Atividades - POO

Nível Fácil

1. `late` e `final`

Atividade 1:

- **Contexto:** Um sistema precisa inicializar uma variável apenas quando ela for necessária, mas garantir que essa variável não seja reatribuída após a inicialização.
- **Tarefa:** Crie uma classe `Usuario` com uma propriedade `String nome`. Use o modificador `late` para inicializar o nome no método `inicializarNome` e o modificador `final` para garantir que ele não seja reatribuído.

Atividade 2:

- **Contexto:** Um sistema de controle de inventário precisa garantir que o valor de `codigoProduto` seja atribuído uma vez e nunca alterado.
- **Tarefa:** Crie uma classe `Produto` com uma propriedade `final int codigoProduto`. Inicialize essa propriedade no construtor e tente modificá-la posteriormente para ver o erro.

2. Classes

Atividade 3:

- **Contexto:** Um aplicativo de gerenciamento de produtos precisa armazenar informações sobre cada produto.
- **Tarefa:** Crie uma classe `Produto` com as propriedades `nome` e `preco`. Defina um construtor que inicialize essas propriedades e exiba os detalhes do produto.

Atividade 4:

- **Contexto:** Um aplicativo de contatos precisa armazenar informações básicas de uma pessoa.
- **Tarefa:** Escreva uma classe `Pessoa` com as propriedades `nome` e `idade`. Inicialize essas propriedades no construtor e crie um método para exibir as informações da pessoa.

3. Construtor

Atividade 5:

- **Contexto:** Um sistema de eventos precisa inicializar objetos `Evento` com o nome e a data do evento.
- **Tarefa:** Escreva um construtor para a classe `Evento` que receba `nome` e `data` como parâmetros e inicialize as propriedades correspondentes.

Atividade 6:

- **Contexto:** Um aplicativo de endereço precisa criar objetos `Endereco` com propriedades nomeadas `rua`, `cidade`, e `estado`.
- **Tarefa:** Escreva a classe `Endereco` com um construtor que inicialize essas propriedades.

4. Métodos

Atividade 7:

- **Contexto:** Um aplicativo de agendamento deseja exibir uma saudação ao usuário.
- **Tarefa:** Adicione um método `saudacao` na classe `Agenda` que receba o nome do usuário e imprima uma mensagem personalizada.

Atividade 8:

- **Contexto:** Um sistema de geometria precisa calcular a área de um retângulo.
- **Tarefa:** Crie uma classe `Retangulo` com as propriedades `largura` e `altura`, e um método `calcularArea` que retorne a área do retângulo.

Atividade 9:

- **Contexto:** Um sistema de CRM precisa formatar os detalhes de contato de um cliente.
- **Tarefa:** Adicione um método `formatarContato` na classe `Cliente` que retorne uma string com o nome e telefone formatados.

Atividade 10:

- **Contexto:** Um aplicativo de utilidades precisa verificar se um número é par.
- **Tarefa:** Adicione um método estático `ePar` na classe `Util` que retorne se um número é par.

Atividade 11:

- **Contexto:** Um sistema de biblioteca precisa exibir informações sobre os livros.
- **Tarefa:** Adicione um método `exibirInformacoes` na classe `Livro` que imprima o título e o autor.

5. Construtores com Valor Padrão

Atividade 12:

- **Contexto:** Um programa de descontos precisa inicializar objetos `Produto` com preço original e um desconto opcional.
- **Tarefa:** Escreva um construtor para a classe `Produto` que inicialize essas propriedades, com o desconto tendo um valor padrão de 0.

Atividade 13:

- **Contexto:** Um sistema de gestão de produtos precisa criar objetos `Produto` com um construtor nomeado `Produto.desconto` que inicialize a propriedade `preco` com desconto aplicado.
- **Tarefa:** Escreva a classe `Produto` com este construtor nomeado.

Nível Intermediário

6. Encapsulamento

Atividade 14:

- **Contexto:** Um sistema de segurança deseja proteger os dados dos usuários.
- **Tarefa:** Crie uma classe `ContaBancaria` com um atributo privado `_saldo`. Adicione métodos públicos para depositar e sacar dinheiro, e para obter o saldo atual.

Atividade 15:

- **Contexto:** Um sistema de controle de acesso precisa verificar se uma pessoa é maior de idade.
- **Tarefa:** Adicione um método `eMaiorDeIdade` na classe `Pessoa` que retorne se a idade é maior ou igual a 18.

Atividade 16:

- **Contexto:** Um sistema de gerenciamento de clientes precisa formatar o nome dos clientes para maiúsculas.
- **Tarefa:** Adicione um método `formatarNome` na classe `Cliente` que converta o nome para letras maiúsculas.

7. Herança

Atividade 17:

- **Contexto:** Um jogo precisa criar diferentes tipos de personagens, todos com um nome e uma vida.
- **Tarefa:** Crie uma classe `Personagem` com as propriedades `nome` e `vida`, e um método `atacar`. Em seguida, crie uma subclasse `Guerreiro` que herde de `Personagem` e sobrescreva o método `atacar`.

Atividade 18:

- **Contexto:** Um sistema de gestão de funcionários precisa categorizar os empregados em diferentes níveis hierárquicos.
- **Tarefa:** Crie uma classe `Funcionario` com as propriedades `nome` e `salario`. Em seguida, crie subclasses `Gerente` e `Assistente` que herdem de `Funcionario` e adicionem propriedades específicas.

Atividade 19:

- **Contexto:** Um sistema de veículos precisa lidar com diferentes tipos de veículos.
- **Tarefa:** Crie uma classe `Veiculo` com a propriedade `nome`. Crie subclasses `Carro` e `Moto` que herdem de `Veiculo` e adicionem métodos específicos.

8. Polimorfismo

Atividade 20:

- **Contexto:** Um sistema de gerenciamento de veículos precisa tratar carros e motos de maneira polimórfica.
- **Tarefa:** Crie uma classe `Veiculo` com um método `mover`. Em seguida, crie subclasses `Carro` e `Moto` que sobrescrevam o método `mover` de maneiras diferentes.

Atividade 21:

- **Contexto:** Um sistema de gestão de eventos precisa realizar diferentes tipos de ações com base no tipo de evento.
- **Tarefa:** Crie uma classe `Evento` com um método `executar`. Em seguida, crie subclasses `Show` e `Palestra` que sobrescrevam o método `executar` com implementações específicas para cada tipo de evento.

Atividade 22:

- **Contexto:** Um sistema escolar precisa calcular a média das notas dos alunos.
- **Tarefa:** Adicione um método `calcularMedia` na classe `Aluno` que receba uma lista de notas e retorne a média.

9. Construtores e Métodos

Atividade 23:

- **Contexto:** Um aplicativo precisa converter valores de dólares para euros.
- **Tarefa:** Adicione um método `converterMoeda` na classe `Financas` que receba um valor em dólares e retorne o valor em euros, considerando uma taxa de conversão fixa.

Atividade 24:

- **Contexto:** Um aplicativo precisa exibir uma saudação ao usuário dependendo do horário do dia.
- **Tarefa:** Adicione um método `saudacao` na classe `Usuario` que receba o nome do usuário e o horário (manhã, tarde, noite) como parâmetros nomeados e imprima uma saudação adequada.

Atividade 25:

- **Contexto:** Um sistema de busca precisa encontrar um valor em uma lista ordenada.
- **Tarefa:** Adicione um método `buscar` na classe `Busca` que implemente a busca binária e retorne a posição do valor na lista.

Atividade 26:

- **Contexto:** Um sistema de palavras precisa verificar se uma palavra é um palíndromo.
- **Tarefa:** Adicione um método `ePalindromo` na classe `Palavra` que receba uma palavra e retorne se ela é um palíndromo.

Nível Avançado

10. Classe Abstrata e Método Abstrato

Atividade 27:

- **Contexto:** Um sistema de desenho geométrico precisa suportar diferentes formas geométricas.
- **Tarefa:** Crie uma classe abstrata `Forma` com um método abstrato `calcularArea`. Em seguida, crie as subclasses `Circulo` e `Retangulo` que implementam o método `calcularArea` de forma específica.

Atividade 28:

- **Contexto:** Um sistema de classificação precisa agrupar pessoas por faixas etárias.
- **Tarefa

:** Adicione um método `classificarIdades` na classe `Classificacao` que receba uma lista de idades e retorne uma lista de strings com as classificações ("criança", "adolescente", "adulto", "idoso").

Atividade 29:

- **Contexto:** Um sistema acadêmico precisa calcular a média ponderada das notas dos alunos.
- **Tarefa:** Adicione um método `calcularMediaPonderada` na classe `Academico` que receba uma lista de notas e uma lista de pesos e retorne a média ponderada.

Atividade 30:

- **Contexto:** Um aplicativo meteorológico precisa converter temperaturas de Celsius para Fahrenheit.
- **Tarefa:** Adicione um método `converterParaFahrenheit` na classe `Temperatura` que receba uma temperatura em Celsius e retorne o valor em Fahrenheit.
