accenture

Strategy | Digital | Technology | Operations

High performance. Delivered.

Application Delivery Fundamentals: Java

Module 6: Inheritance

Introduction

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

MIT Faculty Video:

“Inheritance”

and

“Virtual Methods and Polymorphism”

Copyright © 2020 Accenture All Rights Reserved.

Module Objectives

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

No final deste módulo, os participantes serão capazes de:

- Definir herança
- Descreva o relacionamento 'is-a'
- Explique como usar a palavra-chave 'extends' para definir um relacionamento de herança
- Identifique os efeitos dos modificadores de acesso em um relacionamento de herança
- Explicar como substituir métodos herdados
- Definir classes abstratas e seu uso
- Definir interfaces e seu uso
- Descreva a diferença entre estender de uma classe e implementar interfaces
- Discutir referências de conversão para cima / para baixo
- Discutir a chamada de método virtual em Java

Copyright © 2007 Accenture All Rights Reserved

Herança

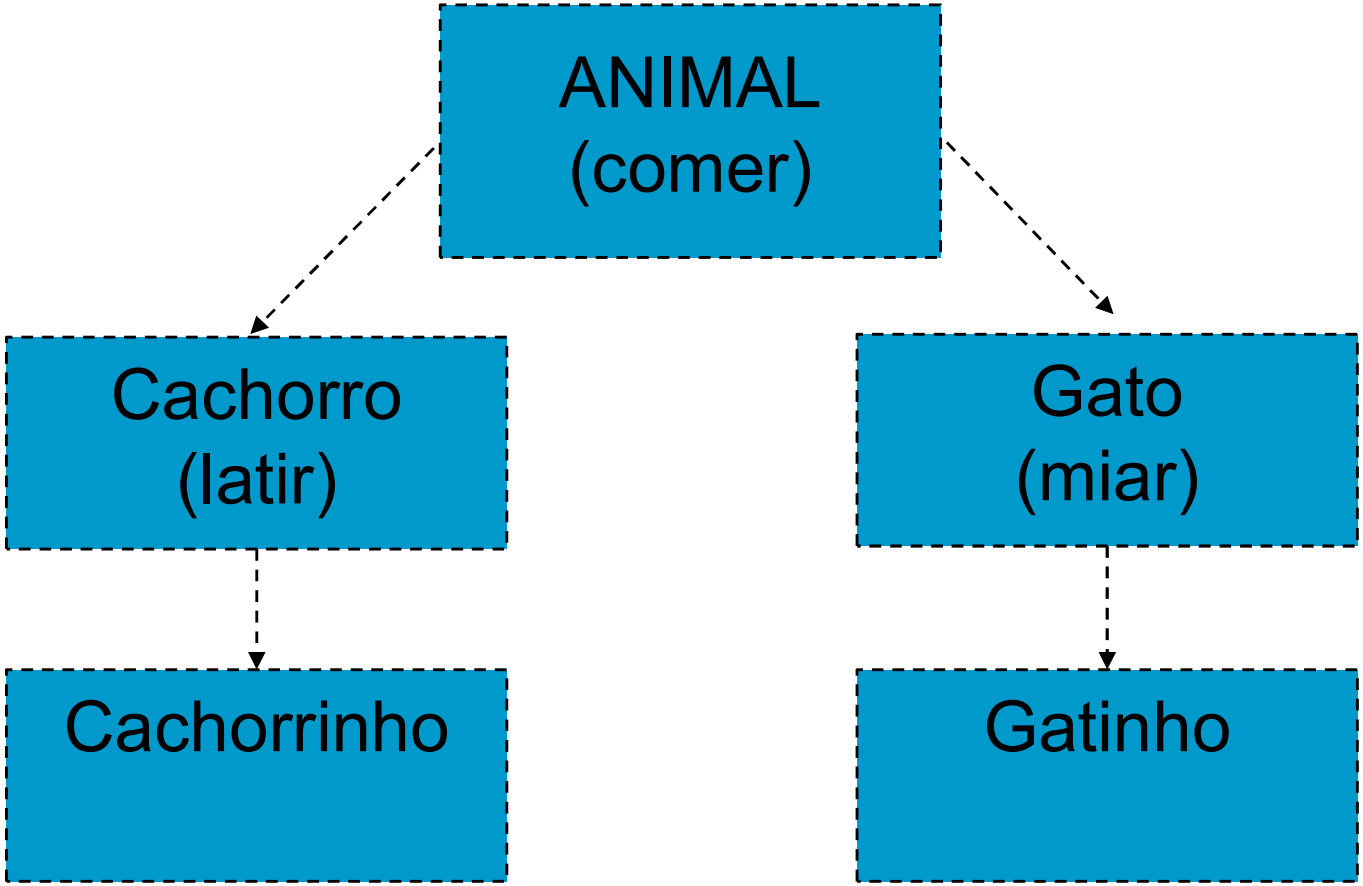
2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

“Herança”

Copyright © 2020 Accenture All Rights Reserved.

Introduction

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070



Copyright © 2020 Accenture All Rights Reserved.

Introduction

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

```java
public class Animal {

    public void comer(){
        System.out.println("Comendo…");
    }

}
```

```java
public class Cachorro extends Animal{

    public void latir(){
        System.out.println("Latindo…");
    }

}
```

```java
public class Gato extends Animal {

    public void miar(){
        System.out.println("Miando…");
    }

}
```

```java
public class Cachorrinho extends Cachorro {

}
```

```java
public class Gatinho extends Gato {

}
```

Copyright © 2020 Accenture All Rights Reserved.

Introduction

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

```java
public class Animal {

    public void comer(){
        System.out.println("Comendo…");
    }

}
```

```java
public class Cachorro extends Animal{

    public void latir(){
        System.out.println("Latindo…");
    }

}
```

```java
public class Gato extends Animal {

    public void miar(){
        System.out.println("Miando…");
    }

}
```

```java
public class Cachorrinho extends Cachorro {

}
```

```java
public class Gatinho extends Gato {

}
```

Copyright © 2020 Accenture All Rights Reserved.

Definindo Herança

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- **A herança** é uma das construções de linguagem que incentiva o reuso do código, permitindo que o comportamento das classes existentes seja estendido e especializado
- **A herança** define um relacionamento hierárquico entre classes em que uma classe compartilha os atributos e métodos definidos em uma ou mais classes.

- Uma classe da qual os atributos e comportamentos são derivados → Super class/ Base class/Parent class

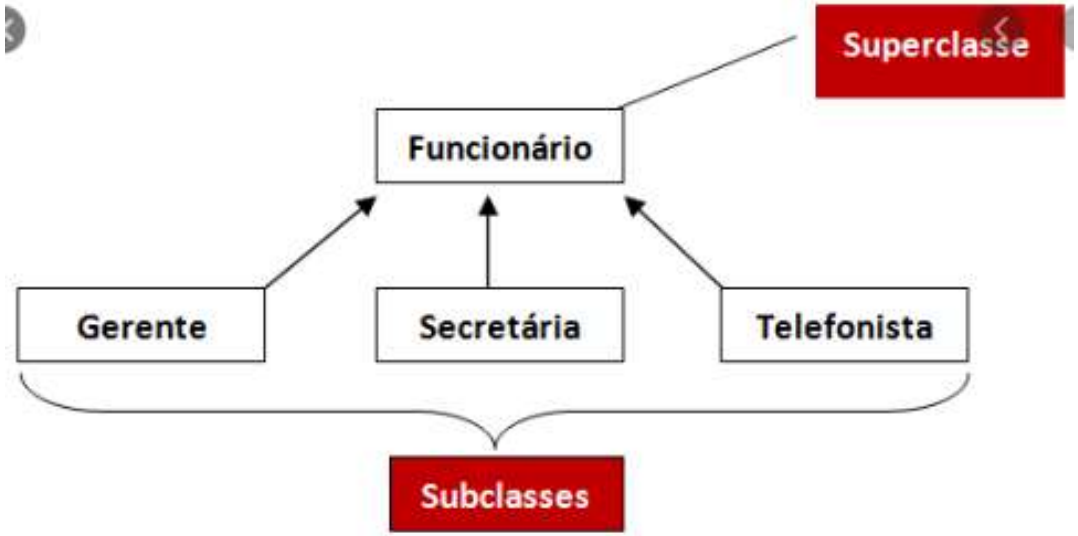
- Uma classe que recebe atributos e comportamentos de outra classe → *Subclass*/Derived class/Child class

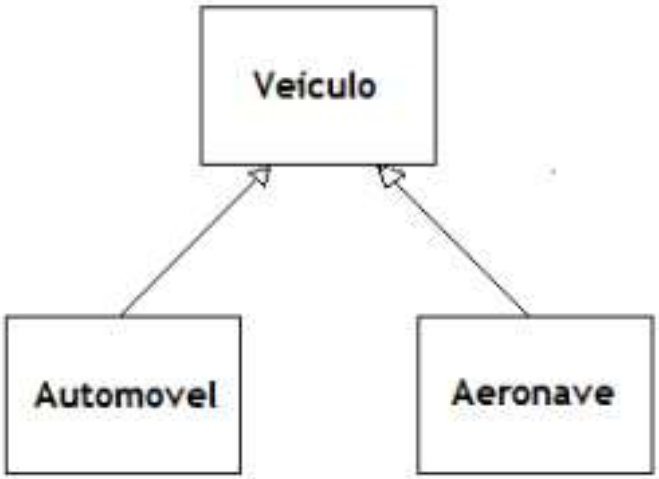
Copyright © 2020 Accenture All Rights Reserved.

Relacionamento ‘é um’

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Um relacionamento de herança é descrito como um relacionamento “é-um" entre duas classes;
- Classe B “é-uma” Classe A significa que a Classe B é uma especialização (subclasse) do Classe A;
- A Classe B terá todos os atributos e comportamento da Classe A, além de seus próprios atributos e comportamento exclusivos

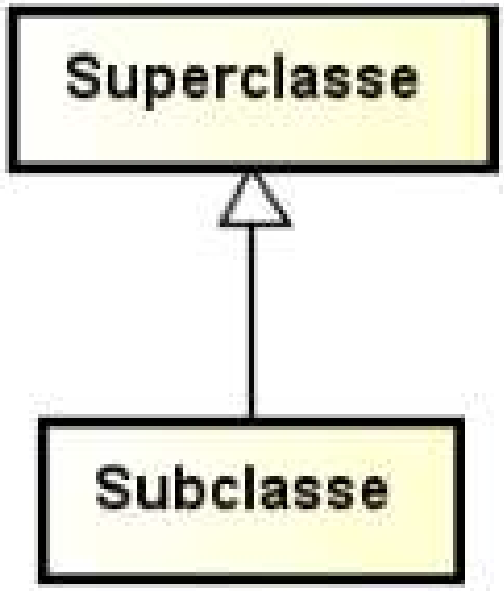


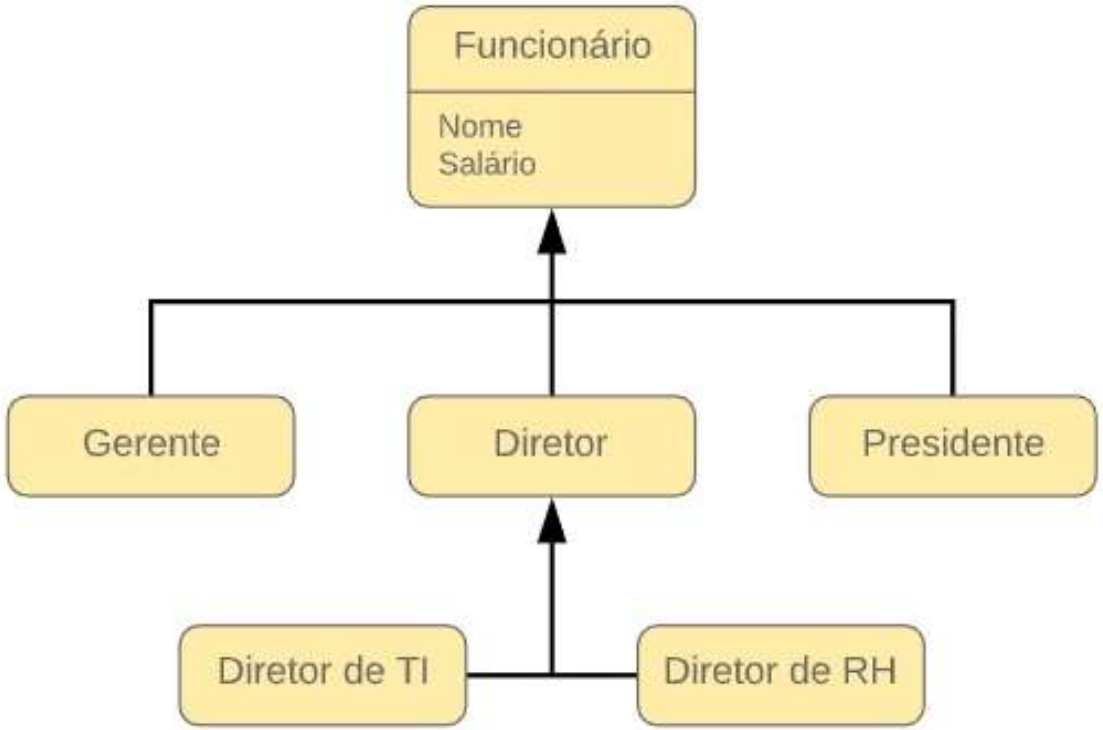


Copyright © 2020 Accenture All Rights Reserved.

Inheritance Hierarchy

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070



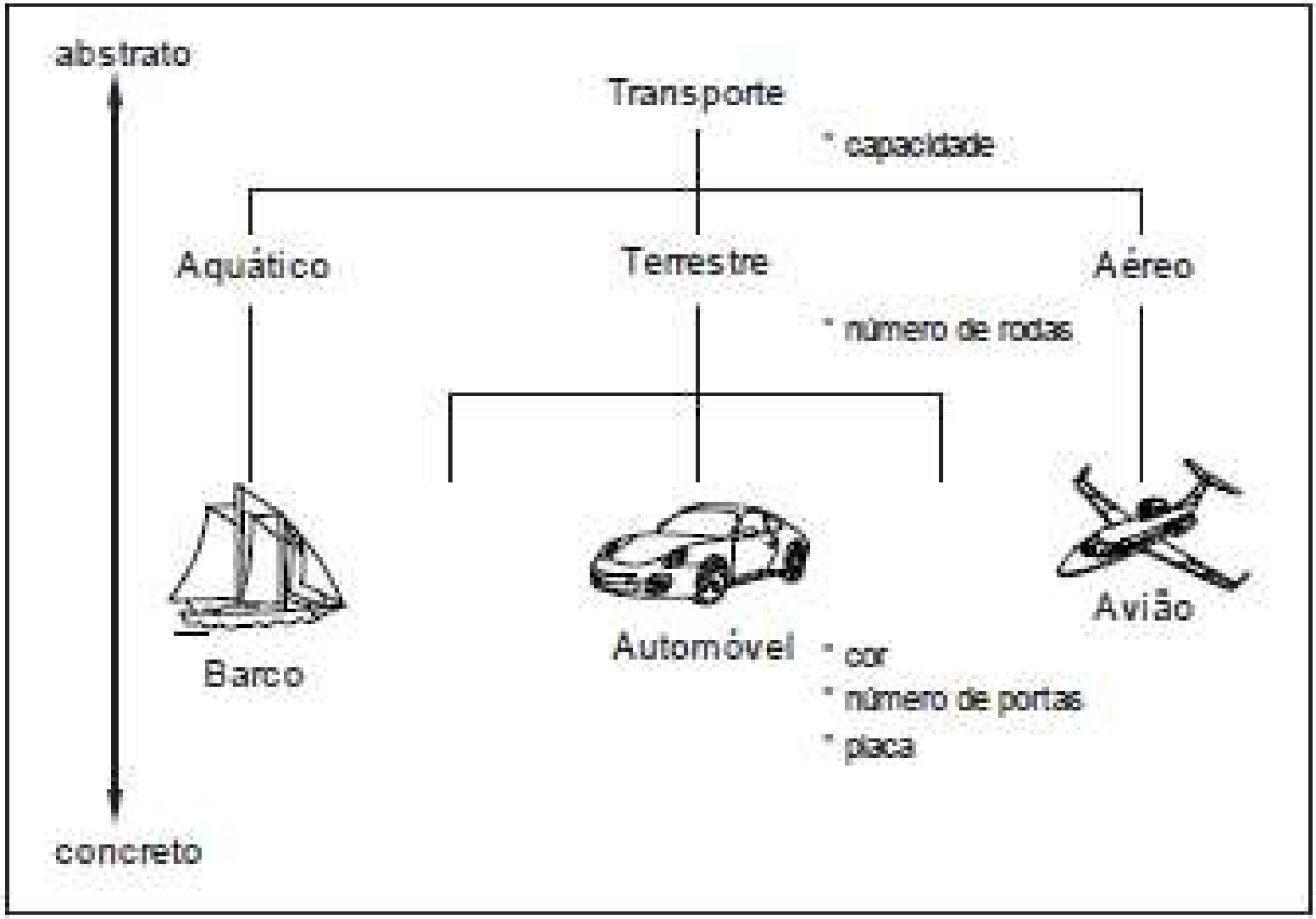


Copyright © 2020 Accenture All Rights Reserved.

Inheritance Hierarchy

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Herança de classe:



Copyright © 2020 Accenture All Rights Reserved.

Herança

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- A herança é implementada em Java com a palavra-chave ***extends*** durante a declaração de classe
- Ao estender outra classe, todos os atributos e comportamento da classe pai são herdados automaticamente pela classe filho

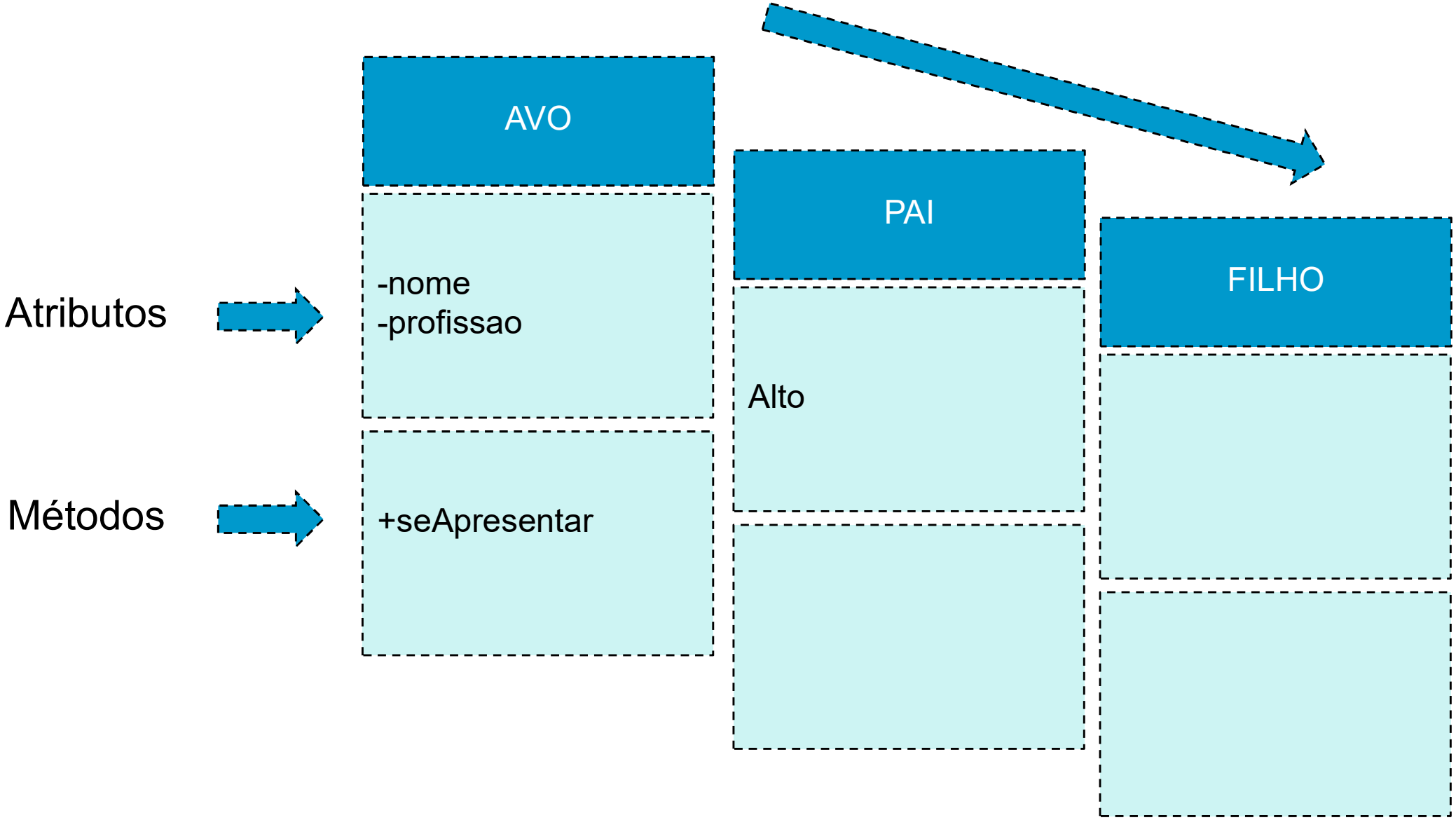
```
Example:
public class Cat extends Animal{
    //Definir os atributos de um gato
}
```

Refer to example Student_I.java, Person_I.java, InheritanceSample.java inside package sef.module6.sample.

Copyright © 2020 Accenture All Rights Reserved.

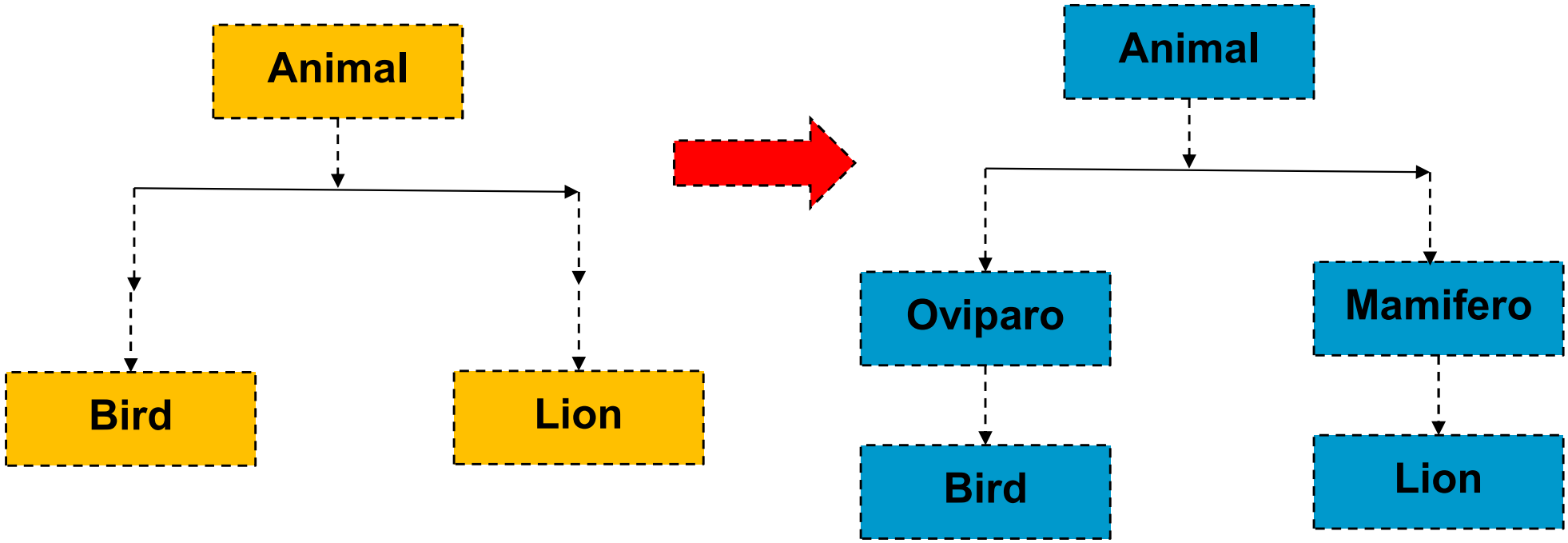
Herança

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070



Copyright © 2020 Accenture All Rights Reserved.

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070



Copyright © 2020 Accenture All Rights Reserved.

Inheritance and Access Modifiers

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Campos públicos e protegidos são herdados e são acessíveis por todas as subclasses;
- Campos privados não são herdados por uma subclasse;
- Métodos públicos e protegidos da superclasse são herdados por subclasses;
- Esses métodos públicos e protegidos na superclasse podem ser usados para acessar campos / métodos privados da superclasse.

i

Refer to example Student_I.java, Person_I.java, InheritanceSample.java inside package sef.module6.sample.

Copyright © 2020 Accenture All Rights Reserved.

Inheritance and Keyword “super”

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- A palavra-chave ***super*** permite que uma subclasse faça referência à um campo ou método que pertence à sua classe pai imediata. O método ***super*** (<parameters>) pode ser chamado para se referir à um construtor de classe pai.

```
class Child extends Parent{
    public Child(){
        //somente o constructor filho pode chamar o pai
        super(“John Doe”);
    }
}
```

- O ***super***.<*field*> pode ser usado para acessar um campo ou um método que pertence ao pai.

```
super.aParentMethod();
```

Refer to example Student_I.java, Person_I.java, InheritanceSample.java inside package sef.module6.sample.

Copyright © 2020 Accenture All Rights Reserved.

Activity – Inheritance

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Nesta atividade, você vai :
  - Abra o arquivo ‘InheritanceActivity.java’ no pacote sef.module6.activity.
  - Leia as instruções e crie o código para completar este programa.

Copyright © 2020 Accenture All Rights Reserved.

OverLoad & Overriding

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

“Overload & Overriding”

Copyright © 2020 Accenture All Rights Reserved.

Overload

2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696

```
10  public class Classe {

11

12      private int numa;
13      private int numb;
14

15      public Classe(){
16          this.numa = 2;
17          this.numb = 2;
18      }
19

20      public Classe(int numa, int numb){
21          this.numa = numa;
22          this.numb = numb;
23      }
24

25      public int somaValores(){
26          return numa+numb;
27      }
28

29      public void somaValores(int a, int b){
30          this.numa = a;
31          this.numb = b;
32          int total = a+b;
33      }
34

35      public int somaValores(double b, double a){
36          double total = a+b;
37          int contotal = (int)total;
38          return contotal;
39      }
```

Sobrecarga (Overload) de Construtores - O primeiro não possui parâmetro, já o segundo possui.

Sobrecarga de Métodos, todos eles tem a mesma função, mais recebem parametros diferentes, e tem retornos diferentes, ou nenhum retorno.

Copyright © 2020 Accenture All Rights Reserved.

Overloading and Overriding

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

	Overloading	Overriding
Description	A Sobrecarga de método permite que uma subclasse redefina métodos com o mesmo nome de método, mas com um número diferente de parâmetros ou tipos diferentes de parâmetros na lista de parâmetros.	A Substituição de Método permite que uma subclasse redefina os métodos da mesma assinatura da superclasse.
Requirements	Um método de sobrecarga deve ter: • O mesmo nome • Número diferente de parâmetros ou tipos • O mesmo ou diferente tipo de retorno	Um método substituído deve ter: • O mesmo nome • O mesmo número de parâmetros e tipos • O mesmo tipo de retorno

Refer to the samples Person_P.java, Student_P.java and PolymorphismSample.java inside package sef.module6.sample.

Copyright © 2020 Accenture All Rights Reserved.

Overloading and Overriding (cont.)

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

	Overloading	Overriding
Binding Type	O método de sobrecarga a ser chamado é baseado no tipo e número de parâmetros e é decidido no momento da compilação. Isso é chamado como Static Binding	O método substituído a ser chamado é baseado no tipo de objeto real e decidido no tempo de execução. Isso é chamado como dynamic Binding.
Sample	Refer to the samples Person_P.java, Student_P.java and PolymorphismSample.java inside package *sef.module6.sample.* *Explain to the participants how method* ***address****() is overloaded.*	Refer to the samples Person_P.java, Student_P.java and PolymorphismSample.java inside package *sef.module6.sample.* *Explain to the participants how method* ***announce****() is overridden.*

Refer to the samples Person_P.java, Student_P.java and PolymorphismSample.java inside package sef.module6.sample.

Copyright © 2020 Accenture All Rights Reserved.

Polymorphism – Static Binding and Dynamic Binding

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

	Static Binding	Dynamic Binding
Definição	Capacidade de chamar comportamento específico (método) no tempo de compilação com base na assinatura do método	Capacidade de definir / adiar o comportamento específico das subclasses em tempo de execução.
Como alcançar	Pode ser alcançado através do método Overloading	Pode ser alcançado através do método Overriding

Copyright © 2020 Accenture All Rights Reserved.

Polimorfismo

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

“Polimorfismo”

Copyright © 2020 Accenture All Rights Reserved.

Polymorphism

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Exemplo do Xadrez:

```java
3   public class Peao extends Peca{
4       @Override
6       public void mover() {
7           System.out.println("ANDAR PRA FRENTE");
7       }
8   }
```

```java
1   package xadrez;
2
    public abstract class Peca {
        public abstract void mover();
5   }
```

```java
3   public class Cavalo extends Peca{
4       @Override
6       public void mover() {
7           System.out.println("ANDAR EM 'L'");
7       }
8   }
```

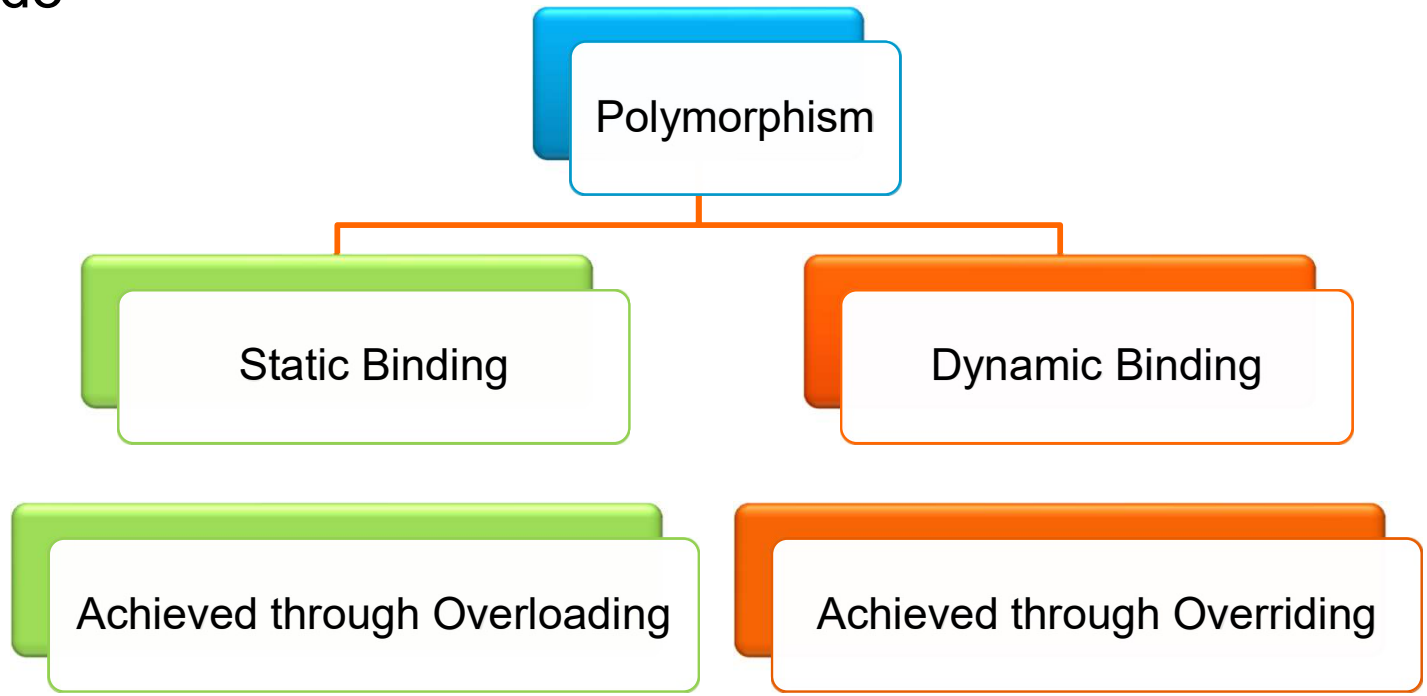
```java
3   public class Bispo extends Peca{
4       @Override
6       public void mover() {
7           System.out.println("ANDAR NA DIAGONAL");
7       }
8   }
```

Copyright © 2020 Accenture All Rights Reserved.

Polymorphism

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- O polimorfismo é um dos princípios básicos da programação orientada a objetos
- Polimorfismo significa "muitas formas"
- Refere-se à capacidade de uma variável de referência alterar o comportamento de acordo com a instância do objeto que está mantendo

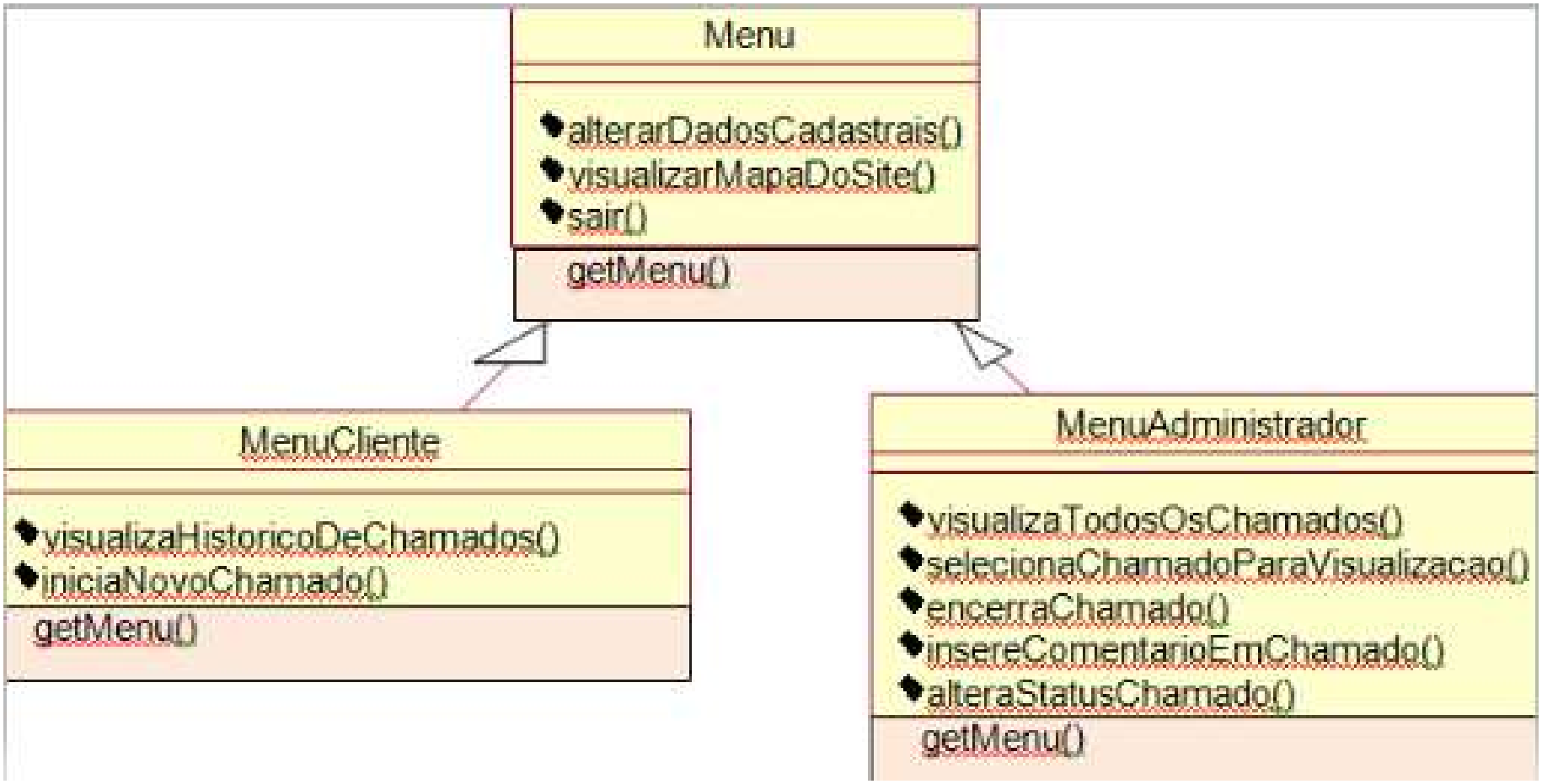


Copyright © 2020 Accenture All Rights Reserved.

Polymorphism

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- **Polimorfismo** é o princípio pelo qual duas ou mais classes derivadas da mesma superclasse podem invocar métodos que têm a mesma assinatura, mas comportamentos distintos;



Copyright © 2020 Accenture All Rights Reserved.

Perguntas:

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

O que é herança?

A herança é um mecanismo da Orientação a Objeto que permite criar novas classes a partir de classes já existentes

O que é overloading?

Sobrecarga (Overload) de métodos em Java. A sobrecarga, ou overload em inglês, permite a existência de vários métodos com o mesmo nome

O que é overriding?

A sobrescrita de um método ocorre quando uma classe filha implementa um método que já existe numa classe mãe, alterando (sobrescrevendo) o comportamento existente.

O que é polimorfismo?

Polimorfismo denota uma situação na qual um objeto pode se comportar de maneiras diferentes ao receber uma mensagem

Copyright © 2020 Accenture All Rights Reserved.

Classe Abstrata

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

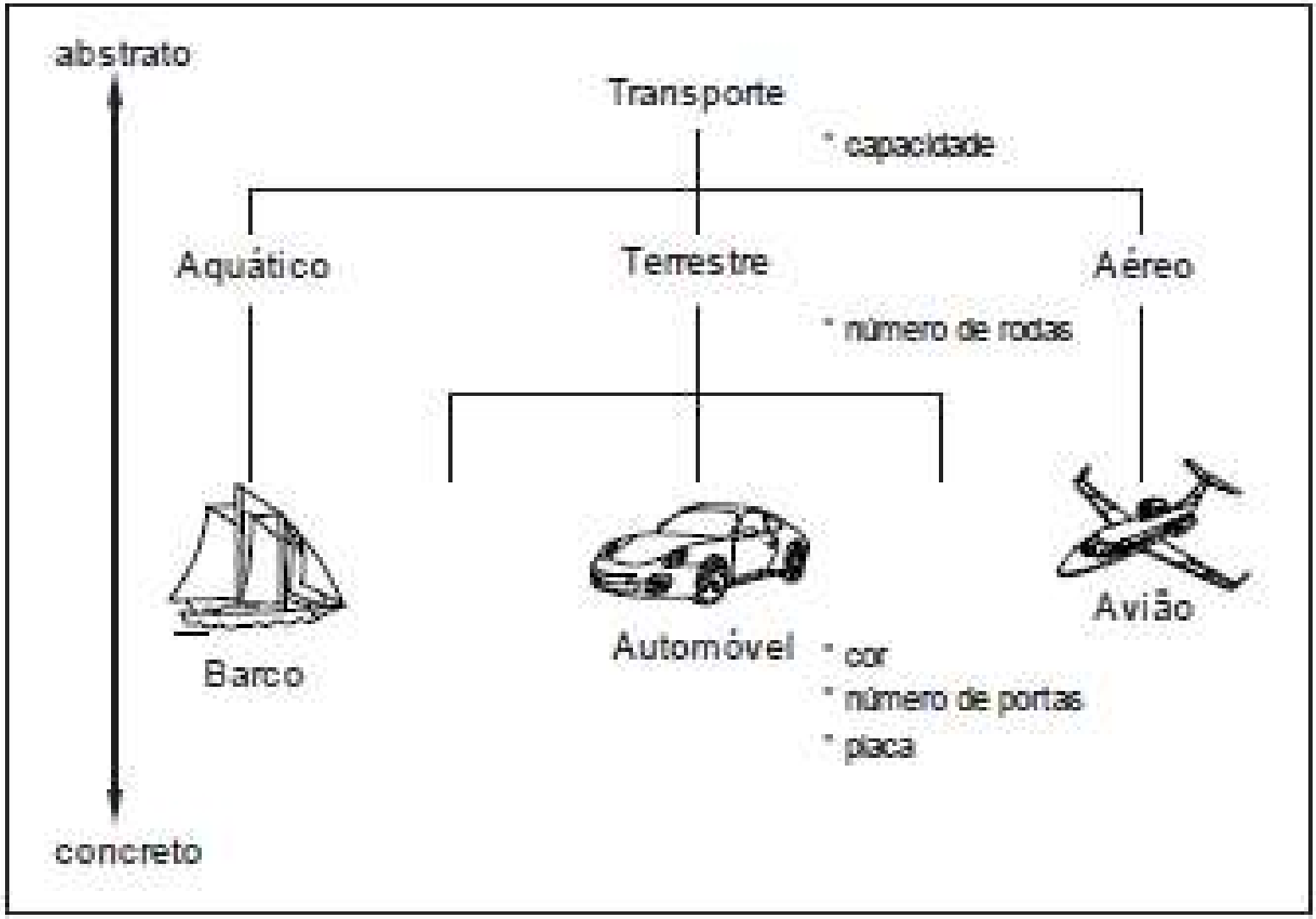
“Classe abstrata”

Copyright © 2020 Accenture All Rights Reserved.

Classes

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Classe Abstrata:



Copyright © 2020 Accenture All Rights Reserved.

Classe Abstrata

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Classes Abstratas

- Uma classe abstrata define apenas parte da implementação
- Não podemos criar objetos para uma classe abstrata
- Deixamos as subclasses fornecerem a implementação que falta da classe abstrata

Copyright © 2020 Accenture All Rights Reserved.

Classe Abstrata

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

```
public abstract class PecaXadrez {
  public abstract boolean movimentoValido(int x, int y);
  public void move(int x, int y) {
      if (movimentoValido(x,y)) {
            //  faz movimento
      } else {
            // erro
      }
  }
}
```

A classe PecaXadrez declara a existência do método movimentoValido mas não o implementa.

Copyright © 2020 Accenture All Rights Reserved.

Classe Abstrata

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Subclasses de classes abstratas têm que implementar todos os métodos abstratos declarados na classe pai. Caso contrário elas também serão classes abstratas.

```
public class Rainha extends PecaXadrez {
  public boolean movimentoValido(int x, int y){
    // faz consistência do movimento para a Rainha
  }
}
public class Rei extends PecaXadrez {
  public boolean movimentoValido(int x, int y){
    // faz consistência do movimento para o Rei
  }
}
```

Copyright © 2020 Accenture All Rights Reserved.

Defining Abstract Class

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Uma classe abstrata é uma classe que fornece comportamento comum em um conjunto de subclasses, mas não é projetada para ter instâncias próprias
- Uma classe abstrata é projetada como um modelo para outras classes seguirem ditando o comportamento que deve ser implementado por suas subclasses
- Uma classe abstrata pode estender uma classe, uma classe abstrata ou implementar uma interface
- Uma classe abstrata pode implementar uma ou várias interfaces
- Uma classe abstrata pode estender apenas uma classe abstrata
- Usando o modificador de classe "abstrato"

```
public abstract class Food{
    public abstract double calculateCalories();
}
```

Copyright © 2020 Accenture All Rights Reserved.

Activity – Abstract Class

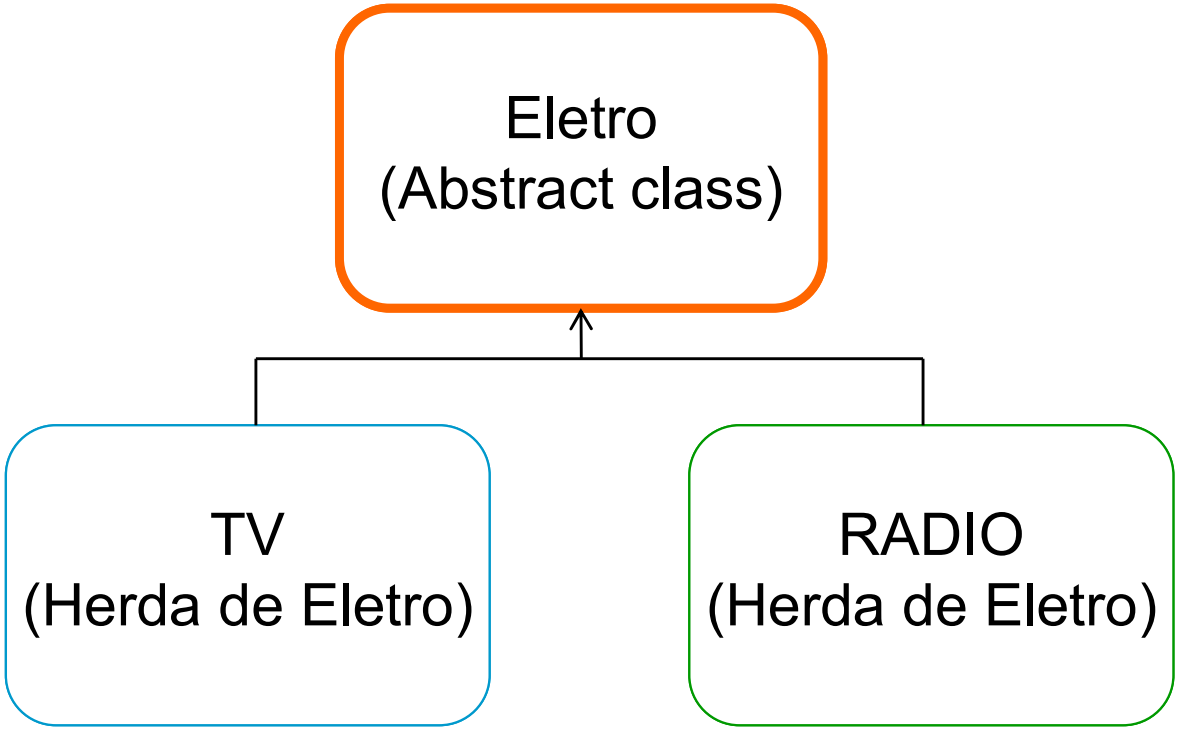
2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Nesta atividade, você vai :
  - Abra o arquivo ‘Shape.java’ no pacote sef.module6.activity.
  - Leia as instruções e crie o código para concluir este programa.

Copyright © 2020 Accenture All Rights Reserved.

Defining Abstract Class

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070



Refer to the Eletrodomestico.java inside package sef.module6.sample.

Copyright © 2020 Accenture All Rights Reserved.

Interface

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

“Interface”

Copyright © 2020 Accenture All Rights Reserved.

Defining Java Interface

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Uma interface é 100% uma classe abstrata;
- Uma interface especifica um conjunto de métodos ou modelos que uma classe de implementação precisa seguir;
- Uma interface fornece apenas uma forma para uma classe, mas nenhuma implementação;
- Uma interface define o que uma classe pode fazer, mas não como a classe fará.

Copyright © 2020 Accenture All Rights Reserved.

Interface

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Interfaces

- Uma interface não tem implementação, todos os métodos são abstratos.
- Uma interface pode definir variáveis estáticas do tipo final (constantes)

```
public interface Cor {
  public int getCor( );
  public void setCor(int cor);
}
```

Copyright © 2020 Accenture All Rights Reserved.

Interface

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Múltiplas Interfaces

- A classe pode estender apenas uma classe, mas implementar múltiplas interfaces

Copyright © 2020 Accenture All Rights Reserved.

Implementing Interfaces

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- A interface define o método que será implementado pela classe;

```
class Myclass implements MyInterface<Integer>{

    public void myMethod(Integer i){

        System.out.println("MyMethod -> i -> "+i);
    }
}
```

```
interface MyInterface {

    void MyMethod(Integer t);
}
```

Copyright © 2020 Accenture All Rights Reserved.

Interface

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- A classe que implementa uma interface deve definir os métodos da interface

```
interface ObjetoSeguro {
  void abre(int segredo );
}

class Cofre implements ObjetoSeguro {
  void abre(int segredo ) {
    if (segredo == 1234) {
      System.out.println("Cofre Aberto");
    }
  }
}
```

Copyright © 2020 Accenture All Rights Reserved.

Implementing Interfaces

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Uma classe que implementa interfaces é necessária para substituir os métodos herdados;
- As interfaces são implementadas usando a palavra-chave **implements**
- Regras para implementar os métodos de interface
  - Deve ter a mesma assinatura de método e tipo de retorno
  - Não é possível restringir a acessibilidade do método
  - Não é possível especificar exceções verificadas mais amplas
- Variáveis de interface são implicitamente `public final static`
- Métodos de interface são implicitamente `public abstract`

Copyright © 2020 Accenture All Rights Reserved.

Rules on Interface

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

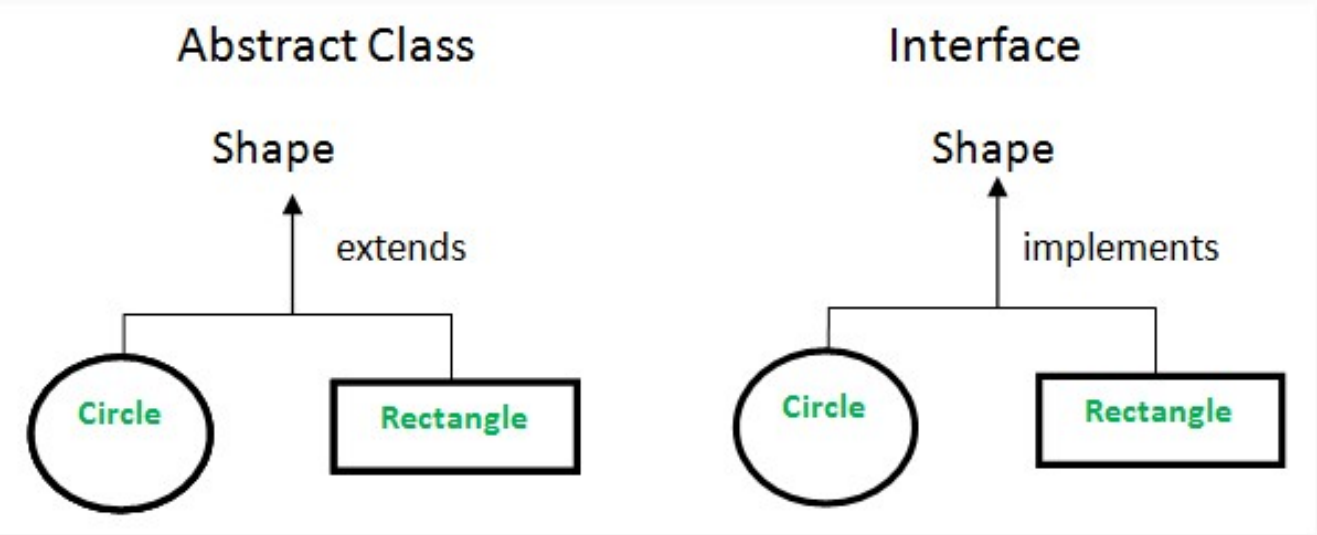
- Uma interface pode estender várias interfaces
- As interfaces podem ser implementadas por qualquer classe
- Uma classe pode implementar várias interfaces
- Uma classe que implementa uma interface parcialmente deve ser declarada abstrata
- Uma interface pode ser declarada como uma variável de referência
- Uma interface não pode ser instanciada

Copyright © 2020 Accenture All Rights Reserved.

Interfaces vs Classe abstrata

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- A classe **implements** uma interface;
- A classe **extends** uma classe abstrata;



Copyright © 2020 Accenture All Rights Reserved.

Interfaces quando usar?

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Considere o uso de interfaces se alguma dessas declarações se aplicar à sua situação:

- É uma abstração total. Todos os métodos declarados em uma interface devem ser implementados pela (s) classe (s) que implementam essa interface.
- Uma classe pode implementar mais de uma interface. É chamado de herança múltipla.
- Você deseja especificar o comportamento de um determinado tipo de dados, mas não se preocupa com quem implementa seu comportamento.

Copyright © 2020 Accenture All Rights Reserved.

Classes abstratas quando usar?

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Considere o uso de classes abstratas se alguma dessas declarações se aplicar à sua situação:
- No aplicativo java, existem algumas classes relacionadas que precisam compartilhar algumas linhas de código, então você pode colocar essas linhas de código dentro da classe abstrata e essa classe abstrata deve ser estendida por todas essas classes relacionadas.
- Você pode definir campo (s) não static (s) ou não final (is) na classe abstrata, de modo que, por meio de um método, você possa acessar e modificar o estado do objeto ao qual eles pertencem.
- Você pode esperar que as classes que estendem uma classe abstrata tenham muitos métodos ou campos comuns ou exijam modificadores de acesso diferentes de públicos (como protected e private).

Copyright © 2020 Accenture All Rights Reserved.

Atividade 1

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Neste exercício, você vai criar uma hierarquia de classes usando uma classe abstrata para representar diferentes tipos de contas bancárias.

Crie uma classe abstrata chamada **ContaBancaria** com os seguintes atributos e métodos:

- Atributos:
  - numeroConta (String)
  - saldo (double)
- Métodos:
  - Construtor para inicializar todos os atributos.
  - Método depositar(double valor) que adiciona o valor ao saldo.
  - Método abstrato sacar(double valor).
  - Método abstrato transferir(valor, conta2).
  - Método exibirDetalhes() que imprime os detalhes da conta.

Copyright © 2020 Accenture All Rights Reserved.

Atividade 1

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Crie uma classe chamada **ContaCorrente** que herda da classe **ContaBancaria** e adiciona os seguintes atributos e métodos:
  - **Atributos**:
    - taxaDeOperacao (double)
  - **Métodos**:
    - Construtor para inicializar todos os atributos (incluindo os da classe base).
    - Implementação do método sacar(double valor) que desconta a taxa de operação ao sacar.
    - Implementação do método transferir(valor, conta2) que que sacce de uma conta e deposita na outra..
    - Sobrescrever o método exibirDetalhes() para incluir a taxa de operação.

Copyright © 2020 Accenture All Rights Reserved.

Atividade 1

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Crie uma classe chamada ContaPoupanca que herda da classe ContaBancaria e adiciona os seguintes atributos e métodos:

- **Atributos**:
  - taxaDeJuros (double)
- **Métodos**:
  - Construtor para inicializar todos os atributos (incluindo os da classe base).
  - Implementação do método sacar(double valor) que permite o saque apenas se houver saldo suficiente.
  - Implementação do método transferir(valor, conta2) que permite o saque apenas se houver saldo suficiente.
  - Método aplicarJuros() que aplica a taxa de juros ao saldo.
  - Sobrescrever o método exibirDetalhes() para incluir a taxa de juros.

No método **main**, crie objetos das classes **ContaCorrente** e **ContaPoupanca**, inicialize-os com valores, e realize operações de depósito, transferência e saque. Chame o método **exibirDetalhes**() para cada uma das contas.

Copyright © 2020 Accenture All Rights Reserved.

Questions and Comments

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- What questions or comments do you have?

Copyright © 2020 Accenture All Rights Reserved.