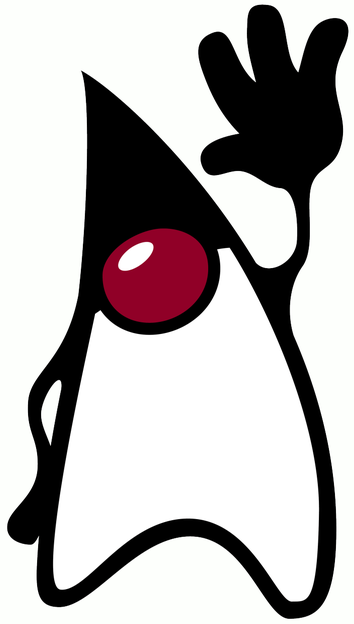
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Interfaces

QXD0007 - Programação Orientada a Objetos



Prof. Bruno Góis Mateus (brunomateus@ufc.br)

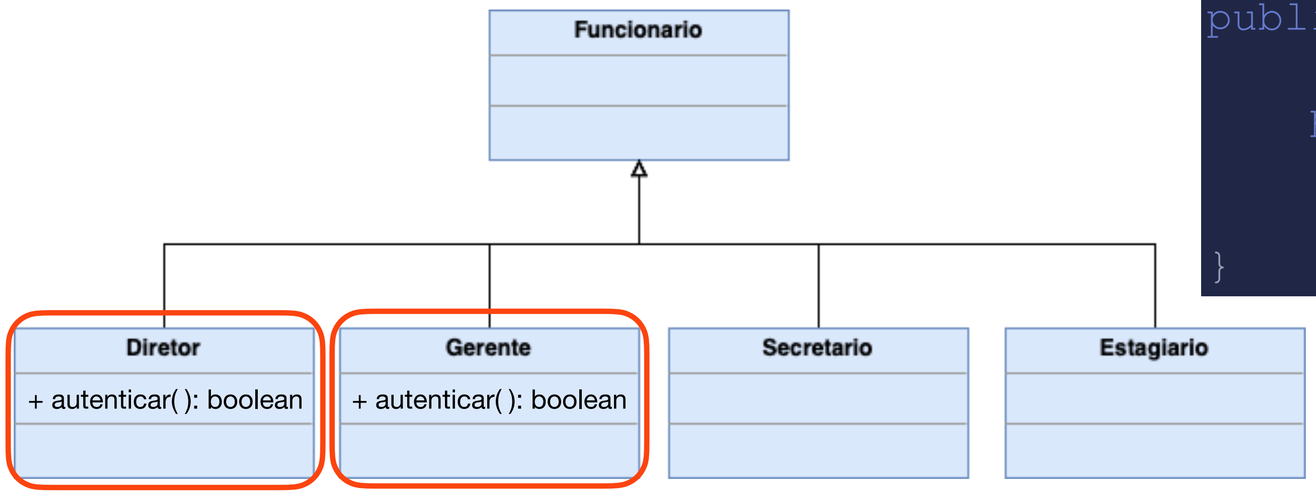
Conteúdo

- Interfaces
- Mão na massa

Interfaces

Interfaces

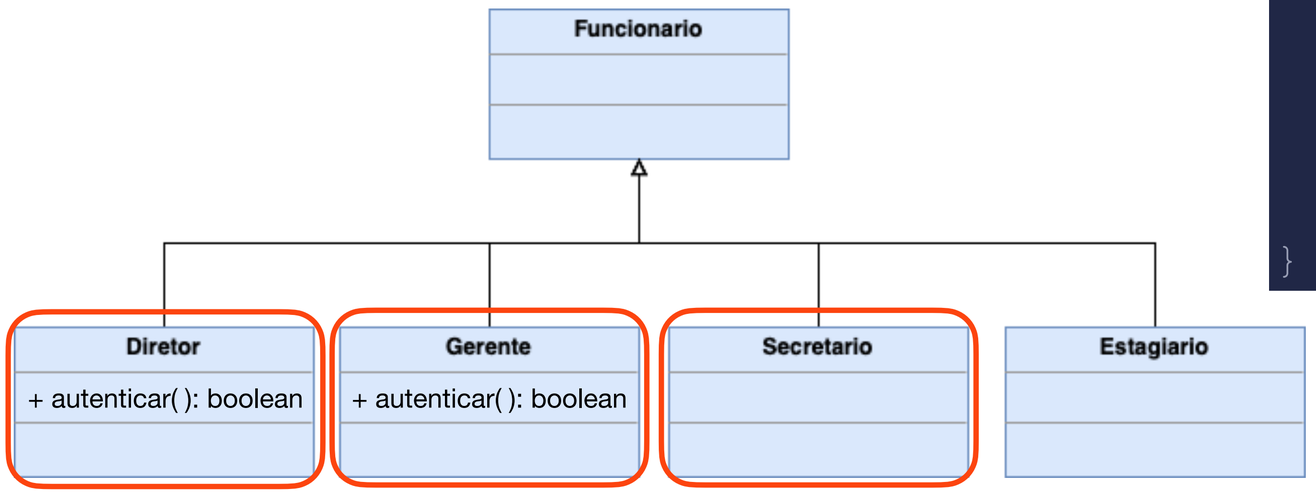
- Imagine um sistema de controle do banco que pode ser acessado por Gerentes e Diretores:



?

```
public class SistemaInterno {

    public void login(Funcionario funcionario){
        funcionario.autenticar()
    }
}
```

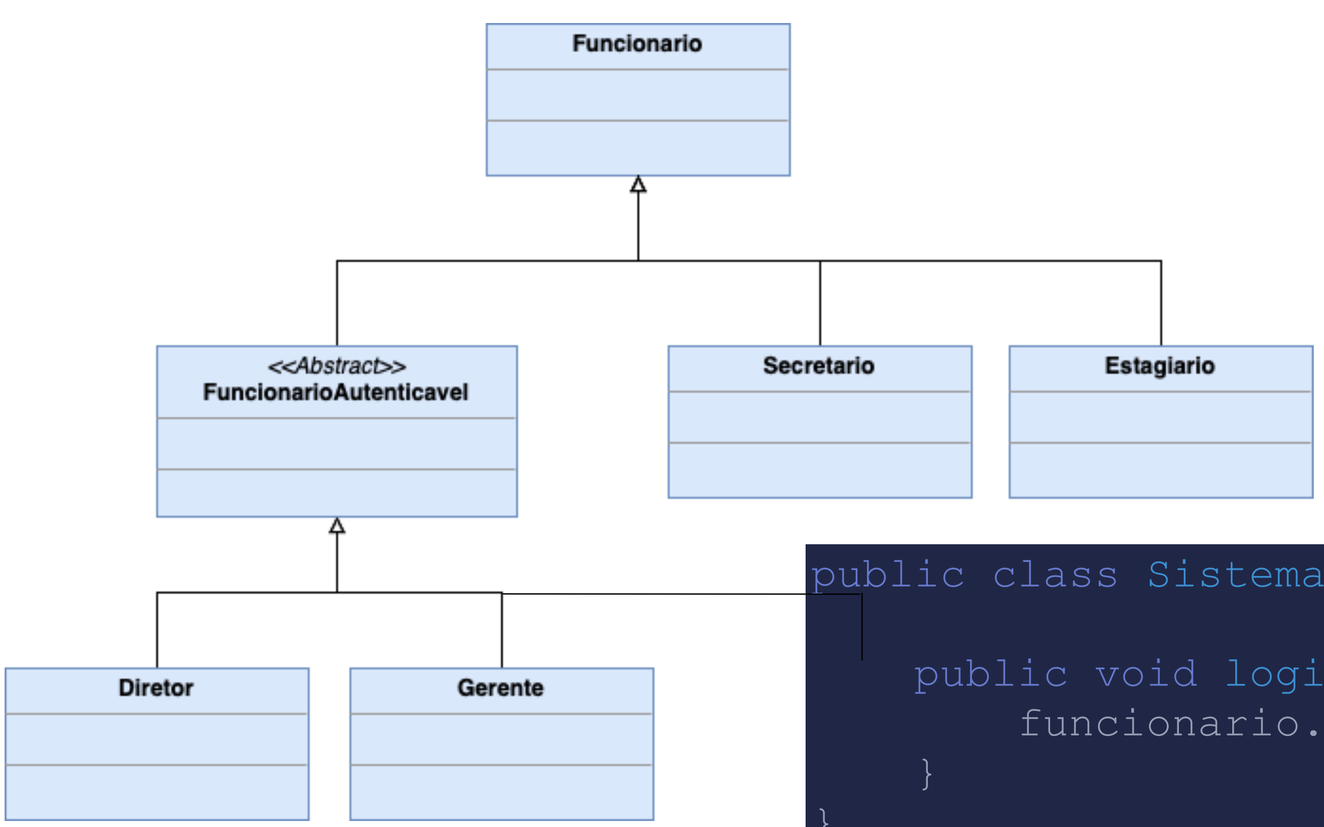
Interfaces

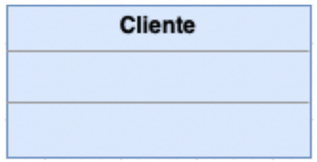



```
public class SistemaInterno {

    public void login(Diretor funcionario) {
        funcionario.autenticar();
    }

    public void login(Gerente funcionario) {
        funcionario.autentica();
    }
}
```

?

Interfaces





?

```
public class SistemaInterno {

    public void login(FuncionarioAutenticavel funcionario) {
        funcionario.autenticar();
    }
}
```

Interfaces

- Uma interface descreve um comportamento comum entre diferentes classes
- Expõem somente o que o objeto deve fazer:
  - Não determina como o objeto tem que fazer
  - Pode definir um série métodos que são abstratos e públicos por padrão
  - A classe que implementa a interface define o como
- Não têm construtores
- O nome de uma interface é freqüentemente um adjetivo. Ex: Sortable, Comparable

Interfaces

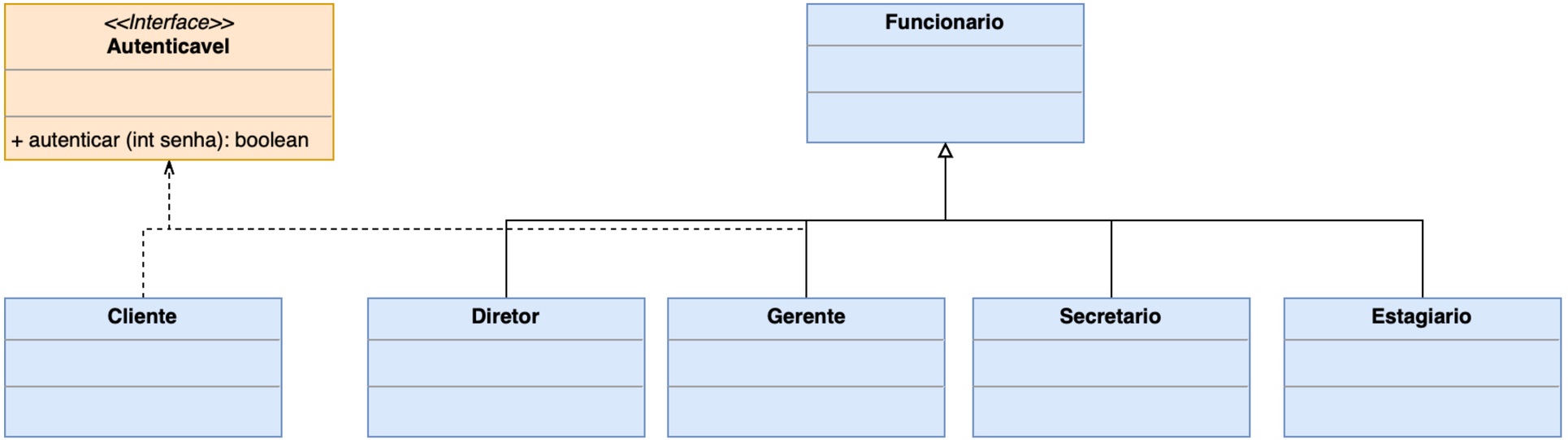
Exemplo

- Contrato:

```
interface Autenticavel {
    boolean autentica(int senha);
}
```

  - Quem quiser ser Autenticavel precisa:
    - Autenticar dada uma senha, devolvendo um booleano

Interfaces




```
public class SistemaInterno {

    public void login(Autenticavel funcionario) {
        …
        funcionario.autenticar(senha);
    }
}
```

Interfaces

Múltiplos supertipos

- Uma classe pode implementar mais de uma interface
- A classe que implementa uma interface deve definir (não necessariamente implementar!) os métodos da interface

```
public abstract class Shape implements Drawable, Nomeavel, AreaCalculavel{

  public Shape(String color, Point point) {

  }
}
}
  }
```

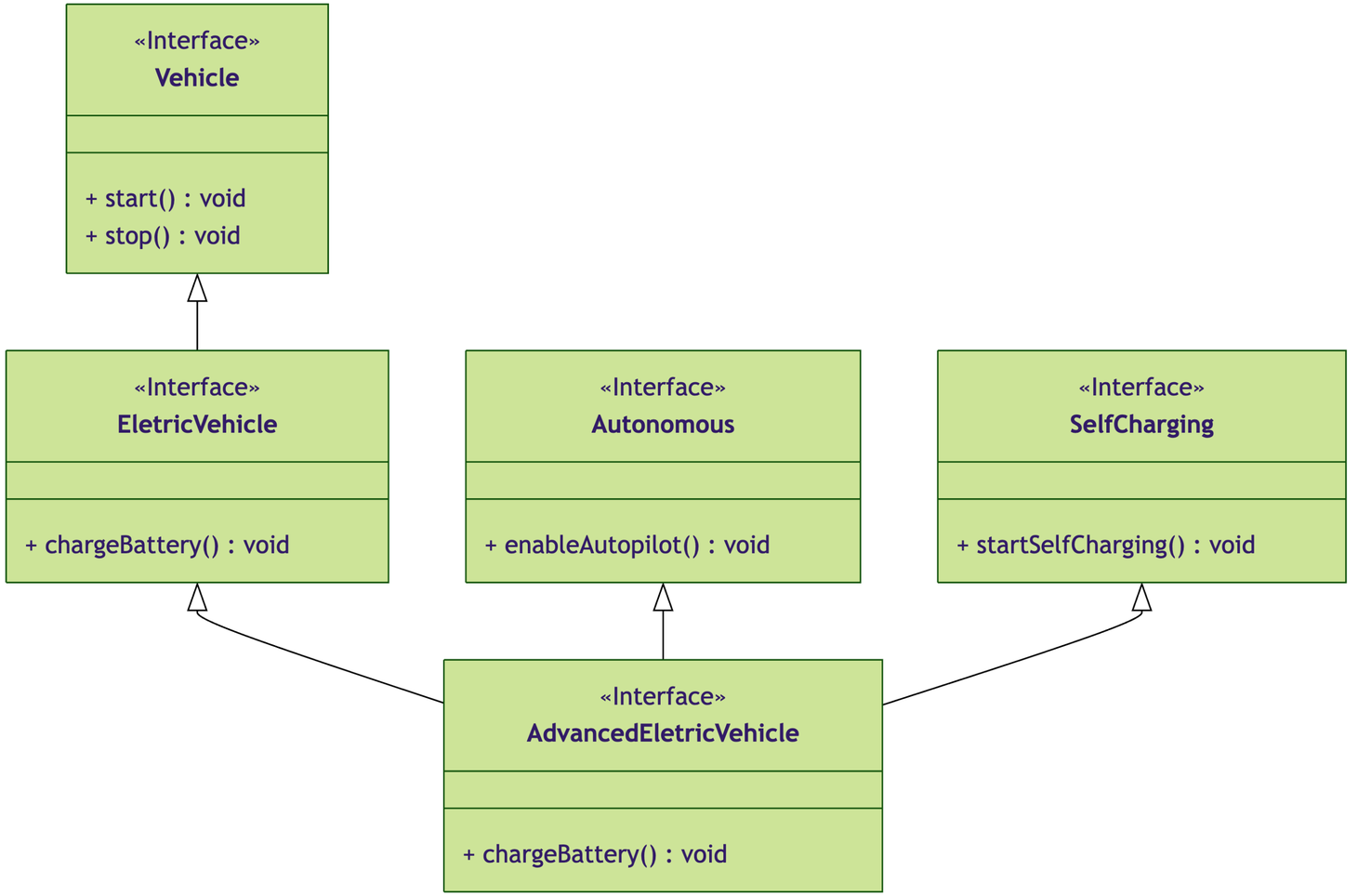
Interfaces

Herança Múltiplas

- Suponha que no sistema que você vai construir é necessário representar um veículo
- Suponha que existem um tipo de veículo que são os elétricos
- E se fosse necessário adicionar veículos autônomos?
- E se fosse necessário adicionar veículos autocarregavéis?
- E se fosse necessário adicionar veículos elétricos, autônomos, e autocarregavéis?

Interfaces

Herança Múltiplas



Interfaces

Atributos e métodos default

- Desde o Java 8, a interface pode ter métodos com implementação
  - Se forem método instâncias eles devem ser default
    - Podem ser herdados e sobrescritos
  - Podem ser métodos estáticos
  - Contudo, não podem possuir atributos de instância (públicos e estáticos por padrão)

Interfaces

Motivação para uso de default

- Com a introdução do Java Streams, o Java queria modificar a API de Collections
- No entanto, adicionar mais comportamentos a uma interface implicaria em erros nas implementações existes
- Para manter a compatibilidade com versões anteriores, o Java introduziu o conceito métodos default
    - É possível adicionar novos métodos – com comportamentos padrão – às interfaces, e suas implementações preexistentes continuam funcionando

Interfaces

Reusabilidade

- Evita duplicação de código através da definição de um tipo genérico, tendo como subtipos várias classes não relacionadas
- Pode agrupar objetos de várias classes definidas independentemente
  - Sem compartilhar código via herança (menor acoplamento)
  - Implementações totalmente diferentes

Interfaces

Interfaces Java comumente utilizadas

- Comumente implementadas:
  - java.lang.Comparable, java.lang.Runnable, java.io.Serializable
- Comumente utilizadas:
  - java.lang.Iterable (Collections)
  - java.util.Collection (Collections)
  - java.util.List (Collections)
  - java.util.Map (Collections)
  - java.util.Set (Collections)

Interfaces

Classes abstratas vs Interfaces

	Interfaces	Classes Abstrastas
Modificadores de acesso	public	Todos
Podem ter atributos?	Não	Sim
Podem ter atributos de classe?	Sim (public e final)	Sim
Podem conter métodos abstratos?	Sim	Sim
Podem conter métodos concretos	Não	Sim
Podem conter métodos default?	Sim	Não
Podem conter métodos estáticos?	Sim	Sim
Podem ser instanciadas?	Não	Não
Podem ter construtores?	Não	Sim
Herança	Pode herdar de várias interfaces	Pode herdar de uma única classe
Implementação (*implements*)	Várias interfaces	Não

Interfaces

Classes vs Interfaces

Classes

- O que uma classe faz
  - As assinaturas dos métodos
- Como as classes realizam essas tarefas
  - O corpos dos métodos e seus atributos

Interfaces

- Contrato entre a classe e o mundo externo
  - Determinam o que, porém não indica o como
- Ao implementar uma interface, a classe se compromete a fornecer o comportamento publicado pela interface

Interfaces

Classes vs Interfaces

Classes

- Agrupam objetos com implementações compartilhadas
- Definem novas classes através de herança e implementação
- Uma hierarquia de classes compartilha implementação

Interfaces

- Agrupa objetos com implementações diferentes
- Define novas interfaces por meio de herança (múltipla) de interfaces
- Uma hierarquia de interfaces compartilha obrigações
- Permitem que objetos tenham vários tipos

Interfaces

Mais informações

- Métodos default
  - https://www.amitph.com/introduction-to-default-methods-in-java-8/
  - https://docs.oracle.com/javase/tutorial/java/landl/defaultmethods.html
- Discussão sobre classes abstratas vs interfaces
  - https://stackoverflow.com/questions/8531292/why-to-use-interfaces-multiple-inheritance-vs-interfaces-benefits-of-interface
  - https://betterprogramming.pub/choosing-between-interface-and-abstract-class-7a078551b914

Interfaces

Mais informações

- Commonly used Java-able Interfaces
- Standard Interfaces

Mãos na massa

Por hoje é só