Atividade - Herança, Classe Abstrata e Interface

Observação: faça uso dos conceitos de orientação a objetos: herança, classe abstrata, interface, encapsulamento e polimorfismo.

Exercício 1 (Herança) – Conta Bancária: Elabore uma classe ContaBancaria, com os seguintes membros:

- Cliente
- Número da conta
- Saldo
- Método sacar (o saldo não pode ficar negativo)
- Método depositar

Agora acrescente ao projeto duas classes herdadas de ContaBancaria: ContaPoupança e ContaEspecial, com as seguintes características a mais:

- Classe ContaPoupança:
  - Dia de rendimento;
  - Método "calcularNovoSaldo", recebe a taxa de rendimento da poupança e atualiza o saldo.
- Classe ContaEspecial:
  - Limite;
  - Redefinição do método sacar, permitindo saldo negativo até o valor do limite;

Após a implementação das classes acima, você deverá implementar uma classe Contas, contendo o método principal. Nesta classe, você deverá implementar:

1. Incluir dados relativos a(s) conta(s) de um cliente;
2. Sacar um determinado valor da(s) sua(s) conta(s);
3. Depositar um determinado valor na(s) sua(s) conta(s);
4. Mostrar o novo saldo do cliente, a partir da taxa de rendimento, daqueles que possuem conta poupança;
5. Mostrar os dados da(s) conta(s) de um cliente.

Exercício 2 (Classe Abstrata) - Implemente um sistema de gerenciamento de veículos de uma frota de transporte que utilize uma classe abstrata. O sistema deve permitir a criação de diferentes tipos de veículos: **Carro**, **Caminhão** e **Ônibus**. Todos os veículos possuem as seguintes características comuns:

- **placa** (String)
- **marca** (String)
- **modelo** (String)
- **ano de fabricação** (int)

A classe abstrata Veiculo deve definir métodos abstratos para:

1. **calcularIPVA()**: Calcula o valor do IPVA a ser pago de acordo com o tipo de veículo.
2. **exibirDetalhes()**: Exibe os detalhes completos do veículo.

As subclasses **Carro**, **Caminhão** e **Ônibus** devem implementar o método calcularIPVA() de forma que:

- **Carros** pagam 4% do valor estimado do veículo.
- **Caminhões** pagam 1,5% do valor estimado do veículo.
- **Ônibus** pagam 2% do valor estimado do veículo.

O valor estimado de cada veículo deve ser calculado com base em uma regra própria para cada tipo de veículo, levando em conta o ano de fabricação. Essa regra pode ser baseada em uma constante para simplificação:

- O valor estimado de um Carro reduz R$2.000 a cada ano desde sua fabricação.
- O valor estimado de um Caminhão reduz R$5.000 a cada ano.
- O valor estimado de um Ônibus reduz R$3.000 a cada ano.

O sistema deve exibir o valor do IPVA e todos os detalhes dos veículos cadastrados.

Implemente o sistema com as seguintes classes:

- **Veiculo** (classe abstrata)
- **Carro** (subclasse de Veiculo)
- **Caminhão** (subclasse de Veiculo)
- **Ônibus** (subclasse de Veiculo)

Observações:

- Use encapsulamento com getters e setters quando necessário.
- A classe Veiculo deve ter um construtor que inicialize os atributos comuns.
- As subclasses devem chamar o construtor da classe abstrata.
- O método exibirDetalhes() deve ser sobrescrito em cada subclasse.

Exercício 3 (Interface) - Desenvolva um sistema para gerenciar um hotel utilizando interfaces múltiplas. O sistema deve permitir a criação de diferentes tipos de acomodações: **QuartoSimples**, **QuartoDuplo** e **Suíte**. Cada tipo de acomodação deve implementar a interface **Acomodacao** e a interface **ServicoAdicional**.

Interfaces:

1. Acomodacao
    a. double calcularDiaria(); - Calcula o valor da diária da acomodação.
    b. void exibirDetalhes(int dias); - Exibe os detalhes da acomodação, incluindo o custo total baseado na quantidade de dias.

2. ServicoAdicional
    a. double calcularServico(); - Calcula o custo de serviços adicionais (como café da manhã e limpeza extra).

Tipos de Acomodações:

- O **QuartoSimples** tem uma diária fixa de R$100.
- O **QuartoDuplo** tem uma diária fixa de R$180.
- A **Suíte** tem uma diária fixa de R$350.

Serviços Adicionais:

- Café da manhã: R$20 por pessoa por dia.
- Limpeza extra: R$30 por diária.

Regras do Sistema:

1. O sistema deve solicitar ao usuário a quantidade de dias que a acomodação será utilizada.
2. Cada acomodação pode ter um número variável de pessoas.
3. O método exibirDetalhes(int dias) deve considerar a quantidade de dias para calcular e exibir o custo total, que será a soma do custo da diária multiplicada pelo número de dias e os custos dos serviços adicionais.

Implemente as seguintes classes:

- QuartoSimples (implementa Acomodacao e ServicoAdicional)
- QuartoDuplo (implementa Acomodacao e ServicoAdicional)
- Suite (implementa Acomodacao e ServicoAdicional)

Requisitos:

- Use encapsulamento com getters e setters quando necessário.
- As classes devem implementar os métodos das interfaces de forma apropriada.
- Crie uma classe principal que teste as acomodações e exiba os detalhes, incluindo o custo total (diária * dias + serviços).