Instituto Federal Baiano - Campus Senhor do Bonfim
Curso Técnico em Informática Integrado ao Ensino Médio Integrado
Programação Web II - Prof. Lucas Sampaio Leite

Lista de Exercícios 01

Instruções:

- Esta lista deve ser respondida individualmente ou em duplas. Ela reúne o problema apresentado na Aula 03 e uma questão complementar, que deverá ser resolvida durante o sábado letivo remoto, no dia 12/04/2025.
- As respostas devem ser enviadas até terça-feira, 15/04, antes do início da aula, para o e-mail: lucas.leite@ifbaiano.edu.br, com o assunto: *Lista de Exercícios 01 - WEB II*. No corpo da mensagem, é necessário identificar os membros da equipe.
- Não serão aceitas entregas após o início da aula, pois a correção será realizada durante esse período.
- Esta lista comporá parte da nota da Avaliação 1.

Problema – Sistema Bancário com Herança

Implemente um sistema bancário com uma superclasse Conta, que possui os atributos titular e saldo, além dos métodos depositar(valor) e sacar(valor). A partir dela, crie duas subclasses: ContaCorrente e Poupanca. A ContaCorrente permite saques com uma taxa fixa de R$ 2,00 por operação, enquanto a Poupanca permite saques apenas se houver saldo suficiente e possui um método adicional render_juros() que aplica um rendimento de 0,5% ao saldo atual. Reescreva o método sacar em ambas as subclasses para respeitar essas regras. Demonstre o uso das classes com exemplos de depósitos, saques e rendimento.

Questão Complementar – Cliente e Relacionamento com Conta

Agora que você já implementou as classes Conta, ContaCorrente e Poupanca, crie uma nova classe chamada Cliente, que possui os seguintes atributos:

- nome: nome do cliente
- cpf: CPF do cliente

- contas: uma lista com as contas que ele possui (podem ser contas corrente ou poupança)

A classe deve possuir os seguintes métodos:

- adicionar_conta(conta): adiciona uma conta à lista de contas do cliente.
- saldo_total(): retorna a soma dos saldos de todas as contas do cliente.
- exibir_contas(): imprime o tipo de cada conta e o respectivo saldo.

Todos os métodos devem ser testados e ao executar o script Python, os resultados do teste devem ser exibidos no terminal.

Revisão do conceito de composição para auxíliar na resolução da questão complementar:

O que é Composição?

A composição é um tipo de relacionamento entre classes onde uma classe é composta por objetos de outras classes. Esse relacionamento é do tipo "tem-um", diferente da herança, que é do tipo "é-um".

Ou seja, ao usar composição, estamos dizendo que um objeto de uma classe contém um ou mais objetos de outras classes como parte de sua estrutura.

Por que usar Composição?

- Para reutilizar código sem precisar herdar de uma superclasse.
- Para representar relacionamentos mais realistas do mundo real.
- Para diminuir o acoplamento entre as classes, promovendo maior flexibilidade e manutenção do código.

Um exemplo prático: https://www.youtube.com/watch?v=O6F77N09HdI