DESENVOLVIMENTO DE SOFTWARE - EXERCÍCIOS OO – LISTA 01

Prof. Jason A. P. Sobreiro.

EXERCÍCIO 01:

Num mundo repleto de equações e problemas matemáticos, um estudante de programação se depara com o desafio de simplificar suas tarefas diárias. Ele percebe que, ao invés de recorrer sempre à calculadora física ou ao celular, poderia praticar suas habilidades de programação criando uma ferramenta personalizada.

Inspirado por essa necessidade, ele decide desenvolver sua própria calculadora digital em Java. Essa calculadora simplificada deverá realizar as quatro operações matemáticas básicas: adição, subtração, multiplicação e divisão. Além disso, ele quer garantir que a calculadora seja fácil de usar, com métodos claramente definidos para cada operação.

Dica:

“Crie uma classe `Calculadora` que contenha métodos para adicionar, subtrair, multiplicar e dividir, aceitando dois números reais como parâmetros para cada método e retornando o resultado. Estes métodos devem ser estáticos. Logo, não será necessário instanciar um objeto do tipo Calculadora para poder utilizar esses métodos. Certifique-se de que a divisão por zero seja tratada de maneira a evitar erros de execução.**”**

EXERCÍCIO 02:

Imagine uma pequena biblioteca pessoal, uma coleção que você começou a construir com seus livros favoritos. No entanto, com o tempo e o acúmulo de mais e mais livros, você percebe a necessidade de organizar e acessar rapidamente as informações de cada um deles.

Para ajudar nessa organização, você decide criar um sistema simples em Java que permitirá registrar cada livro com detalhes cruciais: o título, o autor e o ano de publicação. Além de armazenar esses dados, você quer uma maneira fácil de exibir as informações de qualquer livro da sua coleção, para quando quiser relembrar um título ou sugerir uma leitura a um amigo.

Dica:

“Desenvolva a classe Livro com os atributos título, autor e ano. Implemente um construtor para inicializar esses atributos quando um novo livro for instanciado no seu sistema. Adicione um método exibirInfo() para mostrar todos os dados do livro na tela.**”**

EXERCÍCIO 03:

Em uma pequena cidade, um desenvolvedor entusiasta decide criar um sistema para ajudar a gerenciar as contas bancárias dos cidadãos locais. Com o crescimento da comunidade, ficou evidente a necessidade de um método mais eficiente e seguro para o manejo de transações bancárias, que até então eram feitas manualmente e levavam muito tempo.

Inspirado por essa necessidade comunitária, o objetivo é desenvolver um sistema bancário simples em Java, focando na orientação a objetos. Este sistema permitirá a criação de contas bancárias, cada uma capaz de realizar operações básicas como depósitos, saques e consultas de saldo. O desafio é estruturar o programa de forma que cada conta seja um objeto, refletindo os princípios da programação orientada a objetos.

Dica:

“Projete uma classe `ContaBancaria` que encapsule os atributos de uma conta, como `numeroConta`, `nomeTitular` e `saldo`. Utilize modificadores de acesso para garantir o encapsulamento e a segurança dos dados. Implemente métodos para `depositar(valor)`, `sacar(valor)` e um método que retorne o `saldo` atual da conta. Lembre-se de que cada operação deve ajustar o saldo da conta de acordo com a transação realizada.**”**

EXERCÍCIO 04:

Em um mundo cada vez mais digital, um grupo de amigos apaixonados por videogames percebe uma oportunidade de combinar seus hobbies com a prática de programação. Eles decidem criar um sistema para gerenciar perfis de jogadores em seus jogos favoritos, algo que permitiria acompanhar o progresso, as conquistas e as estatísticas de cada jogador de forma personalizada e automatizada.

Inspirado por esse projeto entre amigos, o desafio é desenvolver uma parte desse sistema de gerenciamento de jogadores utilizando Java, com um foco especial na orientação a objetos. O sistema deve permitir a criação de perfis de jogadores, armazenando informações como nome, pontuação e nível. Além disso, deve ser possível atualizar a pontuação e o nível dos jogadores no sistema.

Dica:

“Crie a classe `Jogador`, que deverá conter atributos privados para o `nome`, `pontuacao` e `nivel` do jogador. Inclua um construtor para inicializar esses atributos e métodos públicos para permitir a manipulação segura dessas informações — por exemplo, métodos para aumentar a pontuação, subir de nível e exibir as informações do jogador.**”**

EXERCÍCIO 05:

Em uma tranquila cidade conhecida por sua paixão por automobilismo, um grupo de entusiastas de carros clássicos decide organizar um evento para exibir seus veículos. Para tornar o evento mais interessante, eles planejam criar um sistema que permita aos participantes registrarem seus carros, incluindo informações detalhadas e a capacidade de controlar algumas funcionalidades dos carros, como a velocidade.

Motivado por esse evento de carros clássicos, o desafio é desenvolver uma classe `Carro` em Java, aplicando os princípios da orientação a objetos. Este sistema deve permitir aos usuários criarem objetos `Carro` com informações como marca, modelo, ano e velocidade atual. Além disso, os carros devem ter métodos que permitam acelerar e frear, ajustando a velocidade atual de acordo com a ação realizada.

Dica:

“Desenvolva a classe `Carro` com atributos privados para `marca`, `modelo`, `ano` e `velocidadeAtual`. Implemente um construtor para inicializar os atributos do carro quando um novo objeto é criado. Adicione métodos públicos para `acelerar()`, que aumenta a velocidade do carro, e `frear()`, que diminui a velocidade. Não esqueça de incluir um método para exibir as informações do carro, refletindo a importância da encapsulação e da interação entre os objetos em um programa orientado a objetos.**”**

EXERCÍCIO 06 (Bônus):

Inspirado pela organização de um grande evento de networking, onde os participantes trocam contatos e informações profissionais, um desenvolvedor decide criar um sistema para gerenciar uma agenda de contatos. Este sistema ajudará os usuários a manter suas conexões organizadas, permitindo-lhes adicionar, remover e buscar contatos facilmente.

Dica:

“Crie as classes Agenda e Contato em Java. A classe Contato deve armazenar informações como nome e telefone, enquanto a classe Agenda deve gerenciar uma lista de contatos, com métodos para adicionar novos contatos e buscar contatos por nome.**”**