

The background is black with several abstract orange geometric shapes and lines. On the left, there are some thin lines and a small circle. In the center, there is a large orange rounded rectangle. On the right, there is a complex network of orange lines and circles, resembling a molecular structure or a network diagram.

Java Orientado a Objetos

RECODE
pro

Módulo 03 – Java Orientado a Objetos Aula 03



Método construtor

Um construtor é bastante parecido com um método comum, mas ele não é um. Diferente dos métodos, um construtor tem o mesmo nome da classe e não tem um retorno declarado. Mas, se nunca escrevemos esse construtor, quem o fez? Sempre que você não criar um construtor para suas classes, o compilador fará isso para você.

- É uma operação especial da classe, que executa no momento da instanciação do objeto Usos comuns:
- Iniciar valores dos atributos
- Permitir ou obrigar que o objeto receba dados no momento de sua instanciação;
- Se um construtor customizado não for especificado, a classe disponibiliza o construtor padrão: **Product p = new Product();**
- É possível especificar mais de um construtor na mesma classe (sobrecarga)



Módulo 03 – Java Orientado a Objetos Aula 03



```
public class Caixa {  
  
    public double saldo = 0;  
  
    public Caixa(double saldo) {  
        this.saldo = saldo;  
    }  
  
    public Caixa() {  
    }  
  
    void sacar(double valor) {  
        this.saldo = saldo - valor;  
    }  
  
    void depositar(double valor) {  
        this.saldo = saldo + valor;  
    }  
  
    double exibirSaldo() {  
        return this.saldo;  
    }  
}
```

← Método construtor inicializando o atributo saldo, constrói um objeto com o valor saldo setado

← Método construtor padrão, constrói um objeto vazio

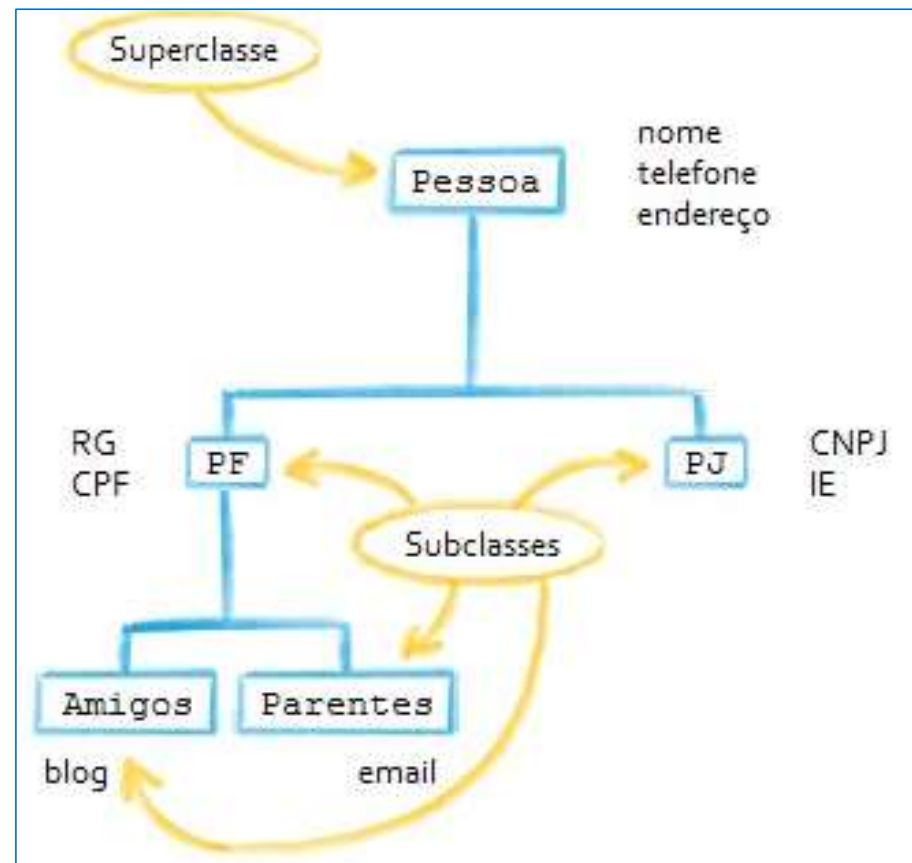


Módulo 03 – Java Orientado a Objetos Aula 03



Herança

Herança (ou generalização) é o mecanismo pelo qual uma classe (subclasse) pode estender outra classe (superclasse), aproveitando seus comportamentos (métodos) e variáveis possíveis (atributos). Um exemplo de herança: Mamífero é superclasse de Humano. Ou seja, um Humano é um mamífero



Módulo 03 – Java Orientado a Objetos Aula 03



Herança

Super classe ou classe base Cliente

```
public class Cliente {  
    String nome;  
    String telefone;  
  
    public Cliente(String nome, String telefone) {  
        this.nome = nome;  
        this.telefone = telefone;  
    }  
  
    public Cliente() {  
  
    }  
}
```



Módulo 03 – Java Orientado a Objetos Aula 03



Herança

Subclasse ou classe filha - ClientePF

```
public class ClientePF extends Cliente {
```

```
// A palavra reservada extends indica que a classe ClientePF  
// vai herdar tudo que a classe Cliente tiver disponível para a // Herança  
(atributos e métodos se houver).
```

```
    String rg;  
    String cpf;
```

```
// Atributos da classe ClientePF
```

```
    public ClientePF() {  
        super();  
    }
```

```
// Método construtor da classe chamando o construtor da classe // mãe, responsável  
por contribuir o objeto da classe na memória
```

```
}
```

```
}
```

Módulo 03 – Java Orientado a Objetos Aula 03



Herança

Subclasse ou classe filha ClientePJ

```
public class ClientePJ extends Cliente{
    // A palavra reservada extends indica que a classe ClientePJ
    // vai herdar tudo que a classe Cliente tiver disponível para a // Herança
    // (atributos e métodos se houver).
    String cnpj;
    String ie;
    // Atributos da classe ClientePJ

    public ClientePJ(String nome, String telefone, String cnpj, String ie) {

        super(nome, telefone);
        this.cnpj = cnpj;
        this.ie = ie;
    }

    // Método construtor da classe chamando o construtor da classe // mãe, e recebendo
    // seus próprios atributos por parâmetros.
}
```



Módulo 03 – Java Orientado a Objetos Aula 03



Polimorfismo

Polimorfismo é a capacidade de um método poder ser implementado de diferentes formas, ou mesmo de realizar coisas diferentes: Sobrescrita e sobrecarga são dois tipos de polimorfismo.

Sobrecarga - Ocorre quando dois ou mais métodos possuindo o mesmo nome são implementados com assinaturas diferentes, ou seja, recebem parâmetros de diferentes tipos ou em diferentes quantidades. A escolha de qual método utilizar baseia-se nos parâmetros recebidos na chamada do mesmo.

```
public ClientePF() {  
  
    }  
    public ClientePF(String nome, String telefone, String rg, String cpf) {  
        super(nome, telefone);  
        this.rg = rg;  
        this.cpf = cpf;  
    }  
}
```



Módulo 03 – Java Orientado a Objetos Aula 03



Sobrescrita - Ocorre quando uma classe filha redefine um método herdado. Os métodos têm o mesmo nome e a mesma assinatura, mas na classe filha ele está implementado de forma diferente, ou seja, ele sobrescreve o método já existente da classe pai.

```
public class Funcionario {  
  
    protected String nome;  
    protected String cpf;  
    protected double salario;  
  
    public double calculaBonus() {  
        return salario * 0.10;  
    }  
}
```

```
public class Gerente extends Funcionario {  
  
    String setor;  
  
    @Override  
    public double calculaBonus() {  
        return this.salario * 0.20;  
    }  
  
    // método sobrescrito (Herdado),  
    calculo diferente da superclasse, porém, mantém o mesmo  
    nome.  
}
```

Módulo 03 – Java Orientado a Objetos Aula 03



Interface

Em Java, há uma forma para se tirar proveito de todos os benefícios do polimorfismo sem ter que acoplar tanto as suas classes com vários níveis de herança. Você pode estabelecer um fator em comum entre as classes, criando uma espécie de contrato. Para esse contrato, não importa a forma como será implementado, a única coisa que importa é que seus métodos (cláusulas) sejam implementados de alguma forma. Esse tipo de contrato Java é conhecido como Interface.



Módulo 03 – Java Orientado a Objetos Aula 03



Exemplo

No sistema de um banco, podemos definir uma interface (contrato) para padronizar as assinaturas dos métodos oferecidos pelos objetos que representam as contas do banco.

```
interface Conta {  
    void deposita(double valor);  
    void saca (double valor);  
}
```

Os métodos de uma interface não possuem corpo (implementação) pois serão implementados nas classes vinculadas a essa interface. Todos os métodos de uma interface devem ser públicos.

As classes que definem os diversos tipos de contas que existem no banco devem implementar (assinar) a interface Conta.



Módulo 03 – Java Orientado a Objetos Aula 03



Classe ContaPoupanca implementando a interface Conta

```
class ContaPoupanca implements Conta {  
  
    public void deposita(double valor) {  
  
        this.saldo = this.saldo + valor  
  
    }  
  
    public void saca(double valor) {  
  
        this.saldo = this.saldo - valor  
  
    }  
  
}
```

Classe ContaCorrente implementando a interface Conta

```
class ContaCorrente implements Conta {  
  
    public void deposita(double valor) {  
        this.saldo = this.saldo + valor  
    }  
  
    public void saca (double valor) {  
        this.saldo = this.saldo + valor  
    }  
  
}
```

Módulo 03 – Java Orientado a Objetos Aula 03



Exercício 01 - Crie uma classe animal com os atributos e métodos gerais para as subclasses, crie uma classe cachorro herdando de animal e faça-o latir. Crie uma classe gato herdando de animal e faça-o miar. Faça os dois animais caminharem mostrando o número de passos de cada um. Observação: os animais só podem andar se estiverem alimentados, para isso crie um atributo para fazer essa condição, A cada 100 passos o animal deve ser alimentado, caso contrário ele não consegue andar.

Módulo 03 – Java Orientado a Objetos Aula 03



Resolução

```
public class Animal {
    String nome;
    String raca;
    int numeroPatas = 0;
    int passos = 0;
    boolean alimentado = false;

    void alimentar() {
        alimentado = true;
        passos = 0;
    }
    void andar() {
        if (this.alimentado && this.passos < 100) {
            this.passos++;
            System.out.println("Andou " + this.passos + " passos");

        } else {
            System.out.println("Alimente o animal");
        }
    }
    void emitirSom(String som) {
        System.out.println("O animal emitiu um som: " + som);
    }
}
```

Módulo 03 – Java Orientado a Objetos Aula 03



```
public class Cachorro extends Animal{  
  
}
```

```
public class Gato extends Animal{  
  
}
```

```
public class Principal {  
  
    public static void main(String[] args) {  
        Gato bia = new Gato();  
        bia.andar();  
        bia.alimentar();  
        bia.andar();  
        bia.emitirSom("Miaaaauuuu");  
    }  
}
```

Resolução