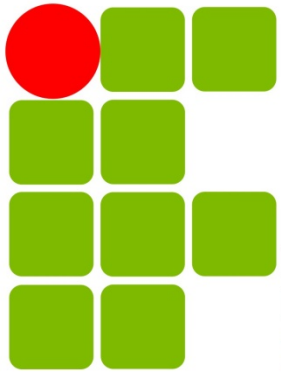


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARAÍBA

Interfaces
Programação Orientado a Objetos

Prof. Katyusco F. Santos

Campina Grande/PB – 2017

O que é uma Interface?

- Define um "contrato" que deve ser cumprido por uma classe que o "assine";
- Descreve as especificações e funcionalidades comuns entre vários objetos.

Em Java, uma interface é um arquivo (*Java*) que possui apenas assinaturas de métodos (*abstratos*) e constantes.

Exemplo: Classe que implementa duas interfaces

```
interface Leitor {
   String lendo();
}

interface Programador {
   void pensando(char[] ideias);
   String digitando();
}
```

```
class ParticipanteForum implements Leitor, Programador {
   String pensamento;
   public String lendo() { // método definido na interface Leitor
      return "Forum";
   }
   public void  pensando(char[] ideias) { // método definido na interface Programador
      pensamento = new String(ideias);
   }
   public String digitando() { // método definido na interface Programador
      return pensamento;
   }
   private String aprendendo() { // método exclusivo desta classe
      return "Java";
   }
}
```

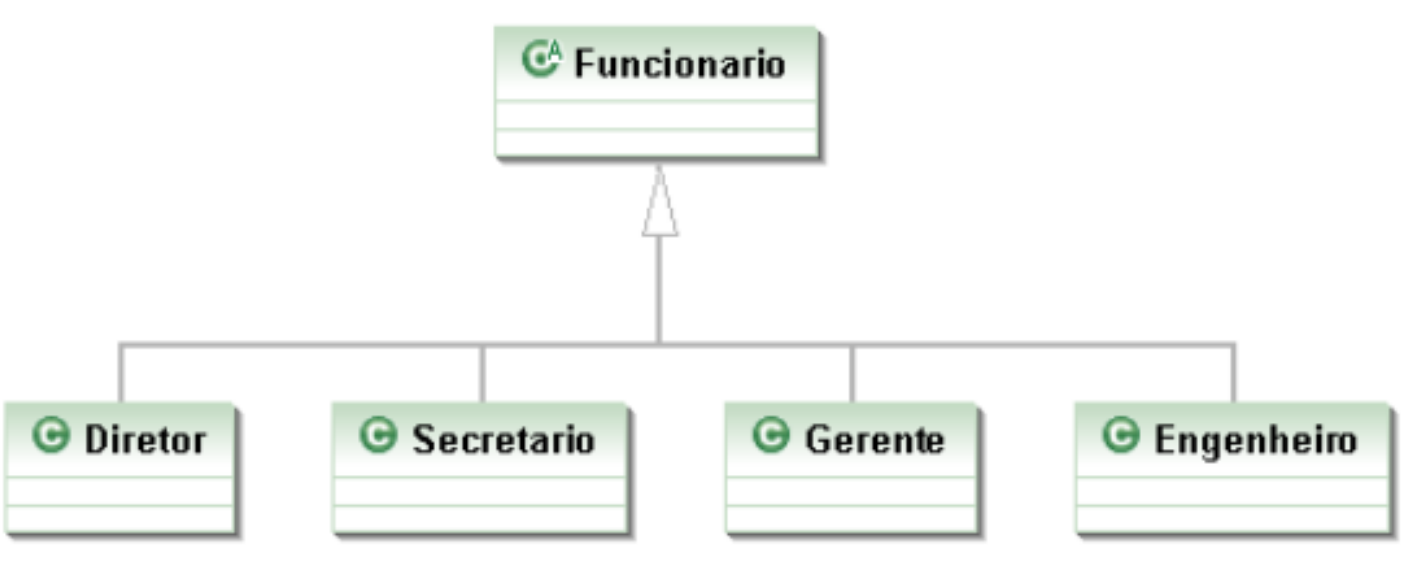
```
public class Demonstracao {
   public static void main(String[] args) {
      ParticipanteForum participante = new ParticipanteForum ();  // instanciado o objeto
      Leitor leitor = participante;  // upcast para Leitor
      System.out.println("O participante está lendo " + leitor.lendo());
      Programador programador = participante; // upcast para Programador
      String java = "Java";
      programador.pensando(java.toCharArray());
      System.out.println ("E programando " + programador.digitando());
   }
}
```

Criamos um objeto "participante" da classe ParticipanteForum, e em seguida referenciamos este objeto por uma variável do tipo Leitor e depois do tipo Programador.

O que é uma Interface?

- **Exemplo:**

Queremos criar um método para autenticação em um sistema de diretores e gerentes



O que é uma Interface?

- **Exemplo:**

```
class SistemaInterno {

  void login(Funcionario funcionario) {
    // invocar o método autentica?
    // não da! Nem todo Funcionario tem
  }
}
```

```
class Diretor extends Funcionario {

  public boolean autentica(int senha) {
    // verifica aqui se a senha confere com a recebida como parametro
  }

}

class Gerente extends Funcionario {

  public boolean autentica(int senha) {
    // verifica aqui se a senha confere com a recebida como parametro
    // no caso do gerente verifica também se o departamento dele
    // tem acesso
  }

}
```

O que é uma Interface?

- **Exemplo:**

```
class SistemaInterno {

  void login(Funcionario funcionario) {
    // invocar o método autentica?
    // não da! Nem todo Funcionario tem
  }
}
```

Problema:

Apenas as subclasses `Diretor` e `Gerente` possuem o método `autentica`!

```
class Diretor extends Funcionario {

  public boolean autentica(int senha) {
    // verifica aqui se a senha confere com a recebida como parametro
  }

}

class Gerente extends Funcionario {

  public boolean autentica(int senha) {
    // verifica aqui se a senha confere com a recebida como parametro
    // no caso do gerente verifica também se o departamento dele
    // tem acesso
  }

}
```

O que é uma Interface?

- **Exemplo:**

```
class SistemaInterno {

  // design problemático
  void login(Diretor funcionario) {
    funcionario.autentica(...);
  }

  // design problemático
  void login(Gerente funcionario) {
    funcionario.autentica(...);
  }

}
```

```
class Diretor extends Funcionario {

  public boolean autentica(int senha) {
    // verifica aqui se a senha confere com a recebida como parametro
  }

}

class Gerente extends Funcionario {

  public boolean autentica(int senha) {
    // verifica aqui se a senha confere com a recebida como parametro
    // no caso do gerente verifica também se o departamento dele
    // tem acesso
  }

}
```

O que é uma Interface?

- **Exemplo:**

```
class SistemaInterno {

  // design problemático
  void login(Diretor funcionario) {
    funcionario.autentica(...);
  }
```

Também não funciona!

Seria necessário adicionar novos métodos `login` ao criar novos funcionários autenticáveis

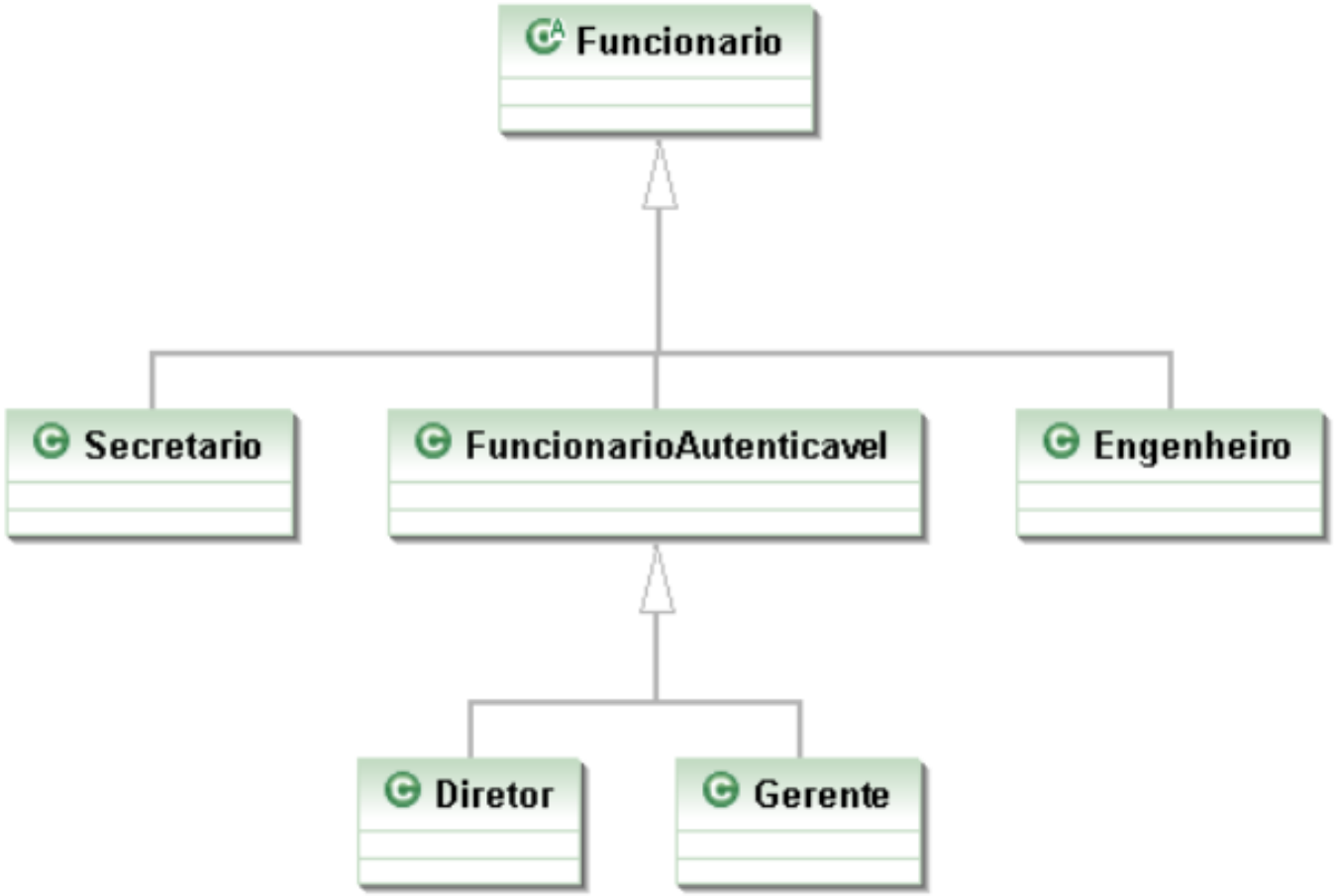
```
class Diretor extends Funcionario {

  public boolean autentica(int senha) {
    // verifica aqui se a senha confere com a recebida como parametro
  }

}

class Gerente extends Funcionario {

  public boolean autentica(int senha) {
    // verifica aqui se a senha confere com a recebida como parametro
    // no caso do gerente verifica também se o departamento dele
    // tem acesso
  }

}
```

O que é uma Interface?

- **Exemplo:**

Podemos usar herança, então!

Criando uma classe abstrata `FuncionarioAutenticavel`

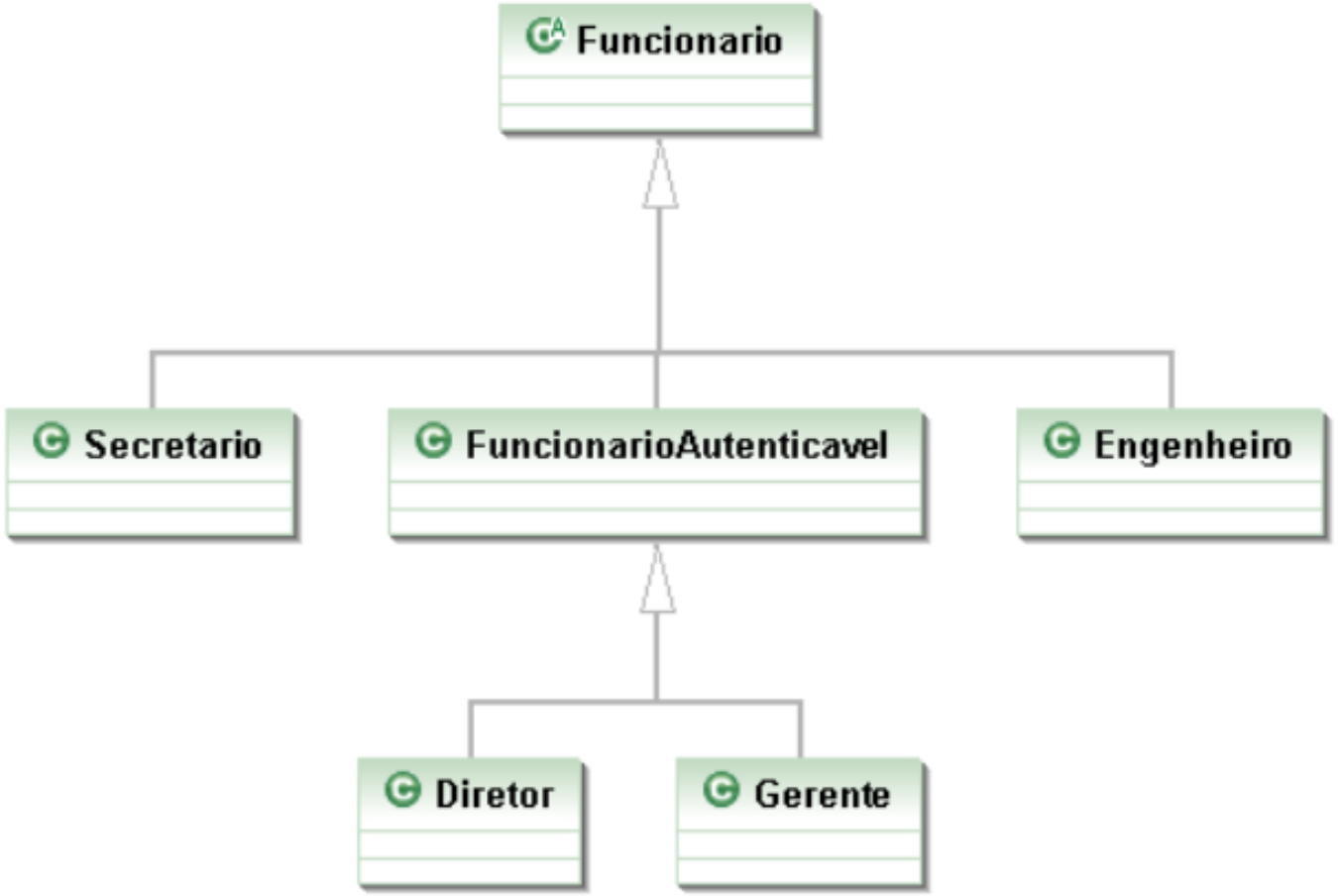


O que é uma Interface?

- **Exemplo:**

Problema:

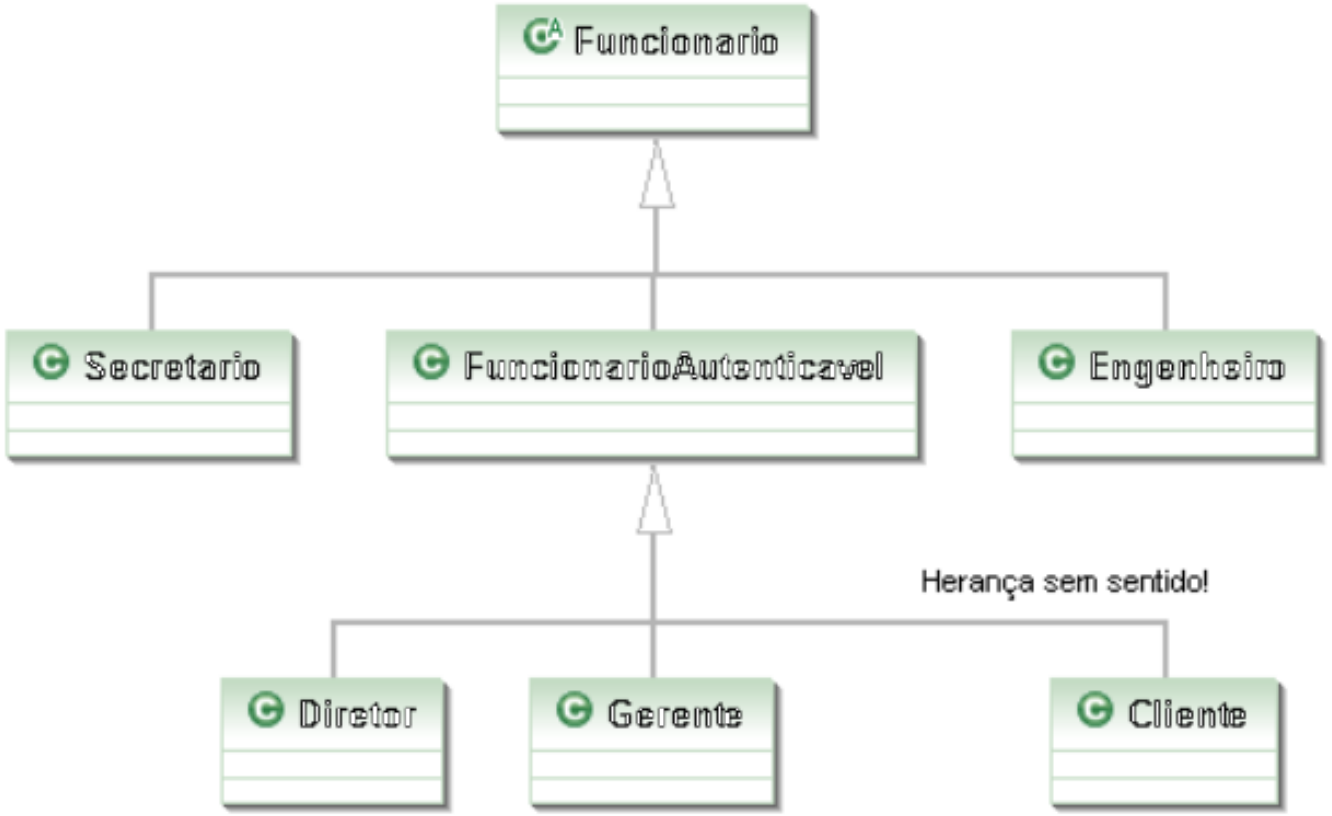
Queremos, agora, que os clientes também tenham acesso ao sistema interno. O que fazer?



O que é uma Interface?

- **Exemplo:**

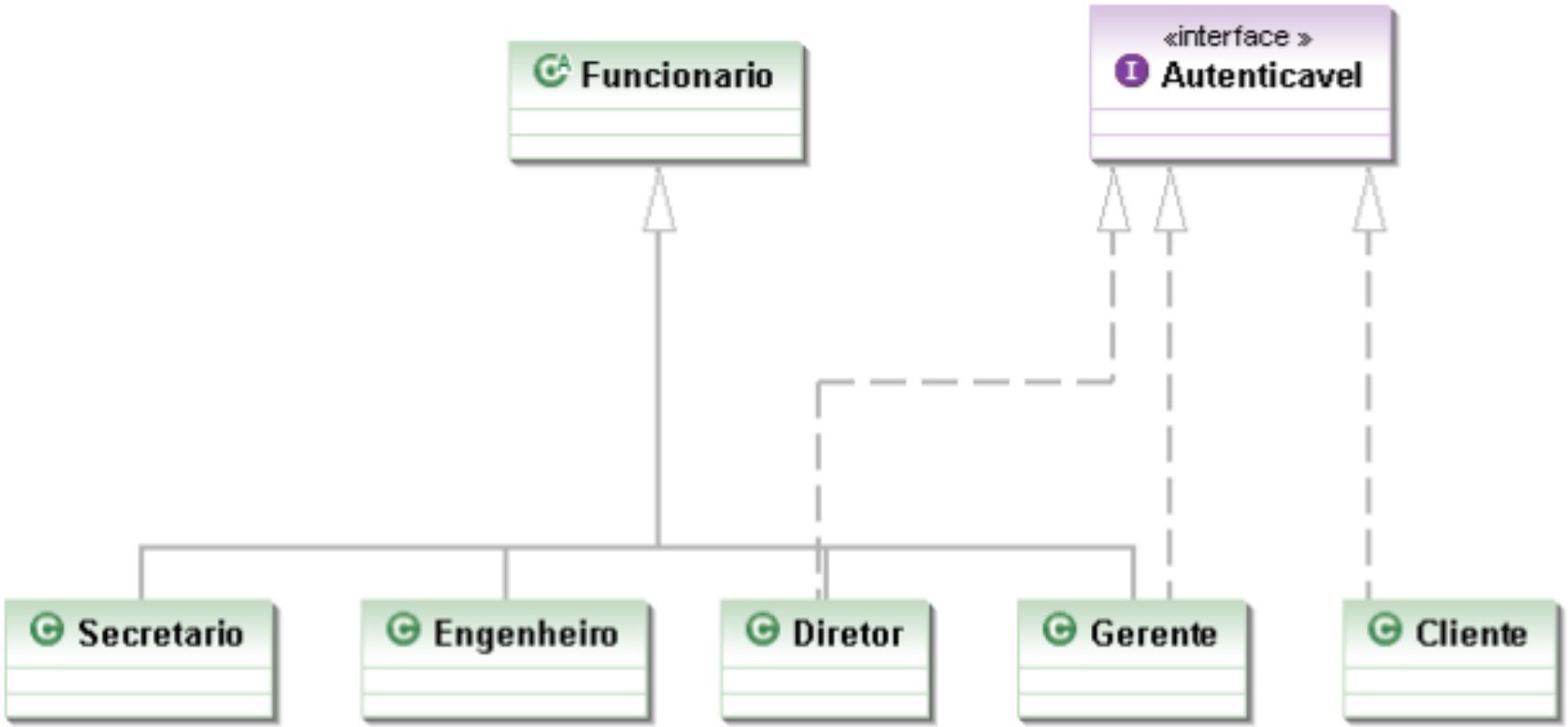
Uma solução (não aconselhável) é criar uma herança entre `FuncionarioAutenticavel` e `Cliente`



O que é uma Interface?

- **Exemplo:**

Em casos como este, o uso de interfaces é o mais recomendável!



O que é uma Interface?

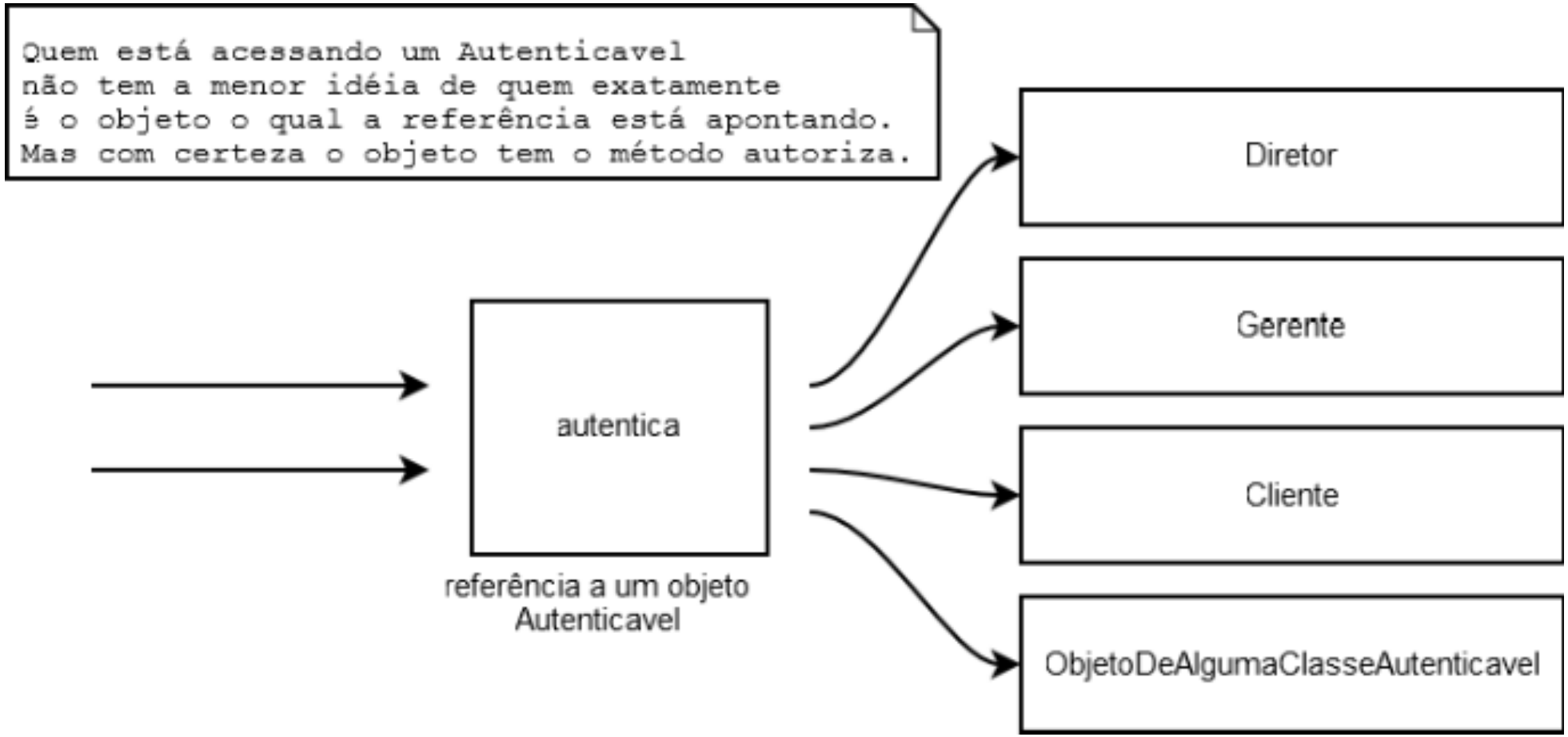
- **Exemplo:**

Variáveis do tipo `Autenticavel` se referem a qualquer objeto de uma classe que implemente `Autenticavel`

```
class SistemaInterno {

  void login(Autenticavel a) {
     int senha = // pega senha de um lugar, ou de um scanner de
polegar
     boolean ok =  a.autentica(senha);

     // aqui eu posso chamar o autentica!
     // não necessariamente é um Funcionario!
     // Mais ainda, eu não sei que objeto a
   // referência "a" está apontando exatamente! Flexibilidade.
  }

}
```

O que é uma Interface?

- **Exemplo:**



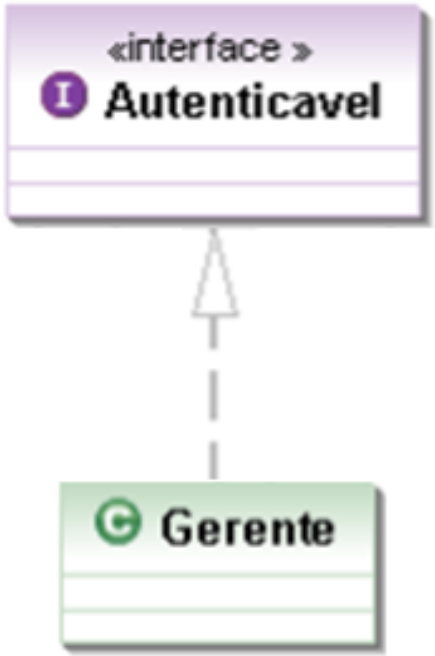
Características de Interfaces em Java

- Todos os métodos são implicitamente públicos (`public`) e abstratos (`abstract`);
- Atributos são implicitamente públicos (`public`), estáticos (`static`) e constantes (`final`);
- Interfaces não podem ser instanciadas;
- Tornam o código mais flexível: é possível modificar trechos de código sem ocasionar maiores problemas!

Características de Interfaces em Java

- Utiliza-se a palavra-chave `interface` para definir uma interface em Java; e a palavra-chave `implements` para classes que “contratam” uma interface.

No exemplo anterior, ao implementar a classe `Autenticavel`, entende-se que “a classe `Gerente` se compromete a ser tratada como `Autenticavel`, sendo **obrigada** a ter os métodos necessários, definidos neste contrato"



Diferenças entre Interfaces e Classes Abstratas

- Uma classe abstrata que possui apenas métodos abstratos pode ser definida como uma interface;
- Em Java, uma classe pode estender apenas uma única classe (herança simples); utilizando interfaces, uma classe pode implementar várias interfaces ao mesmo tempo.

Por falar em interface... uma dica!

Programe sempre em função de interfaces, não de implementação!

Dicas de Leitura & Referências Utilizadas

- Apostila Java e Orientação a Objetos – Capítulo 10 Interfaces
- Orientação a Objetos – Interfaces e Polimorfismo
- Utilizando Interfaces em Java (Linha de Código)