

POO: REVISÃO

Me. Nator Junior

Faculdade Senac Ceará - Av. Tristão Gonçalves, 1245 - Bairro Centro CEP 60015-000 -
Fortaleza - CE

3 de junho de 2024

- 1 **Resumo**
- 2 Modificadores de Acesso
- 3 Herança
- 4 Polimorfismo
- 5 Polimorfismo e Interface
- 6 QUESTÕES
- 7 SQL

- POO é um paradigma de programação baseado em "objetos"
- Objetos são instâncias de classes, que podem conter dados e métodos
- Facilita a modelagem de sistemas complexos

Objeto
{ dados
métodos }



Classe
define

- Classe: um molde para criar objetos
- Objeto: uma instância de uma classe

```
1 public class Pessoa {  
2     private String nome;  
3     private int idade;  
4  
5     public Pessoa(String nome,  
6         int idade) {  
7         this.nome = nome;  
8         this.idade = idade;  
9     }  
10  
11 Pessoa p1 = new Pessoa("Ana",  
    30);
```

- Proteção dos dados internos de um objeto
- Uso de métodos getters e setters

```
1 public class Pessoa {  
2     private String nome;  
3     private int idade;  
4  
5     public Pessoa(String nome,  
6         int idade) {  
7         this.nome = nome;  
8         this.idade = idade;  
9     }  
10    public String getNome() {  
11        return nome;  
12    }  
13  
14    public void setNome(String  
15        nome) {  
16        this.nome = nome;  
17    }  
}
```

- Reutilização de código entre classes
- Classe derivada herda atributos e métodos da classe base

```
1 public class Animal {  
2     protected String nome;  
3  
4     public Animal(String nome) {  
5         this.nome = nome;  
6     }  
7  
8     public void fazerSom() {  
9         System.out.println("Som do animal");  
10    }  
11 }  
12  
13 public class Cachorro extends Animal {  
14     public Cachorro(String nome) {  
15         super(nome);  
16     }  
17  
18     @Override  
19     public void fazerSom() {  
20         System.out.println("Latido");  
21     }  
22 }
```

- Capacidade de utilizar métodos de forma intercambiável
- Métodos com o mesmo nome podem ter implementações diferentes

```
1 public class Gato extends Animal {  
2     public Gato(String nome) {  
3         super(nome);  
4     }  
5  
6     @Override  
7     public void fazerSom() {  
8         System.out.println("Miau");  
9     }  
10 }  
11  
12 public class Main {  
13     public static void main(String[] args) {  
14         Animal cachorro = new Cachorro("Rex");  
15         Animal gato = new Gato("Mimi");  
16  
17         cachorro.fazerSom();  
18         gato.fazerSom();  
19     }  
20 }
```

- Sobrecarga permite definir múltiplos métodos com o mesmo nome, mas diferentes assinaturas
- A assinatura do método inclui o nome do método e os parâmetros
- Facilita a utilização de métodos com funcionalidades semelhantes
- Está associada ao conceito de Polimorfismo

```
1 public class Calculadora {
2     // Método que soma dois inteiros
3     public int somar(int a, int b) {
4         return a + b;
5     }
6
7     // Método que soma três inteiros
8     public int somar(int a, int b, int c) {
9         return a + b + c;
10    }
11
12    // Método que soma dois números de ponto
13    // flutuante
14    public double somar(double a, double b) {
15        return a + b;
16    }
17
18    // Método que concatena duas strings
19    public String somar(String a, String b) {
20        return a + b;
21    }
22 }
23 public class Main {
24     public static void main(String[] args) {
25         Calculadora calc = new Calculadora();
26         System.out.println(calc.somar(5, 3));
27         System.out.println(calc.somar(1, 2, 3));
28         System.out.println(calc.somar(2.5, 3.5));
29         System.out.println(calc.somar("Olá", "
30         Mundo"));
31     }
32 }
```


- Simplificação da complexidade através de interfaces
- Foco no que um objeto faz, não como faz

```
1  abstract class Forma {  
2      abstract double area();  
3  }  
4  
5  public class Retangulo extends Forma {  
6      private double largura;  
7      private double altura;  
8  
9      public Retangulo(double largura, double  
10         altura) {  
11         this.largura = largura;  
12         this.altura = altura;  
13     }  
14     @Override  
15     double area() {  
16         return largura * altura;  
17     }  
18 }
```

- 1 Resumo
- 2 Modificadores de Acesso**
- 3 Herança
- 4 Polimorfismo
- 5 Polimorfismo e Interface
- 6 QUESTÕES
- 7 SQL

Modificadores de acesso controlam a visibilidade de classes, métodos e variáveis dentro de um programa Java.

Modificadores de acesso controlam a visibilidade de classes, métodos e variáveis dentro de um programa Java.

- **public**: Visível globalmente.

Modificadores de acesso controlam a visibilidade de classes, métodos e variáveis dentro de um programa Java.

- **public**: Visível globalmente.
- **private**: Visível apenas dentro da classe.

Modificadores de acesso controlam a visibilidade de classes, métodos e variáveis dentro de um programa Java.

- **public**: Visível globalmente.
- **private**: Visível apenas dentro da classe.
- **protected**: Visível dentro do pacote e por todas as subclasses.

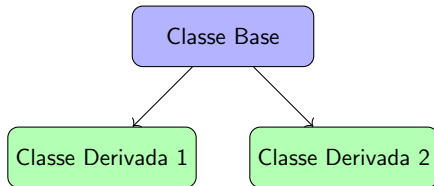
Modificadores de acesso controlam a visibilidade de classes, métodos e variáveis dentro de um programa Java.

- **public**: Visível globalmente.
- **private**: Visível apenas dentro da classe.
- **protected**: Visível dentro do pacote e por todas as subclasses.
- **default**: Visível apenas dentro do pacote.

- 1 Resumo
- 2 Modificadores de Acesso
- 3 Herança**
- 4 Polimorfismo
- 5 Polimorfismo e Interface
- 6 QUESTÕES
- 7 SQL

O que é Herança?

Herança é um princípio da programação orientada a objetos que permite que uma classe derive propriedades e métodos de outra, facilitando a reutilização de código e a criação de uma estrutura hierárquica entre classes.



Em Java, a herança permite que uma classe (*subclasse*) **herde** atributos e métodos de outra (*superclasse*), usando a palavra-chave `extends`.

```
1 class Veiculo {
2     protected int velocidade; // Atributo herdado
3
4     public void mostrarVelocidade() {
5         System.out.println("Velocidade: " + velocidade + " km/h");
6     }
7 }
8
9 class Carro extends Veiculo {
10     private int numeroDePortas; // Atributo específico
11
12     public void mostrarDetalhes() {
13         mostrarVelocidade(); // Uso de método herdado
14         System.out.println("Portas: " + numeroDePortas);
15     }
16 }
```

Ponto-chave

Carro é uma *subclasse* de *Veiculo*, acessando seus métodos e atributos, exemplificando a herança.

Vamos ver um exemplo prático de herança em Java, demonstrando a relação entre uma classe base e suas subclasses.

```
1 class Animal {  
2     protected String som;  
3  
4     public void emitirSom() {  
5         System.out.println("Este animal faz " + som);  
6     }  
7 }  
8  
9 class Cachorro extends Animal {  
10     public Cachorro() {  
11         som = "au au";  
12     }  
13 }  
14  
15 class Gato extends Animal {  
16     public Gato() {  
17         som = "miau";  
18     }  
19 }
```

Aplicação da Herança

A classe 'Animal' serve como base, enquanto 'Cachorro' e 'Gato' são subclasses que herdam e especializam o comportamento de 'Animal'.


Explorando a **sobreposição de métodos** e o uso de `super` na herança em Java.

```
1  class Animal {
2      String som = "algum som";
3
4      public void emitirSom() {
5          System.out.println("Som do animal: " + som);
6      }
7  }
8
9  class Cachorro extends Animal {
10     public Cachorro() {
11         som = "au au";
12     }
13
14     @Override
15     public void emitirSom() {
16         super.emitirSom(); // Chama método da superclasse
17         System.out.println("O cachorro faz: " + som);
18     }
19 }
```

Explorando a **sobreposição de métodos** e o uso de `super` na herança em Java.

```
1 class Animal {  
2     String som = "algum som";  
3  
4     public void emitirSom() {  
5         System.out.println("Som do animal: " + som);  
6     }  
7 }  
8  
9 class Cachorro extends Animal {  
10     public Cachorro() {  
11         som = "au au";  
12     }  
13  
14     @Override  
15     public void emitirSom() {  
16         super.emitirSom(); // Chama método da superclasse  
17         System.out.println("O cachorro faz: " + som);  
18     }  
19 }
```

Chamada do método
da superclasse (`super`)



Explorando a **sobreposição de métodos** e o uso de `super` na herança em Java.

```
1 class Animal {
2     String som = "algum som";
3
4     public void emitirSom() {
5         System.out.println("Som do animal: " + som);
6     }
7 }
8
9 class Cachorro extends Animal {
10     public Cachorro() {
11         som = "au au";
12     }
13
14     @Override
15     public void emitirSom() {
16         super.emitirSom(); // Chama método da superclasse
17         System.out.println("O cachorro faz: " + som);
18     }
19 }
```

Chamada do método da superclasse (super)

Sobreposição do método na subclasse

Explorando a **sobreposição de métodos** e o uso de `super` na herança em Java.

```
1 class Animal {
2     String som = "algum som";
3
4     public void emitirSom() {
5         System.out.println("Som do animal: " + som);
6     }
7 }
8
9 class Cachorro extends Animal {
10     public Cachorro() {
11         som = "au au";
12     }
13
14     @Override
15     public void emitirSom() {
16         super.emitirSom(); // Chama método da superclasse
17         System.out.println("O cachorro faz: " + som);
18     }
19 }
```

Chamada do método da superclasse (super)

Sobreposição do método na subclasse

- **super**: Permite acessar e usar métodos e atributos da superclasse.

Explorando a **sobreposição de métodos** e o uso de `super` na herança em Java.

```
1 class Animal {
2     String som = "algum som";
3
4     public void emitirSom() {
5         System.out.println("Som do animal: " + som);
6     }
7 }
8
9 class Cachorro extends Animal {
10     public Cachorro() {
11         som = "au au";
12     }
13
14     @Override
15     public void emitirSom() {
16         super.emitirSom(); // Chama método da superclasse
17         System.out.println("O cachorro faz: " + som);
18     }
19 }
```

Chamada do método da superclasse (super)

Sobreposição do método na subclasse

- **super:** Permite acessar e usar métodos e atributos da superclasse.
- **Sobreposição de Métodos:** Subclasse modifica comportamento dos métodos herdados.

- 1 Resumo
- 2 Modificadores de Acesso
- 3 Herança
- 4 Polimorfismo**
- 5 Polimorfismo e Interface
- 6 QUESTÕES
- 7 SQL

O **polimorfismo** é um conceito fundamental em Java que permite que objetos de diferentes classes sejam tratados como objetos de uma classe comum. Isso é possível através da herança e da implementação de interfaces.

O **polimorfismo** é um conceito fundamental em Java que permite que objetos de diferentes classes sejam tratados como objetos de uma classe comum. Isso é possível através da herança e da implementação de interfaces.

- Permite que um método tenha várias formas.

O **polimorfismo** é um conceito fundamental em Java que permite que objetos de diferentes classes sejam tratados como objetos de uma classe comum. Isso é possível através da herança e da implementação de interfaces.

- Permite que um método tenha várias formas.
- Facilita a reutilização de código e aumenta a flexibilidade do programa.

O **polimorfismo** é um conceito fundamental em Java que permite que objetos de diferentes classes sejam tratados como objetos de uma classe comum. Isso é possível através da herança e da implementação de interfaces.

- Permite que um método tenha várias formas.
- Facilita a reutilização de código e aumenta a flexibilidade do programa.
- Em Java, o polimorfismo pode ser alcançado por meio da sobreposição de métodos e do uso de interfaces.

```
1 class Animal {
2     public void emitirSom() {
3         System.out.println("Este animal emite um som genérico");
4     }
5 }
6
7 class Cachorro extends Animal {
8     @Override
9     public void emitirSom() {
10         System.out.println("O cachorro faz: au au");
11     }
12 }
13
14 class Gato extends Animal {
15     @Override
16     public void emitirSom() {
17         System.out.println("O gato faz: miau");
18     }
19 }
20
21 public class TestePolimorfismo {
22     public static void main(String[] args) {
23         Animal meuAnimal = new Cachorro();
24         meuAnimal.emitirSom();
25
26         meuAnimal = new Gato();
27         meuAnimal.emitirSom();
28     }
29 }
```

```
1 class Animal {
2     public void emitirSom() {
3         System.out.println("Este animal emite um som genérico");
4     }
5 }
6
7 class Cachorro extends Animal {
8     @Override
9     public void emitirSom() {
10         System.out.println("O cachorro faz: au au");
11     }
12 }
13
14 class Gato extends Animal {
15     @Override
16     public void emitirSom() {
17         System.out.println("O gato faz: miau");
18     }
19 }
20
21 public class TestePolimorfismo {
22     public static void main(String[] args) {
23         Animal meuAnimal = new Cachorro();
24         meuAnimal.emitirSom();
25
26         meuAnimal = new Gato();
27         meuAnimal.emitirSom();
28     }
29 }
```

Sobreposição em Cachorro

```
1 class Animal {
2     public void emitirSom() {
3         System.out.println("Este animal emite um som genérico");
4     }
5 }
6
7 class Cachorro extends Animal {
8     @Override
9     public void emitirSom() {
10         System.out.println("O cachorro faz: au au");
11     }
12 }
13
14 class Gato extends Animal {
15     @Override
16     public void emitirSom() {
17         System.out.println("O gato faz: miau");
18     }
19 }
20
21 public class TestePolimorfismo {
22     public static void main(String[] args) {
23         Animal meuAnimal = new Cachorro();
24         meuAnimal.emitirSom();
25
26         meuAnimal = new Gato();
27         meuAnimal.emitirSom();
28     }
29 }
```

Sobreposição em Cachorro

Sobreposição em Gato


```
1 class Animal {
2     public void emitirSom() {
3         System.out.println("Este animal emite um som genérico");
4     }
5 }
6
7 class Cachorro extends Animal {
8     @Override
9     public void emitirSom() {
10         System.out.println("O cachorro faz: au au");
11     }
12 }
13
14 class Gato extends Animal {
15     @Override
16     public void emitirSom() {
17         System.out.println("O gato faz: miau");
18     }
19 }
20
21 public class TestePolimorfismo {
22     public static void main(String[] args) {
23         Animal meuAnimal = new Cachorro();
24         meuAnimal.emitirSom();
25
26         meuAnimal = new Gato();
27         meuAnimal.emitirSom();
28     }
29 }
```

Sobreposição em Cachorro

Sobreposição em Gato

Polimorfismo com Cachorro

```
1 class Animal {
2     public void emitirSom() {
3         System.out.println("Este animal emite um som genérico");
4     }
5 }
6
7 class Cachorro extends Animal {
8     @Override
9     public void emitirSom() {
10         System.out.println("O cachorro faz: au au");
11     }
12 }
13
14 class Gato extends Animal {
15     @Override
16     public void emitirSom() {
17         System.out.println("O gato faz: miau");
18     }
19 }
20
21 public class TestePolimorfismo {
22     public static void main(String[] args) {
23         Animal meuAnimal = new Cachorro();
24         meuAnimal.emitirSom();
25
26         meuAnimal = new Gato();
27         meuAnimal.emitirSom();
28     }
29 }
```

Sobreposição em Cachorro

Sobreposição em Gato

Polimorfismo com Cachorro

Polimorfismo com Gato

- 1 Resumo
- 2 Modificadores de Acesso
- 3 Herança
- 4 Polimorfismo
- 5 Polimorfismo e Interface**
- 6 QUESTÕES
- 7 SQL

Interfaces em Java definem um contrato que as classes implementadoras devem seguir, estabelecendo um padrão de métodos sem fornecer implementações concretas.

Interfaces em Java definem um contrato que as classes implementadoras devem seguir, estabelecendo um padrão de métodos sem fornecer implementações concretas.

Características das Interfaces

Interfaces em Java definem um contrato que as classes implementadoras devem seguir, estabelecendo um padrão de métodos sem fornecer implementações concretas.

Características das Interfaces

- Definem métodos sem implementações (apenas assinaturas).

Interfaces em Java definem um contrato que as classes implementadoras devem seguir, estabelecendo um padrão de métodos sem fornecer implementações concretas.

Características das Interfaces

- Definem métodos sem implementações (apenas assinaturas).
- Permitem que uma classe implemente múltiplas interfaces.

Interfaces em Java definem um contrato que as classes implementadoras devem seguir, estabelecendo um padrão de métodos sem fornecer implementações concretas.

Características das Interfaces

- Definem métodos sem implementações (apenas assinaturas).
- Permitem que uma classe implemente múltiplas interfaces.
- Facilitam a separação entre o que fazer e como fazer.

Interfaces em Java definem um contrato que as classes implementadoras devem seguir, estabelecendo um padrão de métodos sem fornecer implementações concretas.

Características das Interfaces

- Definem métodos sem implementações (apenas assinaturas).
- Permitem que uma classe implemente múltiplas interfaces.
- Facilitam a separação entre o que fazer e como fazer.
- São usadas para estabelecer um padrão comum entre classes.

Interfaces em Java definem um contrato que as classes implementadoras devem seguir, estabelecendo um padrão de métodos sem fornecer implementações concretas.

Características das Interfaces

- Definem métodos sem implementações (apenas assinaturas).
- Permitem que uma classe implemente múltiplas interfaces.
- Facilitam a separação entre o que fazer e como fazer.
- São usadas para estabelecer um padrão comum entre classes.

Exemplo de Interface

```
1 interface Comunicavel {  
2     void comunicar();  
3 }
```

Interfaces em Java definem um contrato que as classes implementadoras devem seguir, estabelecendo um padrão de métodos sem fornecer implementações concretas.

Características das Interfaces

- Definem métodos sem implementações (apenas assinaturas).
- Permitem que uma classe implemente múltiplas interfaces.
- Facilitam a separação entre o que fazer e como fazer.
- São usadas para estabelecer um padrão comum entre classes.

Exemplo de Interface

```
1 interface Comunicavel {  
2     void comunicar();  
3 }
```

Interfaces são cruciais na programação orientada a objetos em Java, promovendo modularidade, reutilização e manutenção do código.

Interfaces em Java são contratos que especificam um conjunto de métodos que as classes implementadoras devem fornecer.

Interfaces em Java são contratos que especificam um conjunto de métodos que as classes implementadoras devem fornecer.

Sintaxe Básica

```
1 interface NomeDaInterface {  
2     // métodos abstratos  
3     void metodoUm();  
4     int metodoDois(String parametro);  
5 }
```

Interfaces em Java são contratos que especificam um conjunto de métodos que as classes implementadoras devem fornecer.

Sintaxe Básica

```
1 interface NomeDaInterface {  
2     // métodos abstratos  
3     void metodoUm();  
4     int  metodoDois(String parametro);  
5 }
```

Características

Interfaces podem conter:

- Métodos abstratos (sem corpo)
- Métodos default (com corpo)
- Constantes (public static final)

Classes em Java implementam interfaces para fornecer a funcionalidade especificada pelo contrato da interface.

Classes em Java implementam interfaces para fornecer a funcionalidade especificada pelo contrato da interface.

Exemplo de Implementação

```
1 interface Comunicavel {  
2     void comunicar();  
3 }  
4  
5 class Humano implements Comunicavel {  
6     @Override  
7     public void comunicar() {  
8         System.out.println("Humano comunica.");  
9     }  
10 }
```


Classes em Java implementam interfaces para fornecer a funcionalidade especificada pelo contrato da interface.

Exemplo de Implementação

```
1 interface Comunicavel {  
2     void comunicar();  
3 }  
4  
5 class Humano implements Comunicavel {  
6     @Override  
7     public void comunicar() {  
8         System.out.println("Humano comunica.");  
9     }  
10 }
```

Regras

Quando uma classe implementa uma interface, ela deve fornecer implementações para todos os métodos abstratos da interface.

Java 8 introduziu métodos default em interfaces, permitindo que interfaces tenham métodos com implementação.

Java 8 introduziu métodos default em interfaces, permitindo que interfaces tenham métodos com implementação.

Exemplo de Método Default

```
1 interface Veiculo {  
2     default void mover() {  
3         System.out.println("Veículo em movimento");  
4     }  
5 }
```

Java 8 introduziu métodos default em interfaces, permitindo que interfaces tenham métodos com implementação.

Exemplo de Método Default

```
1 interface Veiculo {  
2     default void mover() {  
3         System.out.println("Veículo em movimento");  
4     }  
5 }
```

Utilidade

Métodos default permitem a adição de novas funcionalidades às interfaces sem afetar as classes que as implementam.

```
1 interface Comunicavel {
2     void comunicar();
3 }
4
5 class Humano implements Comunicavel {
6     @Override
7     public void comunicar() {
8         System.out.println("O humano fala.");
9     }
10 }
11
12 class Cachorro implements Comunicavel {
13     @Override
14     public void comunicar() {
15         System.out.println("O cachorro late.");
16     }
17 }
18
19 public class TestePolimorfismo {
20     public static void main(String[] args) {
21         Comunicavel c = new Humano();
22         c.comunicar();
23
24         c = new Cachorro();
25         c.comunicar();
26     }
27 }
```

Exemplos de Polimorfismo em Java

Método da interface

```
1 interface Comunicavel {
2     void comunicar();
3 }
4
5 class Humano implements Comunicavel {
6     @Override
7     public void comunicar() {
8         System.out.println("O humano fala.");
9     }
10 }
11
12 class Cachorro implements Comunicavel {
13     @Override
14     public void comunicar() {
15         System.out.println("O cachorro late.");
16     }
17 }
18
19 public class TestePolimorfismo {
20     public static void main(String[] args) {
21         Comunicavel c = new Humano();
22         c.comunicar();
23
24         c = new Cachorro();
25         c.comunicar();
26     }
27 }
```

Exemplos de Polimorfismo em Java

Método da interface

```
1 interface Comunicavel {  
2     void comunicar();  
3 }  
4  
5 class Humano implements Comunicavel {  
6     @Override  
7     public void comunicar() {  
8         System.out.println("O humano fala.");  
9     }  
10 }  
11  
12 class Cachorro implements Comunicavel {  
13     @Override  
14     public void comunicar() {  
15         System.out.println("O cachorro late.");  
16     }  
17 }  
18  
19 public class TestePolimorfismo {  
20     public static void main(String[] args) {  
21         Comunicavel c = new Humano();  
22         c.comunicar();  
23  
24         c = new Cachorro();  
25         c.comunicar();  
26     }  
27 }
```

Implementação em Humano

Exemplos de Polimorfismo em Java

Método da interface

```
1 interface Comunicavel {  
2     void comunicar();  
3 }  
4  
5 class Humano implements Comunicavel {  
6     @Override  
7     public void comunicar() {  
8         System.out.println("O humano fala.");  
9     }  
10 }  
11  
12 class Cachorro implements Comunicavel {  
13     @Override  
14     public void comunicar() {  
15         System.out.println("O cachorro late.");  
16     }  
17 }  
18  
19 public class TestePolimorfismo {  
20     public static void main(String[] args) {  
21         Comunicavel c = new Humano();  
22         c.comunicar();  
23  
24         c = new Cachorro();  
25         c.comunicar();  
26     }  
27 }
```

Implementação em Humano

Implementação em Cachorro

Exemplos de Polimorfismo em Java

Método da interface

```
1 interface Comunicavel {  
2     void comunicar();  
3 }  
4  
5 class Humano implements Comunicavel {  
6     @Override  
7     public void comunicar() {  
8         System.out.println("O humano fala.");  
9     }  
10 }  
11  
12 class Cachorro implements Comunicavel {  
13     @Override  
14     public void comunicar() {  
15         System.out.println("O cachorro late.");  
16     }  
17 }  
18  
19 public class TestePolimorfismo {  
20     public static void main(String[] args) {  
21         Comunicavel c = new Humano();  
22         c.comunicar();  
23  
24         c = new Cachorro();  
25         c.comunicar();  
26     }  
27 }
```

Implementação em Humano

Implementação em Cachorro

Polimorfismo com Humano

Exemplos de Polimorfismo em Java

Método da interface

```
1 interface Comunicavel {  
2     void comunicar();  
3 }  
4  
5 class Humano implements Comunicavel {  
6     @Override  
7     public void comunicar() {  
8         System.out.println("O humano fala.");  
9     }  
10 }  
11  
12 class Cachorro implements Comunicavel {  
13     @Override  
14     public void comunicar() {  
15         System.out.println("O cachorro late.");  
16     }  
17 }  
18  
19 public class TestePolimorfismo {  
20     public static void main(String[] args) {  
21         Comunicavel c = new Humano();  
22         c.comunicar();  
23  
24         c = new Cachorro();  
25         c.comunicar();  
26     }  
27 }
```

Implementação em Humano

Implementação em Cachorro

Polimorfismo com Humano

Polimorfismo com Cachorro

- 1 Resumo
- 2 Modificadores de Acesso
- 3 Herança
- 4 Polimorfismo
- 5 Polimorfismo e Interface
- 6 QUESTÕES**
- 7 SQL

Considere a seguinte classe em Java:

```
1 public class ContaBancaria {  
2     private double saldo;  
3  
4     public ContaBancaria() {  
5         this.saldo = 0.0;  
6     }  
7  
8     public void depositar(double valor) {  
9         this.saldo += valor;  
10    }  
11  
12    public void sacar(double valor) {  
13        if (valor <= this.saldo) {  
14            this.saldo -= valor;  
15        }  
16    }  
17  
18    public double getSaldo() {  
19        return this.saldo;  
20    }  
21 }
```

```
1 ContaBancaria conta = new ContaBancaria();  
2 conta.depositar(100);  
3 conta.sacar(30);  
4 conta.depositar(20);  
5 conta.sacar(50);  
6 System.out.println(conta.getSaldo());
```

Após a execução do código acima, qual será o saldo da conta?

- a) 0.0
- b) 20.0
- c) 40.0
- d) 50.0
- e) 70.0

Considere a seguinte classe em Java:

```
1 public class ContaBancaria {
2     private double saldo;
3
4     public ContaBancaria() {
5         this.saldo = 0.0;
6     }
7
8     public void depositar(double valor) {
9         this.saldo += valor;
10    }
11
12    public void sacar(double valor) {
13        if (valor <= this.saldo) {
14            this.saldo -= valor;
15        }
16    }
17
18    public double getSaldo() {
19        return this.saldo;
20    }
21 }
```

```
1 ContaBancaria conta = new ContaBancaria();
2 conta.depositar(100);
3 conta.sacar(30);
4 conta.depositar(20);
5 conta.sacar(50);
6 System.out.println(conta.getSaldo());
```

Após a execução do código acima, qual será o saldo da conta?

- a) 0.0
- b) 20.0
- c) **40.0**
- d) 50.0
- e) 70.0

Considere a seguinte classe em Java:

```
1 public class Livro {
2     private int paginasLidas;
3
4     public Livro() {
5         this.paginasLidas = 0;
6     }
7
8     public void lerPaginas(int paginas) {
9         this.paginasLidas += paginas;
10    }
11
12    public void relerPaginas(int paginas) {
13        if (paginas <= this.paginasLidas) {
14            this.paginasLidas -= paginas;
15        }
16    }
17
18    public int getPaginasLidas() {
19        return this.paginasLidas;
20    }
21 }
```

```
1 Livro livro = new Livro();
2 livro.lerPaginas(80);
3 livro.relerPaginas(30);
4 livro.lerPaginas(40);
5 livro.relerPaginas(20);
6 System.out.println(livro.getPaginasLidas());
```

Após a execução do código acima, quantas páginas foram lidas?

- a) 30
- b) 50
- c) 70
- d) 90
- e) 110

Considere a seguinte classe em Java:

```
1 public class Livro {  
2     private int paginasLidas;  
3  
4     public Livro() {  
5         this.paginasLidas = 0;  
6     }  
7  
8     public void lerPaginas(int paginas) {  
9         this.paginasLidas += paginas;  
10    }  
11  
12    public void relerPaginas(int paginas) {  
13        if (paginas <= this.paginasLidas) {  
14            this.paginasLidas -= paginas;  
15        }  
16    }  
17  
18    public int getPaginasLidas() {  
19        return this.paginasLidas;  
20    }  
21 }
```

```
1 Livro livro = new Livro();  
2 livro.lerPaginas(80);  
3 livro.relerPaginas(30);  
4 livro.lerPaginas(40);  
5 livro.relerPaginas(20);  
6 System.out.println(livro.getPaginasLidas());
```

Após a execução do código acima, quantas páginas foram lidas?

- a) 30
- b) 50
- c) **70**
- d) 90
- e) 110

Considere a seguinte classe em Java:

```
1 public class Carro {  
2     private int combustivel;  
3  
4     public Carro() {  
5         this.combustivel = 0;  
6     }  
7  
8     public void abastecer(int litros) {  
9         this.combustivel += litros;  
10    }  
11  
12    public void consumir(int litros) {  
13        if (litros <= this.combustivel) {  
14            this.combustivel -= litros;  
15        }  
16    }  
17  
18    public int getCombustivel() {  
19        return this.combustivel;  
20    }  
21 }
```

```
1 Carro carro = new Carro();  
2 carro.abastecer(40);  
3 carro.consumir(10);  
4 carro.abastecer(20);  
5 carro.consumir(30);  
6 System.out.println(carro.getCombustivel());
```

Após a execução do código acima, quantos litros de combustível restam?

- a) 0
- b) 10
- c) 20
- d) 30
- e) 40

Considere a seguinte classe em Java:

```
1 public class Carro {  
2     private int combustivel;  
3  
4     public Carro() {  
5         this.combustivel = 0;  
6     }  
7  
8     public void abastecer(int litros) {  
9         this.combustivel += litros;  
10    }  
11  
12    public void consumir(int litros) {  
13        if (litros <= this.combustivel) {  
14            this.combustivel -= litros;  
15        }  
16    }  
17  
18    public int getCombustivel() {  
19        return this.combustivel;  
20    }  
21 }
```

```
1 Carro carro = new Carro();  
2 carro.abastecer(40);  
3 carro.consumir(10);  
4 carro.abastecer(20);  
5 carro.consumir(30);  
6 System.out.println(carro.getCombustivel());
```

Após a execução do código acima, quantos litros de combustível restam?

- a) 0
- b) 10
- c) 20
- d) 30
- e) 40

Considere a seguinte classe em Java:

```
1 public class Caixa {  
2     private double dinheiro;  
3  
4     public Caixa() {  
5         this.dinheiro = 0.0;  
6     }  
7  
8     public void adicionarDinheiro(double valor) {  
9         this.dinheiro += valor;  
10    }  
11  
12    public void retirarDinheiro(double valor) {  
13        if (valor <= this.dinheiro) {  
14            this.dinheiro -= valor;  
15        }  
16    }  
17  
18    public double getDinheiro() {  
19        return this.dinheiro;  
20    }  
21 }
```

Após a execução do código acima, qual será o saldo de dinheiro na caixa?

- a) 30.0
- b) 50.0
- c) 80.0
- d) 100.0
- e) 150.0

```
1 Caixa caixa = new Caixa();  
2 caixa.adicionarDinheiro(200);  
3 caixa.retirarDinheiro(70);  
4 caixa.adicionarDinheiro(50);  
5 caixa.retirarDinheiro(100);  
6 System.out.println(caixa.getDinheiro());
```

Considere a seguinte classe em Java:

```
1 public class Caixa {
2     private double dinheiro;
3
4     public Caixa() {
5         this.dinheiro = 0.0;
6     }
7
8     public void adicionarDinheiro(double valor) {
9         this.dinheiro += valor;
10    }
11
12    public void retirarDinheiro(double valor) {
13        if (valor <= this.dinheiro) {
14            this.dinheiro -= valor;
15        }
16    }
17
18    public double getDinheiro() {
19        return this.dinheiro;
20    }
21 }
```

```
1 Caixa caixa = new Caixa();
2 caixa.adicionarDinheiro(200);
3 caixa.retirarDinheiro(70);
4 caixa.adicionarDinheiro(50);
5 caixa.retirarDinheiro(100);
6 System.out.println(caixa.getDinheiro());
```

Após a execução do código acima, qual será o saldo de dinheiro na caixa?

- a) 30.0
- b) 50.0
- c) **80.0**
- d) 100.0
- e) 150.0

Considere a seguinte classe em Java:

```
1 public class Estoque {  
2     private int quantidade;  
3  
4     public Estoque() {  
5         this.quantidade = 0;  
6     }  
7  
8     public void adicionar(int qtd) {  
9         this.quantidade += qtd;  
10    }  
11  
12    public void remover(int qtd) {  
13        if (qtd <= this.quantidade) {  
14            this.quantidade -= qtd;  
15        }  
16    }  
17  
18    public int getQuantidade() {  
19        return this.quantidade;  
20    }  
21 }
```

```
1 Estoque estoque = new Estoque();  
2 estoque.adicionar(150);  
3 estoque.remover(60);  
4 estoque.adicionar(30);  
5 estoque.remover(40);  
6 System.out.println(estoque.getQuantidade());
```

Após a execução do código acima, qual será a quantidade restante no estoque?

- a) 20
- b) 30
- c) 50
- d) 70
- e) 80

Considere a seguinte classe em Java:

```
1 public class Estoque {  
2     private int quantidade;  
3  
4     public Estoque() {  
5         this.quantidade = 0;  
6     }  
7  
8     public void adicionar(int qtd) {  
9         this.quantidade += qtd;  
10    }  
11  
12    public void remover(int qtd) {  
13        if (qtd <= this.quantidade) {  
14            this.quantidade -= qtd;  
15        }  
16    }  
17  
18    public int getQuantidade() {  
19        return this.quantidade;  
20    }  
21 }
```

```
1 Estoque estoque = new Estoque();  
2 estoque.adicionar(150);  
3 estoque.remover(60);  
4 estoque.adicionar(30);  
5 estoque.remover(40);  
6 System.out.println(estoque.getQuantidade());
```

Após a execução do código acima, qual será a quantidade restante no estoque?

- a) 20
- b) 30
- c) 50
- d) 70
- e) **80**

Sobre os conceitos de herança em POO, classifique V para as sentenças verdadeiras e F para as falsas.

- ☐ A herança é um mecanismo que permite que uma classe adquira os atributos e métodos de outra classe.
- ☐ Uma classe filha pode sobrescrever métodos da classe pai.
- ☐ A herança múltipla é permitida em Java e a palavra-chave que deve ser usada é *implements*.
- ☐ A palavra-chave para indicar herança em Java é *extends*.

Assinale a alternativa que apresenta a sequência CORRETA:

- a) F - V - V - F.
- b) V - F - F - V.
- c) V - V - V - F.
- d) V - V - F - V.
- e) V - F - V - V.

Sobre os conceitos de herança em POO, classifique V para as sentenças verdadeiras e F para as falsas.

- ☐ A herança é um mecanismo que permite que uma classe adquira os atributos e métodos de outra classe.
- ☐ Uma classe filha pode sobrescrever métodos da classe pai.
- ☐ A herança múltipla é permitida em Java e a palavra-chave que deve ser usada é *implements*.
- ☐ A palavra-chave para indicar herança em Java é *extends*.

Assinale a alternativa que apresenta a sequência CORRETA:

- a) F - V - V - F.
- b) V - F - F - V.
- c) V - V - V - F.
- d) **V - V - F - V.**
- e) V - F - V - V.

Sobre polimorfismo em POO, classifique V para as sentenças verdadeiras e F para as falsas.

- ☐ Polimorfismo permite que objetos de diferentes classes sejam tratados como objetos de uma classe comum.
- ☐ Em Java, o polimorfismo é alcançado apenas através de interfaces.
- ☐ Polimorfismo é uma característica que permite a reescrita de métodos na classe filha.
- ☐ Em Java, o polimorfismo é alcançado apenas através de Herança.

Assinale a alternativa que apresenta a sequência CORRETA:

- a) V - F - V - V.
- b) F - V - F - V.
- c) V - F - F - V.
- d) F - F - V - F.
- e) V - F - V - F.

Sobre polimorfismo em POO, classifique V para as sentenças verdadeiras e F para as falsas.

- ☐ Polimorfismo permite que objetos de diferentes classes sejam tratados como objetos de uma classe comum.
- ☐ Em Java, o polimorfismo é alcançado apenas através de interfaces.
- ☐ Polimorfismo é uma característica que permite a reescrita de métodos na classe filha.
- ☐ Em Java, o polimorfismo é alcançado apenas através de Herança.

Assinale a alternativa que apresenta a sequência CORRETA:

- a) V - F - V - V.
- b) F - V - F - V.
- c) V - F - F - V.
- d) F - F - V - F.
- e) **V - F - V - F.**

Questão 8

Sobre encapsulamento em POO, classifique V para as sentenças verdadeiras e F para as falsas.

- ☐ O encapsulamento é o processo de ocultar a implementação dos detalhes internos de um objeto, expondo apenas o necessário.
- ☐ Em Java, o encapsulamento é alcançado utilizando modificadores de acesso como *private*, *protected* e *public* para controlar a visibilidade dos membros da classe.
- ☐ O encapsulamento impede que dados sensíveis sejam acessados diretamente, permitindo o acesso apenas através de métodos específicos.
- ☐ O encapsulamento permite que dados de uma classe sejam acessados diretamente por qualquer outra classe.

Assinale a alternativa que apresenta a sequência CORRETA:

- a) V - V - V - F.
- b) F - V - F - V.
- c) V - F - F - V.
- d) V - V - F - V.
- e) F - F - V - F.

Questão 8

Sobre encapsulamento em POO, classifique V para as sentenças verdadeiras e F para as falsas.

- ☐ O encapsulamento é o processo de ocultar a implementação dos detalhes internos de um objeto, expondo apenas o necessário.
- ☐ Em Java, o encapsulamento é alcançado utilizando modificadores de acesso como *private*, *protected* e *public* para controlar a visibilidade dos membros da classe.
- ☐ O encapsulamento impede que dados sensíveis sejam acessados diretamente, permitindo o acesso apenas através de métodos específicos.
- ☐ O encapsulamento permite que dados de uma classe sejam acessados diretamente por qualquer outra classe.

Assinale a alternativa que apresenta a sequência CORRETA:

- a) **V - V - V - F.**
- b) F - V - F - V.
- c) V - F - F - V.
- d) V - V - F - V.
- e) F - F - V - F.

Sobre abstração em POO, classifique V para as sentenças verdadeiras e F para as falsas.

- ☐ Abstração é o processo de se concentrar nos aspectos essenciais de uma entidade.
- ☐ Abstração permite que detalhes internos da implementação sejam ocultados.
- ☐ Em Java, a abstração pode ser implementada utilizando classes abstratas e interfaces.
- ☐ Abstração é a mesma coisa que encapsulamento.

Assinale a alternativa que apresenta a sequência CORRETA:

- a) V - V - V - F.
- b) F - V - F - V.
- c) V - F - F - V.
- d) V - V - F - V.
- e) F - F - V - F.

Sobre abstração em POO, classifique V para as sentenças verdadeiras e F para as falsas.

- ☐ Abstração é o processo de se concentrar nos aspectos essenciais de uma entidade.
- ☐ Abstração permite que detalhes internos da implementação sejam ocultados.
- ☐ Em Java, a abstração pode ser implementada utilizando classes abstratas e interfaces.
- ☐ Abstração é a mesma coisa que encapsulamento.

Assinale a alternativa que apresenta a sequência CORRETA:

- a) **V - V - V - F.**
- b) F - V - F - V.
- c) V - F - F - V.
- d) V - V - F - V.
- e) F - F - V - F.

Sobre interfaces em POO, classifique V para as sentenças verdadeiras e F para as falsas.

- ☐ Uma interface em Java pode conter métodos abstratos e métodos padrão (default).
- ☐ Em Java, uma classe pode implementar várias interfaces.
- ☐ Uma interface é usada junto ao conceito de herança e encapsulamento.
- ☐ Uma interface pode ser instanciada diretamente.

Assinale a alternativa que apresenta a sequência CORRETA:

- a) V - V - F - F.
- b) F - V - F - V.
- c) V - F - F - V.
- d) V - V - F - V.
- e) F - F - V - F.

Sobre interfaces em POO, classifique V para as sentenças verdadeiras e F para as falsas.

- ☐ Uma interface em Java pode conter métodos abstratos e métodos padrão (default).
- ☐ Em Java, uma classe pode implementar várias interfaces.
- ☐ Uma interface é usada junto ao conceito de herança e encapsulamento.
- ☐ Uma interface pode ser instanciada diretamente.

Assinale a alternativa que apresenta a sequência CORRETA:

- a) **V - V - F - F.**
- b) F - V - F - V.
- c) V - F - F - V.
- d) V - V - F - V.
- e) F - F - V - F.

Considere o seguinte trecho de código Java, que faz parte de um programa destinado a testar o comportamento de herança e polimorfismo em Java. Analise o código cuidadosamente e determine qual será a saída do programa quando ele for executado.


```
1 class Jogador {  
2     public void atacar() {  
3         System.out.println("Ataque genérico");  
4     }  
5 }  
6  
7 class Guerreiro extends Jogador {  
8     public void atacar() {  
9         System.out.println("Ataque de espada");  
10    }  
11 }  
12  
13 public class Main {  
14     public static void main(String[] args) {  
15         Jogador meuJogador = new Guerreiro();  
16         meuJogador.atacar();  
17     }  
18 }
```

- a) "Ataque genérico"
- b) "Ataque de espada"
- c) Erro de compilação
- d) Sem saída
- e) Exception

```
1 class Jogador {  
2     public void atacar() {  
3         System.out.println("Ataque genérico");  
4     }  
5 }  
6  
7 class Guerreiro extends Jogador {  
8     public void atacar() {  
9         System.out.println("Ataque de espada");  
10    }  
11 }  
12  
13 public class Main {  
14     public static void main(String[] args) {  
15         Jogador meuJogador = new Guerreiro();  
16         meuJogador.atacar();  
17     }  
18 }
```

- a) "Ataque genérico"
- b) "Ataque de espada"**
- c) Erro de compilação
- d) Sem saída
- e) Exception

```
1 class Veiculo {
2     public void mover() {
3         System.out.println("O veículo está se movendo");
4     }
5 }
6
7 class Carro extends Veiculo {
8     public void mover() {
9         System.out.println("O carro está se movendo");
10    }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15         Veiculo meuVeiculo = new Carro();
16         meuVeiculo.mover();
17     }
18 }
```

- a) "O veículo está se movendo"
- b) "O carro está se movendo"
- c) Erro de compilação
- d) Sem saída
- e) Exception

```
1 class Veiculo {
2     public void mover() {
3         System.out.println("O veículo está se movendo");
4     }
5 }
6
7 class Carro extends Veiculo {
8     public void mover() {
9         System.out.println("O carro está se movendo");
10    }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15         Veiculo meuVeiculo = new Carro();
16         meuVeiculo.mover();
17     }
18 }
```

- a) "O veículo está se movendo"
- b) "O carro está se movendo"**
- c) Erro de compilação
- d) Sem saída
- e) Exception

```
1 class Instrumento {
2     public void tocar() {
3         System.out.println("Instrumento tocando");
4     }
5 }
6
7 class Guitarra extends Instrumento {
8     public void tocar() {
9         System.out.println("Guitarra tocando");
10    }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15         Instrumento meuInstrumento = new Guitarra();
16         meuInstrumento.tocar();
17     }
18 }
```

- a) "Instrumento tocando"
- b) "Guitarra tocando"
- c) Erro de compilação
- d) Sem saída
- e) Exception

```
1 class Instrumento {
2     public void tocar() {
3         System.out.println("Instrumento tocando");
4     }
5 }
6
7 class Guitarra extends Instrumento {
8     public void tocar() {
9         System.out.println("Guitarra tocando");
10    }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15         Instrumento meuInstrumento = new Guitarra();
16         meuInstrumento.tocar();
17     }
18 }
```

- a) "Instrumento tocando"
- b) "Guitarra tocando"**
- c) Erro de compilação
- d) Sem saída
- e) Exception

```
1 class Personagem {
2     public void pular() {
3         System.out.println("Pulo genérico");
4     }
5 }
6
7 class Ninja extends Personagem {
8     public void pular() {
9         System.out.println("Pulo mortal");
10    }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15         Personagem meuPersonagem = new Personagem();
16         meuPersonagem.pular();
17     }
18 }
```

- a) "Pulo genérico"
- b) "Pulo mortal"
- c) Erro de compilação
- d) Sem saída
- e) Exception

```
1 class Personagem {
2     public void pular() {
3         System.out.println("Pulo genérico");
4     }
5 }
6
7 class Ninja extends Personagem {
8     public void pular() {
9         System.out.println("Pulo mortal");
10    }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15         Personagem meuPersonagem = new Personagem();
16         meuPersonagem.pular();
17     }
18 }
```

- a) "Pulo genérico"
- b) "Pulo mortal"
- c) Erro de compilação
- d) Sem saída
- e) Exception


```
1 class Personagem {
2     public void atacar() {
3         System.out.println("Ataque simples");
4     }
5 }
6
7 class Mago extends Personagem {
8     public void atacar() {
9         System.out.println("Ataque mágico");
10    }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15         Personagem meuPersonagem = new Personagem();
16         meuPersonagem.atacar();
17     }
18 }
```

- a) "Ataque simples"
- b) "Ataque mágico"
- c) Erro de compilação
- d) Sem saída
- e) Exception

```
1 class Personagem {
2     public void atacar() {
3         System.out.println("Ataque simples");
4     }
5 }
6
7 class Mago extends Personagem {
8     public void atacar() {
9         System.out.println("Ataque mágico");
10    }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15         Personagem meuPersonagem = new Personagem();
16         meuPersonagem.atacar();
17     }
18 }
```

- a) "Ataque simples"
- b) "Ataque mágico"
- c) Erro de compilação
- d) Sem saída
- e) Exception

```
1 class Computador {
2     public void ligar() {
3         System.out.println("Computador ligado");
4     }
5 }
6
7 class Laptop extends Computador {
8     public void ligando() {
9         System.out.println("Laptop ligado");
10    }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15         Computador meuComputador = new Laptop();
16         meuComputador.ligar();
17     }
18 }
```

- a) "Computador ligado"
- b) "Laptop ligado"
- c) Erro de compilação
- d) Sem saída
- e) Exception

```
1 class Computador {
2     public void ligar() {
3         System.out.println("Computador ligado");
4     }
5 }
6
7 class Laptop extends Computador {
8     public void ligando() {
9         System.out.println("Laptop ligado");
10    }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15         Computador meuComputador = new Laptop();
16         meuComputador.ligar();
17     }
18 }
```

- a) **"Computador ligado"**
- b) "Laptop ligado"
- c) Erro de compilação
- d) Sem saída
- e) Exception

Questão 17

```
1 interface Animal {
2     void fazerSom();
3 }
4
5 abstract class Mamifero implements Animal {
6     public void fazerSom() {
7         System.out.println("Som de mamífero");
8     }
9 }
10
11 class Cachorro extends Mamifero {
12     public void fazerSom() {
13         System.out.println("Latido");
14     }
15 }
16
17 public class Main {
18     public static void main(String[] args) {
19         Mamifero meuAnimal = new Mamifero();
20         meuAnimal.fazerSom();
21     }
22 }
```

- a) "Som de mamífero"
- b) "Latido"
- c) Erro de compilação
- d) Sem saída
- e) Exception

```
1 interface Animal {
2     void fazerSom();
3 }
4
5 abstract class Mamifero implements Animal {
6     public void fazerSom() {
7         System.out.println("Som de mamífero");
8     }
9 }
10
11 class Cachorro extends Mamifero {
12     public void fazerSom() {
13         System.out.println("Latido");
14     }
15 }
16
17 public class Main {
18     public static void main(String[] args) {
19         Mamifero meuAnimal = new Mamifero();
20         meuAnimal.fazerSom();
21     }
22 }
```

- a) "Som de mamífero"
- b) "Latido"
- c) **Erro de compilação**
- d) Sem saída
- e) Exception

```
1 interface Desenhavel {
2     void desenhar();
3 }
4
5 abstract class Forma implements Desenhavel {
6     public void desenhar() {
7         System.out.println("Desenhando um círculo");
8     }
9 }
10
11 class Circulo extends Forma {
12     public void desenhar() {
13         System.out.println("Desenhando uma forma");
14     }
15 }
16
17 public class Main {
18     public static void main(String[] args) {
19         Forma minhaForma = new Circulo();
20         minhaForma.desenhar();
21     }
22 }
```

- a) "Desenhando uma forma"
- b) "Desenhando um círculo"
- c) "Desenhando uma forma" seguido de "Desenhando um círculo"
- d) Sem saída

```
1 interface Desenhavel {
2     void desenhar();
3 }
4
5 abstract class Forma implements Desenhavel {
6     public void desenhar() {
7         System.out.println("Desenhando um círculo");
8     }
9 }
10
11 class Circulo extends Forma {
12     public void desenhar() {
13         System.out.println("Desenhando uma forma");
14     }
15 }
16
17 public class Main {
18     public static void main(String[] args) {
19         Forma minhaForma = new Circulo();
20         minhaForma.desenhar();
21     }
22 }
```

a) **"Desenhando uma forma"**

b) "Desenhando um círculo"

c) "Desenhando uma forma" seguido de "Desenhando um círculo"

Considerando o banco de dados *biblioteca* com a tabela *livros* a seguir, qual a saída do código e por quê?

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6
7 public class Biblioteca {
8     public static void main(String[] args) {
9         String url = "jdbc:mysql://localhost:3306/biblioteca";
10        String user = "root";
11        String password = "minhaSenha";
12        try {
13            Connection conn = DriverManager.getConnection(url, user, password);
14            String sql = "SELECT titulo FROM livros WHERE autor_id = 2";
15            PreparedStatement pstmt = conn.prepareStatement(sql);
16            ResultSet rs = pstmt.executeQuery();
17            while (rs.next()) {
18                System.out.println("Titulo: " + rs.getString("titulo"));
19            }
20        } catch (SQLException e) {
21            e.printStackTrace();
22        }
23    }
24 }
```

Considerando o banco de dados *biblioteca* com a tabela *livros* contendo os dados abaixo, qual a saída do código e por quê?

id	titulo	autor_id	ano_publicacao	genero_id
1	O Senhor dos Anéis	1	1954	1
2	1984	2	1949	2
3	O Apanhador no Campo	3	1951	3
4	A Metamorfose	4	1915	4
5	Dom Quixote	5	1605	5

- a) Nenhuma saída, pois nada é impresso pelo programa;
- b) Será impresso apenas o título *1984*;
- c) Serão impressos os títulos *1984* e *Dom Quixote*;
- d) Serão impressos todos os títulos da tabela;
- e) Exception será gerada devido a formatação do SQL.

Considerando o banco de dados *biblioteca* com a tabela *livros* contendo os dados abaixo, qual a saída do código e por quê?

id	titulo	autor_id	ano_publicacao	genero_id
1	O Senhor dos Anéis	1	1954	1
2	1984	2	1949	2
3	O Apanhador no Campo	3	1951	3
4	A Metamorfose	4	1915	4
5	Dom Quixote	5	1605	5

a) Nenhuma saída, pois nada é impresso pelo programa;

b) Será impresso apenas o título 1984;

c) Serão impressos os títulos *1984* e *Dom Quixote*;

d) Serão impressos todos os títulos da tabela;

e) Exception será gerada devido a formatação do SQL.

Considerando o banco de dados *loja* com a tabela *produtos* a seguir, qual a saída do código e por quê?

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6
7 public class Loja {
8     public static void main(String[] args) {
9         String url = "jdbc:mysql://localhost:3306/loja";
10        String user = "root";
11        String password = "minhaSenha";
12        try {
13            Connection conn = DriverManager.getConnection(url, user, password);
14            String sql = "SELECT nome FROM produtos WHERE categoria =
15                          'Eletrônicos'";
16            PreparedStatement pstmt = conn.prepareStatement(sql);
17            ResultSet rs = pstmt.executeQuery();
18            while (rs.next()) {
19                System.out.println("Nome: " + rs.getString("nome"));
20            }
21        } catch (SQLException e) {
22            e.printStackTrace();
23        }
24    }
```

Considerando o banco de dados *loja* com a tabela *produtos* contendo os dados abaixo, qual a saída do código e por quê?

id	nome	preco	categoria
1	Camiseta	30	Roupas
2	Calça	60	Roupas
3	Meias	10	Roupas
4	Tênis	120	Calçados
5	Boné	20	Acessórios

- a) Nenhuma saída, pois nada é impresso pelo programa;
- b) Será impresso apenas o nome *Camiseta*;
- c) Serão impressos os nomes *Camiseta* e *Calça*;
- d) Serão impressos todos os nomes da tabela;
- e) Nenhuma saída, pois a categoria *Eletrônicos* não existe na tabela.

Considerando o banco de dados *loja* com a tabela *produtos* contendo os dados abaixo, qual a saída do código e por quê?

id	nome	preco	categoria
1	Camiseta	30	Roupas
2	Calça	60	Roupas
3	Meias	10	Roupas
4	Tênis	120	Calçados
5	Boné	20	Acessórios

- a) Nenhuma saída, pois nada é impresso pelo programa;
- b) Será impresso apenas o nome *Camiseta*;
- c) Serão impressos os nomes *Camiseta* e *Calça*;
- d) Serão impressos todos os nomes da tabela;
- e) **Nenhuma saída, pois a categoria Eletrônicos não existe na tabela.**

Considerando o banco de dados *loja* com a tabela *produtos* a seguir, qual a saída do código e por quê?

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6
7 public class Loja {
8     public static void main(String[] args) {
9         String url = "jdbc:mysql://localhost:3306/loja";
10        String user = "root";
11        String password = "minhaSenha";
12        try {
13            Connection conn = DriverManager.getConnection(url, user, password);
14            String sql = "SELECT nome FROM produtos WHERE preco < 50";
15            PreparedStatement pstmt = conn.prepareStatement(sql);
16            ResultSet rs = pstmt.executeQuery();
17            while (rs.next()) {
18                System.out.println("Nome: " + rs.getString("nome"));
19            }
20        } catch (SQLException e) {
21            e.printStackTrace();
22        }
23    }
24 }
```

Considerando o banco de dados *loja* com a tabela *produtos* contendo os dados abaixo, qual a saída do código e por quê?

id	nome	preco	categoria
1	Camiseta	30	Roupas
2	Calça	60	Roupas
3	Meias	10	Roupas
4	Tênis	120	Calçados
5	Boné	20	Acessórios

- a) Nenhuma saída, pois nada é impresso pelo programa;
- b) Será impresso apenas o nome *Camiseta*;
- c) Serão impressos os nomes *Camiseta* e *Meias*;
- d) Serão impressos os nomes *Camiseta*, *Meias* e *Boné*;
- e) Exception será gerada devido a formatação do SQL.

Considerando o banco de dados *loja* com a tabela *produtos* contendo os dados abaixo, qual a saída do código e por quê?

id	nome	preco	categoria
1	Camiseta	30	Roupas
2	Calça	60	Roupas
3	Meias	10	Roupas
4	Tênis	120	Calçados
5	Boné	20	Acessórios

a) Nenhuma saída, pois nada é impresso pelo programa;

b) Será impresso apenas o nome *Camiseta*;

c) Serão impressos os nomes *Camiseta* e *Meias*;

d) Serão impressos os nomes *Camiseta*, *Meias* e *Boné*;

e) Exception será gerada devido a formatação do SQL.

Considerando o banco de dados *escola* com a tabela *alunos* a seguir, qual a saída do código e por quê?

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6
7 public class Escola {
8     public static void main(String[] args) {
9         String url = "jdbc:mysql://localhost:3306/escola";
10        String user = "root";
11        String password = "minhaSenha";
12        try {
13            Connection conn = DriverManager.getConnection(url, user, password);
14            String sql = "SELECT nome FROM alunos WHERE idade >= 18";
15            PreparedStatement pstmt = conn.prepareStatement(sql);
16            ResultSet rs = pstmt.executeQuery();
17            while (rs.next()) {
18                System.out.println("Nome: " + rs.getString("nome"));
19            }
20        } catch (SQLException e) {
21            e.printStackTrace();
22        }
23    }
24 }
```

Considerando o banco de dados *escola* com a tabela *alunos* contendo os dados abaixo, qual a saída do código e por quê?

id	nome	idade	turma
1	João	17	A
2	Maria	18	B
3	Pedro	19	C
4	Ana	16	A
5	Lucas	20	D

- a) Nenhuma saída, pois nada é impresso pelo programa;
- b) Será impresso apenas o nome *Maria*;
- c) Serão impressos os nomes *Maria* e *Pedro*;
- d) Serão impressos os nomes *Maria*, *Pedro* e *Lucas*;
- e) Exception será gerada devido a formatação do SQL.

Questão 22 - Tabela de Dados

Considerando o banco de dados *escola* com a tabela *alunos* contendo os dados abaixo, qual a saída do código e por quê?

id	nome	idade	turma
1	João	17	A
2	Maria	18	B
3	Pedro	19	C
4	Ana	16	A
5	Lucas	20	D

a) Nenhuma saída, pois nada é impresso pelo programa;

b) Será impresso apenas o nome *Maria*;

c) Serão impressos os nomes *Maria* e *Pedro*;

d) Serão impressos os nomes *Maria*, *Pedro* e *Lucas*;

e) Exception será gerada devido a formatação do SQL.

Considerando o banco de dados *cinema* com a tabela *filmes* a seguir, qual a saída do código e por quê?

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6
7 public class Cinema {
8     public static void main(String[] args) {
9         String url = "jdbc:mysql://localhost:3306/cinema";
10        String user = "root";
11        String password = "minhaSenha";
12        try {
13            Connection conn = DriverManager.getConnection(url, user, password);
14            String sql = "SELECT titulo FROM filmes WHERE duracao > 170";
15            PreparedStatement pstmt = conn.prepareStatement(sql);
16            ResultSet rs = pstmt.executeQuery();
17            while (rs.next()) {
18                System.out.println("Titulo: " + rs.getString("titulo"));
19            }
20        } catch (SQLException e) {
21            e.printStackTrace();
22        }
23    }
24 }
```

Considerando o banco de dados *cinema* com a tabela *filmes* contendo os dados abaixo, qual a saída do código e por quê?

id	titulo	duracao	genero
1	Vingadores	180	Ação
2	Titanic	195	Romance
3	Toy Story	90	Animação
4	Matrix	136	Ficção
5	A Bela e a Fera	129	Fantasia

- a) Nenhuma saída, pois nada é impresso pelo programa;
- b) Será impresso apenas o título *Vingadores*;
- c) Serão impressos os títulos *Vingadores* e *Titanic*;
- d) Serão impressos os títulos *Vingadores*, *Titanic* e *Matrix*;
- e) Exception será gerada devido a formatação do SQL.

Considerando o banco de dados *cinema* com a tabela *filmes* contendo os dados abaixo, qual a saída do código e por quê?

id	titulo	duracao	genero
1	Vingadores	180	Ação
2	Titanic	195	Romance
3	Toy Story	90	Animação
4	Matrix	136	Ficção
5	A Bela e a Fera	129	Fantasia

a) Nenhuma saída, pois nada é impresso pelo programa;

b) Será impresso apenas o título *Vingadores*;

c) Serão impressos os títulos *Vingadores* e *Titanic*;

d) Serão impressos os títulos *Vingadores*, *Titanic* e *Matrix*;

e) Exception será gerada devido a formatação do SQL.

Considerando o banco de dados *academia* com a tabela *alunos* a seguir, qual a saída do código e por quê?

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6
7 public class Academia {
8     public static void main(String[] args) {
9         String url = "jdbc:mysql://localhost:3306/academia";
10        String user = "root";
11        String password = "minhaSenha";
12        try {
13            Connection conn = DriverManager.getConnection(url, user, password);
14            String sql = "SELECT nome FROM alunos WHERE idade > 27";
15            PreparedStatement pstmt = conn.prepareStatement(sql);
16            ResultSet rs = pstmt.executeQuery();
17            while (rs.next()) {
18                System.out.println("Nome: " + rs.getString("nome"));
19            }
20        } catch (SQLException e) {
21            e.printStackTrace();
22        }
23    }
24 }
```


Considerando o banco de dados *academia* com a tabela *alunos* contendo os dados abaixo, qual a saída do código e por quê?

id	nome	idade	plano
1	João	20	Mensal
2	Maria	30	Anual
3	Pedro	28	Mensal
4	Ana	22	Trimestral
5	Lucas	26	Anual

- a) Nenhuma saída, pois nada é impresso pelo programa;
- b) Será impresso apenas o nome *Maria*;
- c) Serão impressos os nomes *Maria* e *Pedro*;
- d) Serão impressos os nomes *Maria*, *Pedro* e *Lucas*;
- e) Exception será gerada devido a formatação do SQL.

Considerando o banco de dados *academia* com a tabela *alunos* contendo os dados abaixo, qual a saída do código e por quê?

id	nome	idade	plano
1	João	20	Mensal
2	Maria	30	Anual
3	Pedro	28	Mensal
4	Ana	22	Trimestral
5	Lucas	26	Anual

- a) Nenhuma saída, pois nada é impresso pelo programa;
- b) Será impresso apenas o nome *Maria*;
- c) **Serão impressos os nomes *Maria* e *Pedro*;**
- d) Serão impressos os nomes *Maria*, *Pedro* e *Lucas*;
- e) Exception será gerada devido a formatação do SQL.

- 1 Resumo
- 2 Modificadores de Acesso
- 3 Herança
- 4 Polimorfismo
- 5 Polimorfismo e Interface
- 6 QUESTÕES
- 7 SQL**

Atividade SQL

O setor financeiro da empresa precisa de um relatório que mostre o código e o nome dos produtos cujo preço seja menor que 10 ou maior que 100.

Atividade SQL

O setor financeiro da empresa precisa de um relatório que mostre o código e o nome dos produtos cujo preço seja menor que 10 ou maior que 100.

id	name	amount	price
1	Two-door wardrobe	100	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	350	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products

Atividade SQL

O setor financeiro da empresa precisa de um relatório que mostre o código e o nome dos produtos cujo preço seja menor que 10 ou maior que 100.

id	name	amount	price
1	Two-door wardrobe	100	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	350	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products

```
1 SELECT id, name
2 FROM products
3 WHERE price < 10 OR price > 100;
```

Atividade SQL

O setor financeiro da empresa precisa de um relatório que mostre o código e o nome dos produtos cujo preço seja menor que 10 ou maior que 100.

id	name	amount	price
1	Two-door wardrobe	100	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	350	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products

```
1 SELECT id, name
2 FROM products
3 WHERE price < 10 OR price > 100;
```

id	name
2	Dining table
3	Towel holder
5	Chair

Tabela: Exemplo de Saída

Coluna	Tipo
id (PK)	numeric
name	varchar
amount	numeric
price	numeric

Tabela: Esquema da Tabela

Requisito do Relatório

O setor de controle de estoque da empresa precisa de um relatório que mostre o código e o nome dos produtos que estão fora de estoque (quantidade igual a zero).

Requisito do Relatório

O setor de controle de estoque da empresa precisa de um relatório que mostre o código e o nome dos produtos que estão fora de estoque (quantidade igual a zero).

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

Requisito do Relatório

O setor de controle de estoque da empresa precisa de um relatório que mostre o código e o nome dos produtos que estão fora de estoque (quantidade igual a zero).

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```
1 SELECT id, name
2 FROM products2
3 WHERE amount = 0;
```

Requisito do Relatório

O setor de controle de estoque da empresa precisa de um relatório que mostre o código e o nome dos produtos que estão fora de estoque (quantidade igual a zero).

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```
1 SELECT id, name
2 FROM products2
3 WHERE amount = 0;
```

id	name
1	Two-door wardrobe
4	Computer desk

Tabela: Exemplo de Saída

Coluna	Tipo
id (PK)	numeric
name	varchar
amount	numeric
price	numeric

Tabela: Esquema da Tabela products2

Requisito do Relatório

O setor de controle de preços da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm o preço acima de R\$100,00.

Requisito do Relatório

O setor de controle de preços da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm o preço acima de R\$100,00.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

Requisito do Relatório

O setor de controle de preços da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm o preço acima de R\$100,00.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```
1 SELECT id, name
2 FROM products2
3 WHERE price > 100;
```

Requisito do Relatório

O setor de controle de preços da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm o preço acima de R\$100,00.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```
1 SELECT id, name
2 FROM products2
3 WHERE price > 100;
```

id	name
2	Dining table
5	Chair

Tabela: Exemplo de Saída

Coluna	Tipo
id (PK)	numeric
name	varchar
amount	numeric
price	numeric

Tabela: Esquema da Tabela products2

Requisito do Relatório

O setor de controle de estoque da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade menor que 500 unidades.

Requisito do Relatório

O setor de controle de estoque da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade menor que 500 unidades.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

Requisito do Relatório

O setor de controle de estoque da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade menor que 500 unidades.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```
1 SELECT id, name
2 FROM products2
3 WHERE amount < 500;
```

Requisito do Relatório

O setor de controle de estoque da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade menor que 500 unidades.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```

1 SELECT id, name
2 FROM products2
3 WHERE amount < 500;

```

id	name
1	Two-door wardrobe
4	Computer desk

Tabela: Exemplo de Saída

Coluna	Tipo
id (PK)	numeric
name	varchar
amount	numeric
price	numeric

Tabela: Esquema da Tabela products2

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades.

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```
1 SELECT id, name
2 FROM products2
3 WHERE amount > 2000;
```

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```

1 SELECT id, name
2 FROM products2
3 WHERE amount > 2000;
    
```

id	name
3	Towel holder
5	Chair

Tabela: Exemplo de Saída

Coluna	Tipo
id (PK)	numeric
name	varchar
amount	numeric
price	numeric

Tabela: Esquema da Tabela products2

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades e preço maior que R\$200,00.

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades e preço maior que R\$200,00.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades e preço maior que R\$200,00.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```
1 SELECT id, name
2 FROM products2
3 WHERE amount > 2000 AND price
   > 200;
```

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades e preço maior que R\$200,00.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```

1 SELECT id, name
2 FROM products2
3 WHERE amount > 2000 AND price
   > 200;
    
```

id	name
5	Chair

Tabela: Exemplo de Saída

Coluna	Tipo
id (PK)	numeric
name	varchar
amount	numeric
price	numeric

Tabela: Esquema da Tabela products2

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades e preço maior que R\$200,00, ordenados por preço em ordem decrescente.

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades e preço maior que R\$200,00, ordenados por preço em ordem decrescente.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades e preço maior que R\$200,00, ordenados por preço em ordem decrescente.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```
1 SELECT id, name
2 FROM products2
3 WHERE amount > 2000 AND price
   > 200
4 ORDER BY price DESC;
```

Requisito do Relatório

O setor de vendas da empresa precisa de um relatório que mostre o código e o nome dos produtos que têm quantidade maior que 2000 unidades e preço maior que R\$200,00, ordenados por preço em ordem decrescente.

id	name	amount	price
1	Two-door wardrobe	0	80
2	Dining table	1000	560
3	Towel holder	10000	5.50
4	Computer desk	0	100
5	Chair	3000	210.64
6	Single bed	750	99

Tabela: Tabela products2

```

1 SELECT id, name
2 FROM products2
3 WHERE amount > 2000 AND price
  > 200
4 ORDER BY price DESC;
```

id	name
5	Chair

Tabela: Exemplo de Saída

Coluna	Tipo
id (PK)	numeric
name	varchar
amount	numeric
price	numeric

Tabela: Esquema da Tabela products2

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	idade	nota
1	João Silva	20	75
2	Maria Oliveira	22	85
3	Pedro Santos	21	90
4	Ana Costa	23	70
5	Lucas Pereira	19	95
6	Clara Souza	20	88

Tabela: Tabela alunos

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	idade	nota
1	João Silva	20	75
2	Maria Oliveira	22	85
3	Pedro Santos	21	90
4	Ana Costa	23	70
5	Lucas Pereira	19	95
6	Clara Souza	20	88

Tabela: Tabela alunos

```
1 SELECT id, nome
2 FROM alunos
3 WHERE nota > 80;
```

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	idade	nota
1	João Silva	20	75
2	Maria Oliveira	22	85
3	Pedro Santos	21	90
4	Ana Costa	23	70
5	Lucas Pereira	19	95
6	Clara Souza	20	88

Tabela: Tabela alunos

```
1 SELECT id, nome
2 FROM alunos
3 WHERE nota > 80;
```

id	nome
2	Maria Oliveira
3	Pedro Santos
5	Lucas Pereira
6	Clara Souza

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	idade	nota
1	João Silva	20	75
2	Maria Oliveira	22	85
3	Pedro Santos	21	90
4	Ana Costa	23	70
5	Lucas Pereira	19	95
6	Clara Souza	20	88

Tabela: Tabela alunos

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	idade	nota
1	João Silva	20	75
2	Maria Oliveira	22	85
3	Pedro Santos	21	90
4	Ana Costa	23	70
5	Lucas Pereira	19	95
6	Clara Souza	20	88

Tabela: Tabela alunos

```
1 SELECT id, nome
2 FROM alunos
3 WHERE idade < 22 AND nota > 80;
```

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	idade	nota
1	João Silva	20	75
2	Maria Oliveira	22	85
3	Pedro Santos	21	90
4	Ana Costa	23	70
5	Lucas Pereira	19	95
6	Clara Souza	20	88

Tabela: Tabela alunos

```
1 SELECT id, nome
2 FROM alunos
3 WHERE idade < 22 AND nota > 80;
```

id	nome
5	Lucas Pereira
6	Clara Souza

Tabela: Resultado do Relatório

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso
1	João Silva	Engenharia
2	Maria Oliveira	Medicina
3	Pedro Santos	Engenharia
4	Ana Costa	Direito
5	Lucas Pereira	Medicina
6	Clara Souza	Engenharia

Tabela: Tabela alunos

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso
1	João Silva	Engenharia
2	Maria Oliveira	Medicina
3	Pedro Santos	Engenharia
4	Ana Costa	Direito
5	Lucas Pereira	Medicina
6	Clara Souza	Engenharia

Tabela: Tabela alunos

```
1 SELECT DISTINCT curso
2 FROM alunos;
```

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso
1	João Silva	Engenharia
2	Maria Oliveira	Medicina
3	Pedro Santos	Engenharia
4	Ana Costa	Direito
5	Lucas Pereira	Medicina
6	Clara Souza	Engenharia

Tabela: Tabela alunos

```
1 SELECT DISTINCT curso
2 FROM alunos;
```

curso
Engenharia
Medicina
Direito

Tabela: Resultado do Relatório

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	idade	nota
1	João Silva	20	75
2	Maria Oliveira	22	85
3	Pedro Santos	21	90
4	Ana Costa	23	70
5	Lucas Pereira	19	95
6	Clara Souza	20	88

Tabela: Tabela alunos

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	idade	nota
1	João Silva	20	75
2	Maria Oliveira	22	85
3	Pedro Santos	21	90
4	Ana Costa	23	70
5	Lucas Pereira	19	95
6	Clara Souza	20	88

Tabela: Tabela alunos

```
1 SELECT AVG(nota) as  
   alguma_coisa_notas  
2 FROM alunos;
```

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	idade	nota
1	João Silva	20	75
2	Maria Oliveira	22	85
3	Pedro Santos	21	90
4	Ana Costa	23	70
5	Lucas Pereira	19	95
6	Clara Souza	20	88

Tabela: Tabela alunos

```
1 SELECT AVG(nota) as  
   alguma_coisa_notas  
2 FROM alunos;
```

alguma_coisa_notas
83.83

Tabela: Resultado do Relatório

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	nota
1	João Silva	Engenharia	75
2	Maria Oliveira	Medicina	85
3	Pedro Santos	Engenharia	90
4	Ana Costa	Direito	70
5	Lucas Pereira	Medicina	95
6	Clara Souza	Engenharia	88

Tabela: Tabela alunos

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	nota
1	João Silva	Engenharia	75
2	Maria Oliveira	Medicina	85
3	Pedro Santos	Engenharia	90
4	Ana Costa	Direito	70
5	Lucas Pereira	Medicina	95
6	Clara Souza	Engenharia	88

Tabela: Tabela alunos

```
1 SELECT curso, AVG(nota) as  
   media_notas  
2 FROM alunos  
3 GROUP BY curso;
```

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	nota
1	João Silva	Engenharia	75
2	Maria Oliveira	Medicina	85
3	Pedro Santos	Engenharia	90
4	Ana Costa	Direito	70
5	Lucas Pereira	Medicina	95
6	Clara Souza	Engenharia	88

Tabela: Tabela alunos

```

1 SELECT curso, AVG(nota) as
   media_notas
2 FROM alunos
3 GROUP BY curso;

```

curso	media_notas
Engenharia	84.33
Medicina	90.00
Direito	70.00

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso
1	João Silva	Engenharia
2	Maria Oliveira	Medicina
3	Pedro Santos	Engenharia
4	Ana Costa	Direito
5	Lucas Pereira	Medicina
6	Clara Souza	Engenharia
7	Bruno Lima	Direito
8	Paula Araújo	Medicina

Tabela: Tabela alunos2

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso
1	João Silva	Engenharia
2	Maria Oliveira	Medicina
3	Pedro Santos	Engenharia
4	Ana Costa	Direito
5	Lucas Pereira	Medicina
6	Clara Souza	Engenharia
7	Bruno Lima	Direito
8	Paula Araújo	Medicina

Tabela: Tabela alunos2

```
1 SELECT curso, COUNT(*) as  
   quantidade_alunos  
2 FROM alunos2  
3 GROUP BY curso;
```

Atividade 13

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso
1	João Silva	Engenharia
2	Maria Oliveira	Medicina
3	Pedro Santos	Engenharia
4	Ana Costa	Direito
5	Lucas Pereira	Medicina
6	Clara Souza	Engenharia
7	Bruno Lima	Direito
8	Paula Araújo	Medicina

Tabela: Tabela alunos2

```
1 SELECT curso, COUNT(*) as  
   quantidade_alunos  
2 FROM alunos2  
3 GROUP BY curso;
```

curso	quantidade_alunos
Engenharia	3
Medicina	3
Direito	2

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	nota
1	João Silva	Engenharia	75
2	Maria Oliveira	Medicina	85
3	Pedro Santos	Engenharia	90
4	Ana Costa	Direito	70
5	Lucas Pereira	Medicina	95
6	Clara Souza	Engenharia	88
7	Bruno Lima	Direito	78
8	Paula Araújo	Medicina	92

Tabela: Tabela alunos2

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	nota
1	João Silva	Engenharia	75
2	Maria Oliveira	Medicina	85
3	Pedro Santos	Engenharia	90
4	Ana Costa	Direito	70
5	Lucas Pereira	Medicina	95
6	Clara Souza	Engenharia	88
7	Bruno Lima	Direito	78
8	Paula Araújo	Medicina	92

Tabela: Tabela alunos2

```
1 SELECT curso, SUM(nota) as  
   soma_notas  
2 FROM alunos2  
3 GROUP BY curso;
```

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	nota
1	João Silva	Engenharia	75
2	Maria Oliveira	Medicina	85
3	Pedro Santos	Engenharia	90
4	Ana Costa	Direito	70
5	Lucas Pereira	Medicina	95
6	Clara Souza	Engenharia	88
7	Bruno Lima	Direito	78
8	Paula Araújo	Medicina	92

Tabela: Tabela alunos2

```

1 SELECT curso, SUM(nota) as
   soma_notas
2 FROM alunos2
3 GROUP BY curso;
    
```

curso	soma_notas
Engenharia	253
Medicina	272
Direito	148

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	nota
1	João Silva	Engenharia	75
2	Maria Oliveira	Medicina	85
3	Pedro Santos	Engenharia	90
4	Ana Costa	Direito	70
5	Lucas Pereira	Medicina	95
6	Clara Souza	Engenharia	88
7	Bruno Lima	Direito	78
8	Paula Araújo	Medicina	92
9	Carlos Menezes	Engenharia	83
10	Fernanda Lima	Direito	81

Tabela: Tabela alunos3

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	nota
1	João Silva	Engenharia	75
2	Maria Oliveira	Medicina	85
3	Pedro Santos	Engenharia	90
4	Ana Costa	Direito	70
5	Lucas Pereira	Medicina	95
6	Clara Souza	Engenharia	88
7	Bruno Lima	Direito	78
8	Paula Araújo	Medicina	92
9	Carlos Menezes	Engenharia	83
10	Fernanda Lima	Direito	81

```

1 SELECT curso, COUNT(*)
   as quantidade_alunos
2 FROM alunos3
3 WHERE nota > 80
4 GROUP BY curso;

```

Tabela: Tabela alunos3

Pergunta

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	nota
1	João Silva	Engenharia	75
2	Maria Oliveira	Medicina	85
3	Pedro Santos	Engenharia	90
4	Ana Costa	Direito	70
5	Lucas Pereira	Medicina	95
6	Clara Souza	Engenharia	88
7	Bruno Lima	Direito	78
8	Paula Araújo	Medicina	92
9	Carlos Menezes	Engenharia	83
10	Fernanda Lima	Direito	81

```

1 SELECT curso, COUNT(*)
   as quantidade_alunos
2 FROM alunos3
3 WHERE nota > 80
4 GROUP BY curso;

```

curso	quantidade_aluno
Engenharia	3
Medicina	3
Direito	1

Tabela: Resultado do Relatório

Tabela: Tabela alunos3

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	idade
1	João Silva	Engenharia	20
2	Maria Oliveira	Medicina	22
3	Pedro Santos	Engenharia	21
4	Ana Costa	Direito	23
5	Lucas Pereira	Medicina	24
6	Clara Souza	Engenharia	22
7	Bruno Lima	Direito	21
8	Paula Araújo	Medicina	23
9	Carlos Menezes	Engenharia	24
10	Fernanda Lima	Direito	22

Tabela: Tabela alunos3

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	idade
1	João Silva	Engenharia	20
2	Maria Oliveira	Medicina	22
3	Pedro Santos	Engenharia	21
4	Ana Costa	Direito	23
5	Lucas Pereira	Medicina	24
6	Clara Souza	Engenharia	22
7	Bruno Lima	Direito	21
8	Paula Araújo	Medicina	23
9	Carlos Menezes	Engenharia	24
10	Fernanda Lima	Direito	22

```

1 SELECT curso ,
      AVG(idade) as
      media_idade
2 FROM alunos3
3 GROUP BY curso;
    
```

Tabela: Tabela alunos3

Com base na tabela e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso	idade
1	João Silva	Engenharia	20
2	Maria Oliveira	Medicina	22
3	Pedro Santos	Engenharia	21
4	Ana Costa	Direito	23
5	Lucas Pereira	Medicina	24
6	Clara Souza	Engenharia	22
7	Bruno Lima	Direito	21
8	Paula Araújo	Medicina	23
9	Carlos Menezes	Engenharia	24
10	Fernanda Lima	Direito	22

```

1 SELECT curso ,
      AVG(idade) as
      media_idade
2 FROM alunos3
3 GROUP BY curso;

```

curso	media_idade
Engenharia	21.75
Medicina	23.00
Direito	22.00

Tabela: Resultado do Relatório

Tabela: Tabela alunos3

Pergunta

Com base nas tabelas e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	curso
1	João Silva	Engenharia
2	Maria Oliveira	Medicina
3	Pedro Santos	Engenharia
4	Ana Costa	Direito
5	Lucas Pereira	Medicina
6	Clara Souza	Engenharia
7	Bruno Lima	Direito
8	Paula Araújo	Medicina
9	Carlos Menezes	Engenharia
10	Fernanda Lima	Direito

Tabela: Tabela alunos4

curso	coordenador
Engenharia	Dr. Alberto
Medicina	Dr. Beatriz
Direito	Dr. Carlos

Tabela: Tabela cursos

```

1 SELECT alunos4.nome ,
   cursos.coordenador
2 FROM alunos4
3 INNER JOIN cursos
4 ON alunos4.curso =
   cursos.curso;

```

Pergunta

Com base nas tabelas e na consulta SQL fornecidas, qual seria o resultado do relatório?

nome	coordenador
João Silva	Dr. Alberto
Pedro Santos	Dr. Alberto
Clara Souza	Dr. Alberto
Carlos Menezes	Dr. Alberto
Maria Oliveira	Dr. Beatriz
Lucas Pereira	Dr. Beatriz
Paula Araújo	Dr. Beatriz
Ana Costa	Dr. Carlos
Bruno Lima	Dr. Carlos
Fernanda Lima	Dr. Carlos

Tabela: Resultado do Relatório

Pergunta

Com base nas tabelas e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	nome	id_dep
1	Alice Martins	1
2	Roberto Lima	2
3	Carla Souza	1
4	Joana Ferreira	3
5	Paulo Silveira	2
6	Fernanda Costa	1
7	Miguel Alves	3
8	Sofia Rodrigues	2
9	Lucas Barros	1
10	Ana Clara Melo	3

Tabela: Tabela funcionarios

id_dep	depar	gerente
1	Recursos Humanos	Sra. Silva
2	TI	Sr. Almeida
3	Marketing	Srta. Gomes

Tabela: Tabela departamentos

```

1 SELECT funcionarios.nome ,
   departamentos.gerente
2 FROM funcionarios
3 INNER JOIN departamentos
4 ON funcionarios.id_dep =
   departamentos.id_dep;
    
```

Pergunta

Com base nas tabelas e na consulta SQL fornecidas, qual seria o resultado do relatório?

nome	gerente
Alice Martins	Sra. Silva
Carla Souza	Sra. Silva
Fernanda Costa	Sra. Silva
Lucas Barros	Sra. Silva
Roberto Lima	Sr. Almeida
Paulo Silveira	Sr. Almeida
Sofia Rodrigues	Sr. Almeida
Joana Ferreira	Srta. Gomes
Miguel Alves	Srta. Gomes
Ana Clara Melo	Srta. Gomes

Tabela: Resultado do Relatório

Pergunta

Com base nas tabelas e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	produto	id_categ
1	Notebook	1
2	Smartphone	2
3	Teclado	3
4	Monitor	1
5	Mouse	3
6	Tablet	2
7	Impressora	1
8	Headset	3
9	Smartwatch	2
10	Scanner	1

Tabela: Tabela produtos

id_categ	categoria	responsavel
1	Informática	João
2	Mobile	Maria
3	Periféricos	Carlos

Tabela: Tabela categorias

```

1 SELECT produtos.produto,
   categorias.responsavel
2 FROM produtos
3 INNER JOIN categorias
4 ON produtos._categ =
   categorias._categ;

```


Pergunta

Com base nas tabelas e na consulta SQL fornecidas, qual seria o resultado do relatório?

produto	responsavel
Notebook	João
Monitor	João
Impressora	João
Scanner	João
Smartphone	Maria
Tablet	Maria
Smartwatch	Maria
Teclado	Carlos
Mouse	Carlos
Headset	Carlos

Tabela: Resultado do Relatório

Pergunta

Com base nas tabelas e na consulta SQL fornecidas, qual seria o resultado do relatório?

id	funcionario	id_projeto
1	Alice	1
2	Roberto	2
3	Carla	1
4	Joana	3
5	Paulo	2
6	Fernanda	1
7	Miguel	3
8	Sofia	2
9	Lucas	1
10	Ana Clara	3

Tabela: Tabela funcionarios2

id_projeto	projeto	gerente
1	Projeto A	Dr. Silva
2	Projeto B	Dr. Almeida
3	Projeto C	Dr. Gomes

Tabela: Tabela projetos

```

1 SELECT
    funcionarios2.funcionario,
    projetos.gerente
2 FROM funcionarios2
3 INNER JOIN projetos
4 ON
    funcionarios2.id_projeto
    = projetos.id_projeto;
    
```

Pergunta

Com base nas tabelas e na consulta SQL fornecidas, qual seria o resultado do relatório?

funcionario	gerente
Alice	Dr. Silva
Carla	Dr. Silva
Fernanda	Dr. Silva
Lucas	Dr. Silva
Roberto	Dr. Almeida
Paulo	Dr. Almeida
Sofia	Dr. Almeida
Joana	Dr. Gomes
Miguel	Dr. Gomes
Ana Clara	Dr. Gomes

Tabela: Resultado do Relatório

```

1 SELECT funcionarios3.nome, departamentos2.departamento,
   departamentos2.gerente, funcionarios3.salario
2 FROM funcionarios3
3 INNER JOIN departamentos2
4 ON funcionarios3.id_dep = departamentos2.id_dep;

```

id	nome	id_dep	sal			
1	João Silva	1	5000	id_dep	depar	gerente
2	Maria Oliveira	2	6000	1	RH	Dr. Silva
3	Pedro Santos	1	5500	2	TI	Dr. Almeida
4	Ana Costa	3	6200	3	Marketing	Dr. Gomes
5	Lucas Pereira	2	5800	<p>Tabela: Tabela departamentos</p>		
6	Clara Souza	1	5300			
7	Bruno Lima	3	6100			
8	Paula Araújo	2	5700			
9	Carlos Menezes	1	5400			
10	Fernanda Lima	3	6000			

Tabela: Tabela funcionarios3

Pergunta

Com base nas tabelas e na consulta SQL fornecidas, qual seria o resultado do relatório?

nome	departamento	gerente	salario
João Silva	RH	Dr. Silva	5000
Pedro Santos	RH	Dr. Silva	5500
Clara Souza	RH	Dr. Silva	5300
Carlos Menezes	RH	Dr. Silva	5400
Maria Oliveira	TI	Dr. Almeida	6000
Lucas Pereira	TI	Dr. Almeida	5800
Paula Araújo	TI	Dr. Almeida	5700
Ana Costa	Marketing	Dr. Gomes	6200
Bruno Lima	Marketing	Dr. Gomes	6100
Fernanda Lima	Marketing	Dr. Gomes	6000

Tabela: Resultado do Relatório