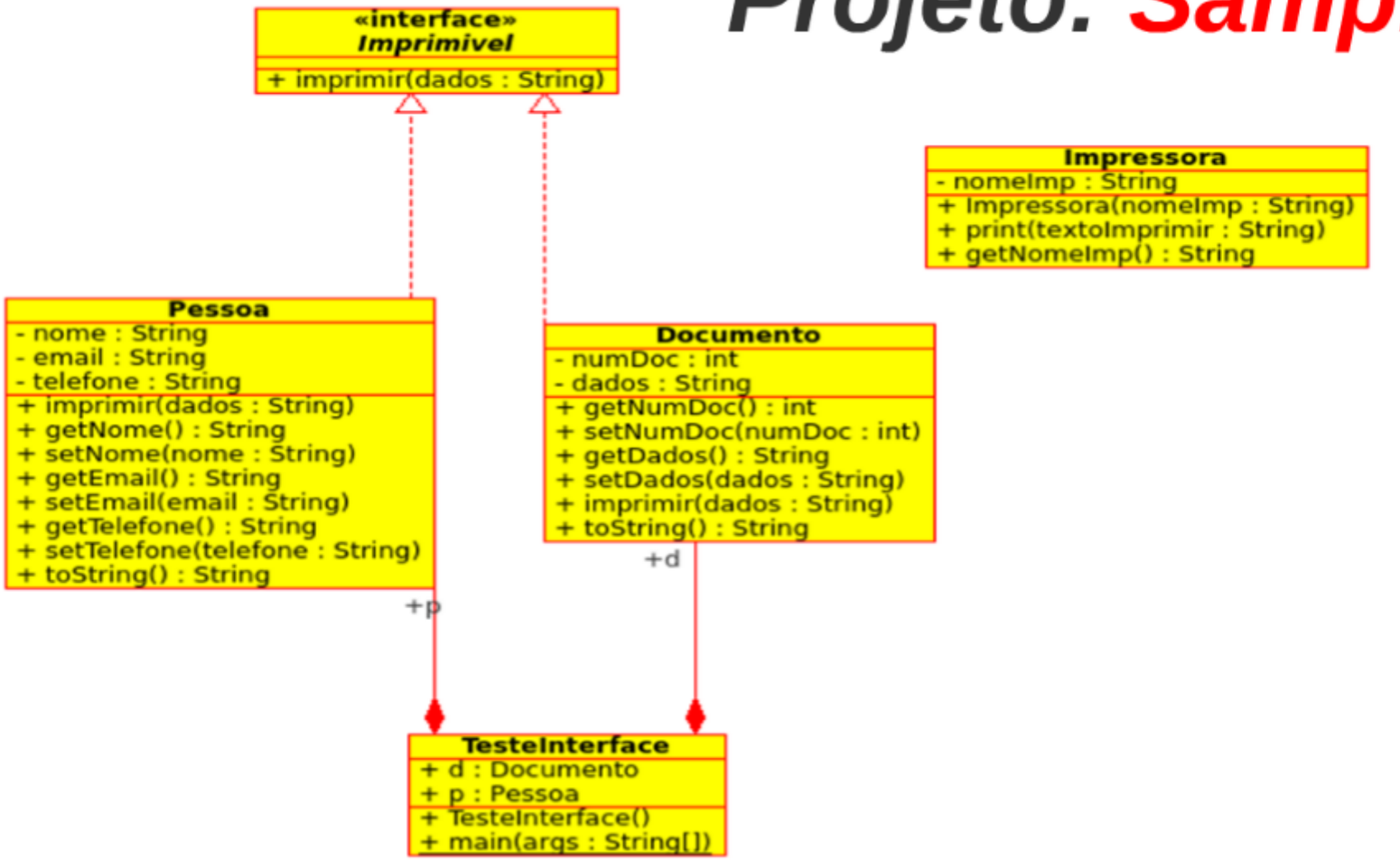
Baseado em CAELUM - Java Objetos Interfaces - pg. 128

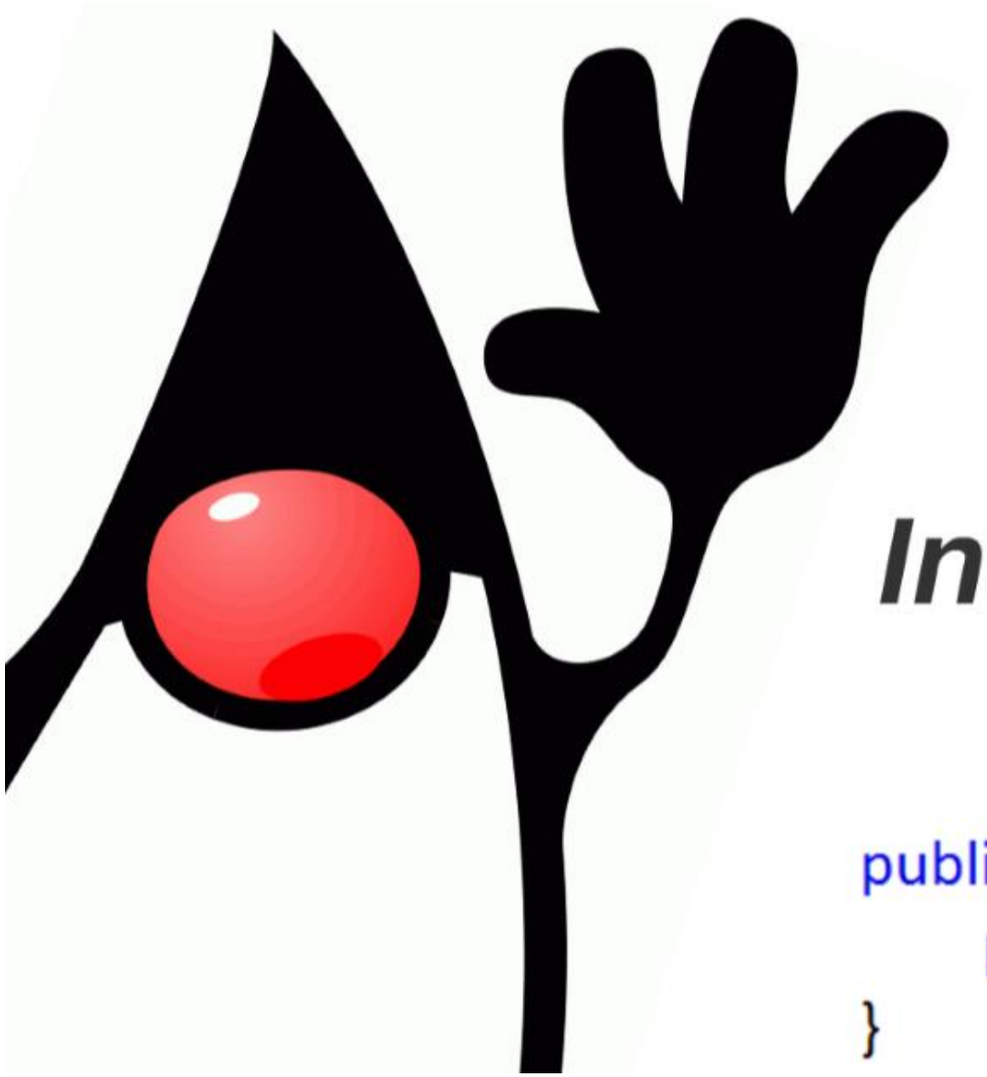
Ainda sobre a definição do conceito

- Pode-se dizer que uma interface costuma ser utilizada no lugar de uma classe abstract quando não há nenhuma implementação padrão a herdar – isto é, nenhum campo e nenhuma implementação padrão de método, como ocorre com classes public abstract.
- Deitel apresenta como um protocolo de comportamento, ela é uma lista de métodos abstratos, inclusive variáveis.

Exemplo Prático

Projeto: SamplePrint





Interface: Imprimivel

```
public interface Imprimivel {
        public void imprimir(String dados);
}
```

Classe: Pessoa

```
package sampleprint;
public class Pessoa implements Imprimivel {
    private String nome;
    private String email;
    private String telefone;
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("Nome:  ").append(getNome()).append("\n");
        sb.append("Email: ").append(getEmail()).append("\n");
        sb.append("Telefone: ").append(getTelefone()).append("\n");
        return sb.toString();
    }
    public String getNome() {...}
    public void setNome(String nome) {...}
    public String getEmail() {...}
    public void setEmail(String email) {...}
    public String getTelefone() {...}
    public void setTelefone(String telefone) {...}
    @Override
    public void imprimir(String dados) {
        Impressora imp = new Impressora("Lexmark da diretoria");
        System.out.println("\nImprimindo na Impressora: " + imp.getNomeImp());
        imp.print(dados);
    }
```

Classe: Documento

```
package sampleprint;
public class Documento implements Imprimivel {
    private int numDoc;
    private String dados;
    public String getDados() {...}
    public void setDados(String dados) {...}
    public int getNumDoc() {...}
    public void setNumDoc(int numDoc) {...}
    @Override
    public void imprimir(String dados) {
        Impressora imp = new Impressora("HP da Secretaria");
        System.out.println("Imprimindo na impressora: " + imp.getNomeImp());
        imp.print(dados);
    }
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("Número do documento: ").append(getNumDoc()).append("\n");
        sb.append("Dados a serem impressos: \n");
        sb.append(getDados());
        sb.append("\n");
        return sb.toString();
    }
}
```

Classe: Impressora

```
1   package sampleprint;
2
3   public class Impressora {
4       private String nomeImp;
5
6       public Impressora(String nomeImp) {
7           this.nomeImp = nomeImp;
8       }
9
10      public String getNomeImp() {
11          return nomeImp;
12      }
13      public void print(String textoImprimir){
14          System.out.println(textoImprimir);
15      }
16
17  }
```

Classe: TesteInterface

```
1   package sampleprint;
2
3   public class TesteInterface {
4       Documento d;
5       Pessoa p;
6       public TesteInterface() {
7           d = new Documento();
8           p = new Pessoa();
9           d.setNumDoc(1);
10          d.setDados("Carta para empresa X Ltda. xxx xxx xxxx");
11          d.imprimir(d.toString());
12          p.setNome("Carlos Santos");
13          p.setEmail("c@carlos.com");
14          p.setTelefone("19-9999-9999");
15          p.imprimir(p.toString());
16      }
17      public static void main(String[] args) {
18          // TODO code application logic here
19          TesteInterface t = new TesteInterface();
20      }
21  }
```

Nota sobre o exemplo anterior

- No exemplo foi possível identificar uma interface **Imprimivel** que é implementada por duas classes: classe **Pessoa** e classe **Documento**.
- Essas duas classes concretas foram obrigadas a implementar o método public void imprimir () definido na interface. Se isso não acontecesse o javac, o compilador do java, não permitiria a compilação das classes.
- Nota-se ainda que as Classes: Pessoa e Documento não possuem nenhuma relação de hierarquia, são díspares; com o recurso de Interface podemos implementar o método imprimir para elas.

Referências

- Foram fontes de consulta os seguintes materiais:
  - Desenvolvimento IBM
  - Deitel & Deitel – Java Como programar
  - Caelum – Apostila Java FJ11

FIM DO MATERIAL

Obrigado,
Maromo