Programação Orientada a Objetos

Professor: Anderson Elias

Estácio

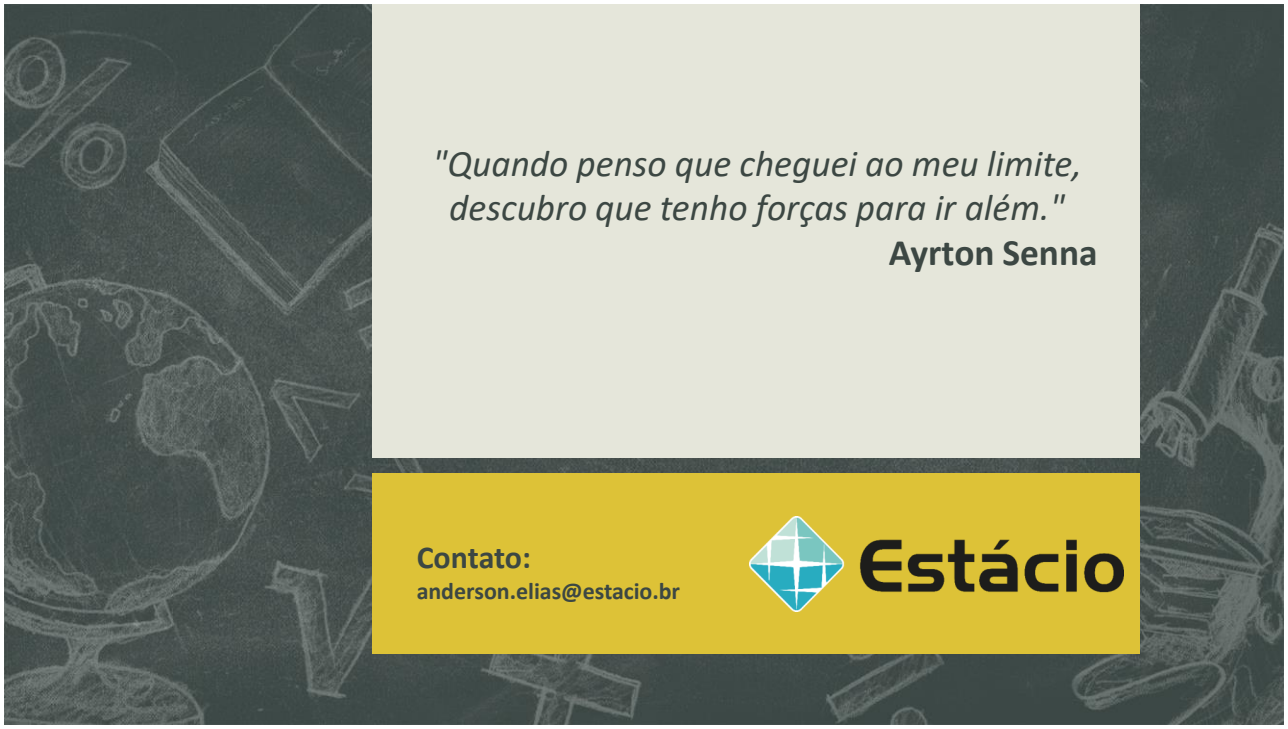
1

Classes Abstratas e Interfaces

Contato:
anderson.elias@estacio.br

Estácio

2



3

Classes, Classes Abstratas e Interfaces

Classe

- Atributos
- Métodos

Classe Abstrata

- Atributos
- Métodos
- Assinatura de Métodos

Interface

- Assinatura de Métodos

4

Classe Abstrata

- Há situações em que não deve haver instâncias de determinadas classes;
- Há modelos incompletos que têm finalidade de servir de base para criação de outros modelos;
- **Às vezes, precisamos definir o quê deve ser feito, mas não como deve ser feito.**

5

Classe Abstrata

- É um tipo especial de classe que não há como criar instâncias dela.
- É usada apenas para ser herdada, funciona como uma superclasse.
- Uma grande vantagem é que força a hierarquia para todas as subclasses.
- É um tipo de contrato que faz com que as subclasses contemplem as mesmas hierarquias e/ou padrões.

6

Classe Abstrata

- São classes mais gerais, servem de base para outras classes;
- Apropriadas para ter atributos que servem para todas as classes que for derivada desta;
- Toda classe que contiver um método abstrato, deve ser considerada uma classe abstrata.
- Toda classe abstrata, nem sempre contém métodos abstratos.

7

Método Abstrato

- Possui nome (identificador), modificadores (abstract obrigatório em classes), tipo de retorno, lista de parâmetros;
- Sem corpo (termina com ponto-e-vírgula);
- Possibilita a chamada a métodos da subclasse a partir de uma referência à superclasse;
- Deve estar em uma classe abstrata ou em uma interface;
- Uma classe não abstrata deve implementar todos os métodos abstratos herdados de uma classe abstrata.

8

Classe Abstrata

- Permitem que se definam métodos sem implementação que devem ser redefinidos em classes derivadas.
- Não podem ser instanciadas.
- O que define se uma classe é abstrata ou não, é a ocorrência de pelo menos um método abstrato.

Observe o uso da palavra reservada **abstract**

```
public abstract class Figura {
    protected int x,y;
    public Figura(int x,int y){
        this.x = x;
        this.y = y;
    }

    abstract public void desenha();
}
```

9

Classe Abstrata

```
public abstract class Ab1 {
  int x = 1;

  public Ab1() {
  }
   void metodo1(int x) {
     System.out.println(x);
  }
  //Assinatura do método abstrato
  abstract public void metodo1();
}
```

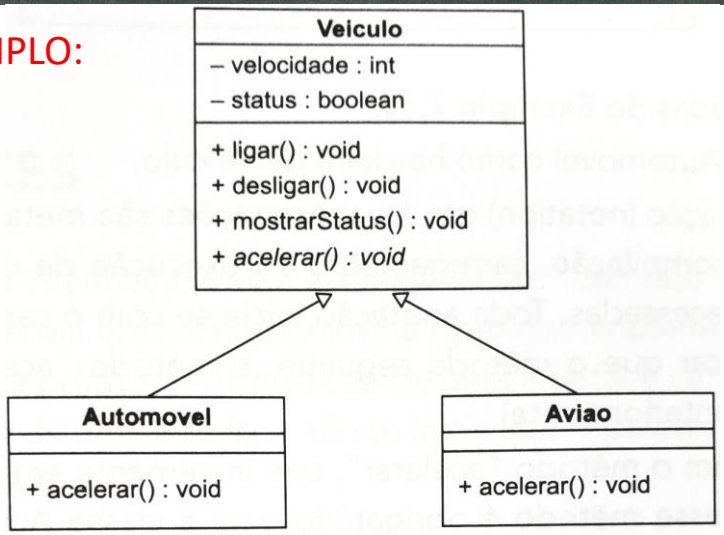
```
1
0
2
1
```

```
public class UsaAb1 extends Ab1{
  int x; //Atributo sem relação com Ab1
  UsaAb1(){
    System.out.println(super.x);
  }
  public void metodo1(){
    this.x = super.x;
  }
  public void metodo1(float s){
    System.out.println(s);
  }
  public void Ab1(){
  }
  public static void main(String[]args){
    UsaAb1 p = new UsaAb1();
    System.out.println(p.x);
    p.x=2;
    System.out.println(p.x);
    p.metodo1();
    System.out.println(p.x);
  }
}
```

10

Classe Abstrata

OUTRO EXEMPLO:



11

Classe Abstrata

```
public abstract class Veiculo {
  public int velocidade;
  public boolean status;

  public void ligar() {
    status = true;
  }
  public void desligar() {
    status = false;
  }
  public void mostrarStatus() {
    System.out.println(status);
  }
  public abstract void acelerar();
}
```

12

Classe Abstrata

```java
public class Automovel extends Veiculo {
  @Override
  public void acelerar() {
    velocidade++;
  }
}
```

```java
public class Aviao extends Veiculo{
  @Override
  public void acelerar() {
    velocidade +=10;
  }
}
```

```java
public class UsaVeiculos {
  public static void main(String[]args) {
    Automovel automovel = new Automovel();
    automovel.ligar();
    automovel.acelerar();
    System.out.println(automovel.velocidade);
    automovel.desligar();

    Aviao aviao = new Aviao();
    aviao.ligar();
    aviao.acelerar();
    System.out.println(aviao.velocidade);
    aviao.desligar();
  }
}
```

13

Classe Abstrata

- Caso se tente criar um objeto da classe Veiculo, será levantada a exceção:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Cannot instantiate the type Veiculo
```

```java
Veiculo(){
   System.out.println("Veiculo");
}
```

- Recapitulando: Não há como criar instâncias de classes abstratas, mesmo que se defina um construtor para elas.

14

Interface

- Uma interface não é considerada uma Classe **e sim uma Entidade**.
- Não possui implementação, apenas assinatura, ou seja, apenas a definição dos seus métodos sem o corpo.
- Todos os métodos são abstratos (***implicitamente***).
- Seus métodos são implicitamente Públicos e Abstratos.

15

Interface

- Não há como fazer uma instância de uma Interface e nem como criar um Construtor.
- Funcionam como um tipo de "**contrato**", onde são especificados os atributos, métodos e funções que as classes que implementem essa interface são obrigadas a implementar.

16

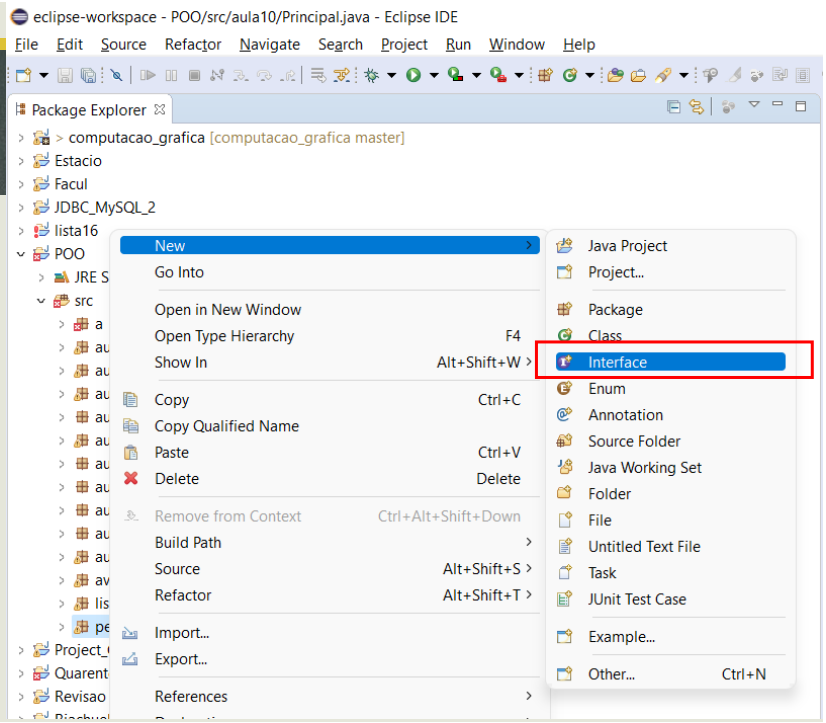
Interface

- Uma classe pode estender uma única classe (abstrata ou não), mas **pode implementar várias interfaces**.
- Todos os atributos em uma interface tem comportamento estático e não podem variar como acontece nas instancias de classes.
- Interfaces são mais apropriadas para comportamento e classes abstratas para implementação.

17

Interface

Para criar uma Interface, realize a operação como na figura ao lado.



18

Interface

```
public interface Anda {
  void avancar(int deslx, int desly);
  void recuar(int deslx, int desly);
}
```

Define um objeto que “anda” (no solo, em qualquer direção)

Se uma classe implementar essa interface, deverá conter o código para os métodos avancar(…) e recuar(…)

19

Interface

```
public interface Voa {
  void subir(int deslAltura);
  void descer(int deslAltura);
}
```

Define um objeto que “voa” (para cima e para baixo)

Se uma classe implementar essa interface, deverá conter o código para os métodos subir(…) e descer(…)

20

Interface

Observe o uso da palavra reservada **implements**

Classe Tartaruga

Tartarugas, em geral, não voam. Portanto, iremos apenas implementar a primeira interface:

```
public class Tartaruga implements Anda{
  private int posx, posy;
  @Override
  public void avancar(int deslx, int desly) {
    posx += deslx;
    posy += desly;
  }
  @Override
  public void recuar(int deslx, int desly) {
    posx -= deslx;
    posy -= desly;
  }
}
```

21

Interface

Classe BemTeVi

Bem-te-vis normalmente voam. Mas também caminham no solo. **Então, iremos implementar ambas interfaces:**

```
public class BemTiVi implements Anda, Voa{
```

```
public class BemTiVi implements Anda, Voa{
  private int posx, posy, altura;
  @Override
  public void subir(int deslAltura) {
    altura += deslAltura;
  }
  @Override
  public void descer(int deslAltura) {
    altura -= deslAltura;
  }
  @Override
  public void avancar(int deslx, int desly) {
    posx += deslx;
    posy += desly;
  }
  @Override
  public void recuar(int deslx, int desly) {
    posx -= deslx;
    posy -= desly;
  }
}
```

22

Interface - atributos

- Se criarmos um atributo em uma Interface, ele terá características como uma constante e será visível por todas as classes que a implementarem.

Alterando a interface Voa, imputando um atributo ***a*** um erro será apresentado.

```
public interface Voa {
  int a;
  void subir(int deslAltura);
  void descer(int deslAltura);
}
```

Para corrigir basta inicializar o atributo.
Ex: `int a=1;`

```
public interface Voa {
  int a=1;
  void subir(int deslAltura);
  void descer(int deslAltura);
}
```

Mas na verdade, o atributo ***a*** terá a seguinte característica:

```
public static final int A = 1;
```

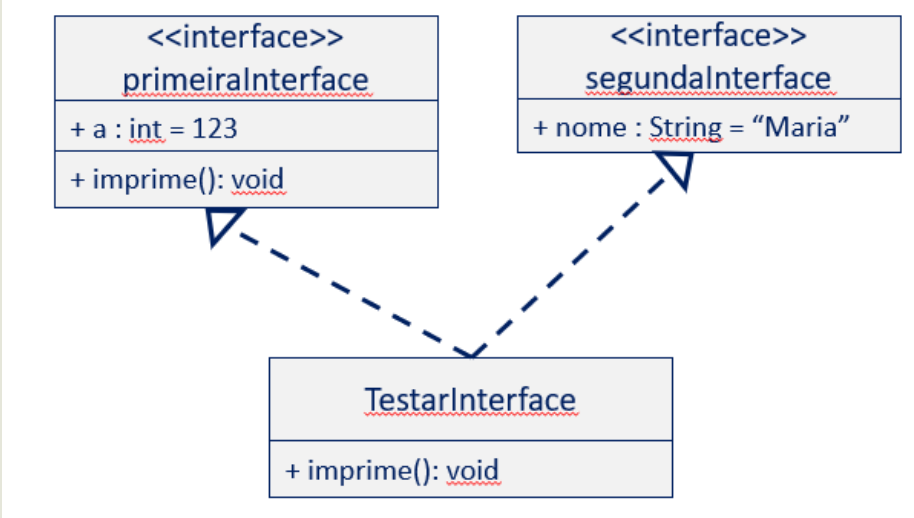
23

Interface - Resumo

- Pode ser vista como uma "**classe abstrata pura**", pois todos os métodos são abstratos
- Não podem ser instanciadas
- Todos os membros são públicos
- Todas as variáveis são ***estáticas*** (de classe) e ***finais*** (constantes)
- Não são estendidas, mas implementadas e pode ser mais de uma(palavra-chave ***implements***)

24

Interface - Resumo

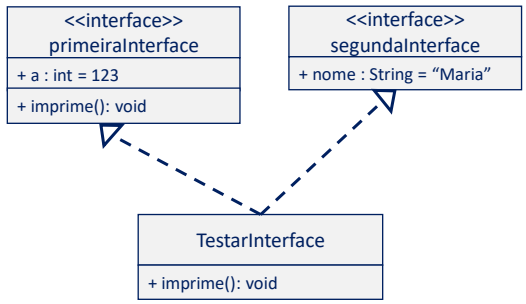


25

Interface - Resumo

```
public interface PrimeiraInterface {
  int a = 123;
  void imprime();
}
```

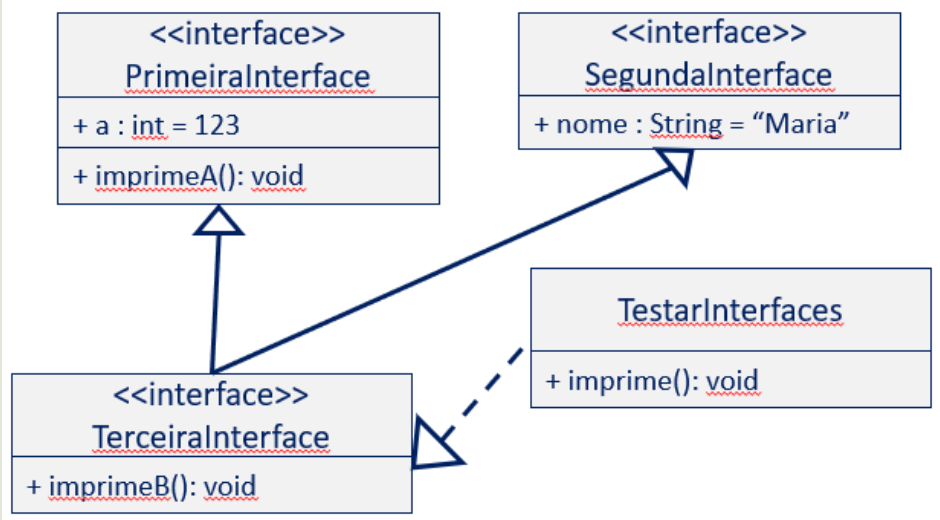
```
public interface SegundaInterface {
  String nome = "Maria";
}
```




```
public class TestarInterfaces implements PrimeiraInterface, SegundaInterface {
  public void imprime() {
    System.out.println("A : " + a);
    System.out.println("Nome: " + nome);
  }
}
```

26

Interface - Resumo

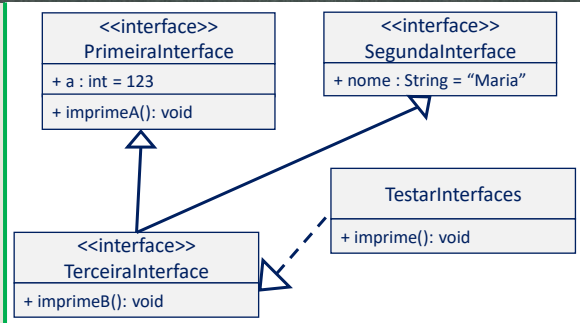


27

Interface - Resumo

COMO FICARIA A IMPLEMENTAÇÃO???

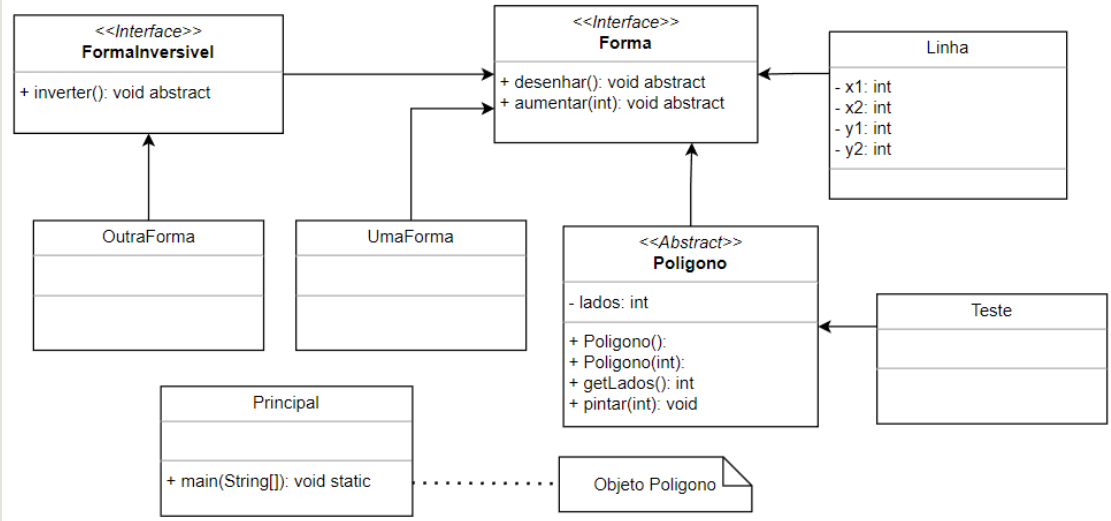
```
public interface TerceiraInterface extends
     PrimeiraInterface, SegundaInterface {
     void imprimeB();
}
```




```
public class TestandoInterfaces implements TerceiraInterface {
  public void imprimeA() {
    System.out.println("A : " + a);  }
  public void imprimeB() {
  System.out.println("Nome: " + nome); }
}
```

28

Exercício (Sala)



29

Resolcução (Sala)

Usando Interfaces hierárquicas.

```
public interface Forma {
   public abstract void desenhar();
   public abstract void aumentar (int t);
}
```

```
public interface FormaInversivel extends Forma {
   public abstract void inverter();
}
```

```
public class UmaForma implements Forma {
   public void desenhar(){
   }
   public void aumentar(int t){
   }
}
```

```
public class OutraForma implements FormaInversivel{
   public void desenhar(){
   }
   public void aumentar(int t){
   }
   public void inverter(){
   }
}
```

30

Interface

Usando Interfaces hierárquicas.

```
abstract public class Poligono implements Forma{
  private int lados;
  public Poligono(){
  }
  public Poligono(int lados){
    this.lados = lados;
  }
  public int getLados(){
    return lados;
  }
  public void pintar(int cor){
  }
}
```

A implementação dos métodos abstratos da interface fica para as subclasses.

```
public class Linha implements Forma {
  private int x1, y1, x2, y2;
  public void desenhar(){
  }
  public void aumentar(int t){
  }
}
```

```
public class Teste extends Poligono {
  public void pintar(int cor){
  }
  public void desenhar(){
  }
  public void aumentar(int t){
  }
}
```

31

(ADICIONAL 1) – Classe Abstrata

A partir do Java 8, podemos...

Criar uma classe principal e instanciar um objeto da classe abstrat Poligono.

Closures

O compilador consegue dar um passo a frente, descobrindo que há construtores.

Permite omitir a declaração do método, podendo haver apenas um método abstrato naquela classe abstrata/interface

```
public class Principal {
  public static void main(String[]args){
    Poligono pol = new Poligono() {
      @Override
      public void desenhar() {

      }
      @Override
      public void aumentar(int t) {

      }
    };
  }
}
```

32

(ADICIONAL 1) – Interface

A partir do Java 8, podemos...

Crie uma interface como segue abaixo.

```
public interface Relogio {
  void ajustaHora();
  default void cronometro(){
    System.out.println("Start...");
  }
}
```

default

Houve uma necessidade de adicionar novas funcionalidades às interfaces sem comprometer as classes que as usa. As classes Orient e Tecnos herdam automaticamente o comportamento default de cronometro da interface Relogio, mas podem ou não sobrescrever esse comportamento.

Crie duas classes que implementa Relogio.

```
public class Orient implements Relogio{
  public void ajustaHora(){
    System.out.println("Hora Ajustada.");
  }
}
```

```
public class Tecnos implements Relogio {
  public void ajustaHora(){
    System.out.println("Hora Ajustada.");
  }
  public void cronometro(){
    System.out.println("Ligando.");
  }
}
```

33

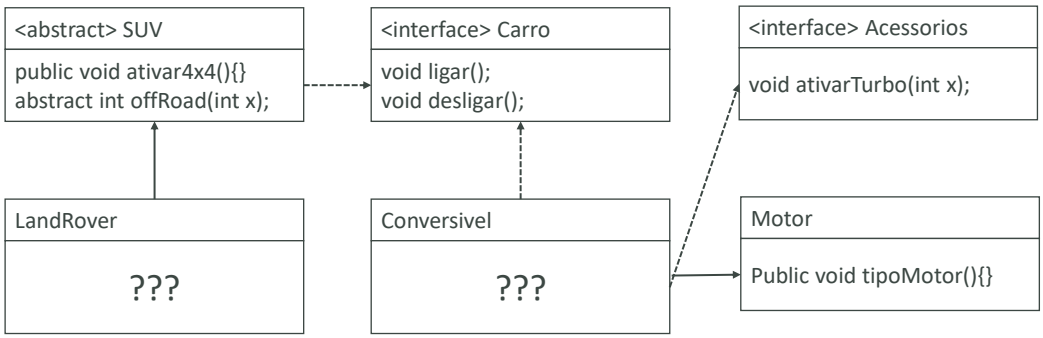
Classe Abstrata X Interface

Quando criamos uma **Classe Abstrata**, estamos criando uma classe base que pode ter um ou mais métodos completos, mas pelo menos um ou mais destes métodos tem de ser incompletos (sem corpo), isto caracteriza uma Classe Abstrata.

Quando criamos uma **Interface**, estamos basicamente criando um conjunto de métodos sem qualquer implementação que deve ser herdado por outras classes já implementadas. A vantagem é que desta forma consegue-se prover um caminho para uma classe ser parte de duas classes: uma herdada hierarquicamente e outra da Interface.

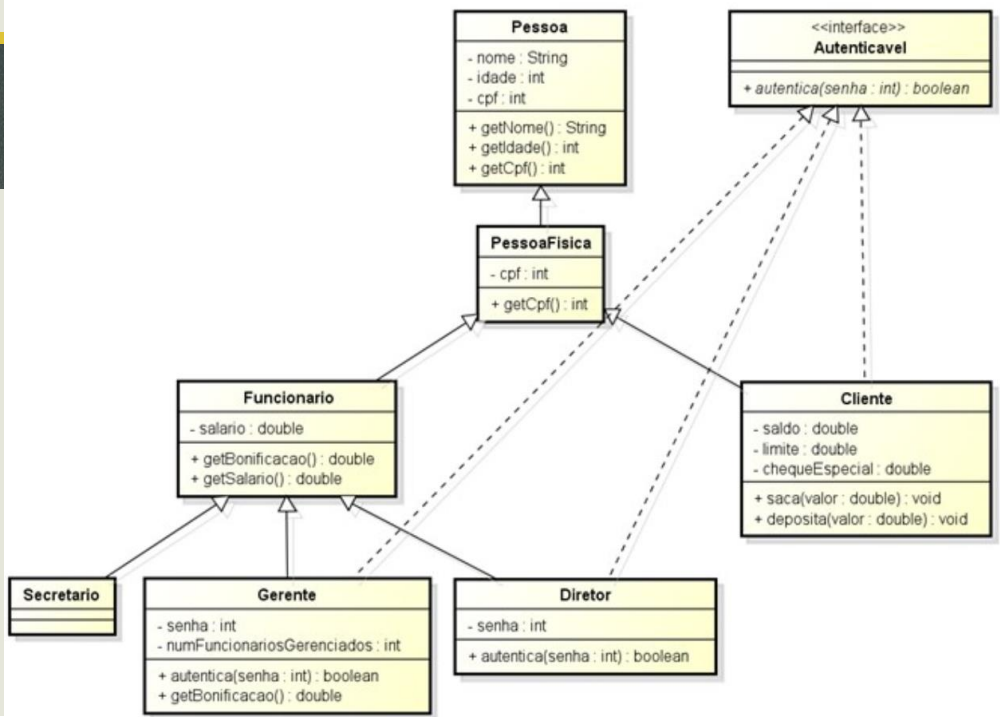
34

EXERCÍCIO



35

Exercício



36

Dúvidas?

Estácio

37