POO

1. **Classe Bichinho Virtual:**Crie uma classe que modele um Tamagushi (Bichinho Eletrônico):
    1. Atributos: Nome, Fome, Saúde e Idade b. Métodos: Alterar Nome, Fome, Saúde e Idade; Retornar Nome, Fome, Saúde e Idade Obs: Existe mais uma informação que devemos levar em consideração, o Humor do nosso tamagushi, este humor é uma combinação entre os atributos Fome e Saúde, ou seja, um campo calculado, então não devemos criar um atributo para armazenar esta informação por que ela pode ser calculada a qualquer momento.

2. **Classe Bichinho Virtual++:** Melhore o programa do bichinho virtual, permitindo que o usuário especifique quanto de comida ele fornece ao bichinho e por quanto tempo ele brinca com o bichinho. Faça com que estes valores afetem quão rapidamente os níveis de fome e tédio caem.

3. Crie uma "porta escondida" no programa do programa do bichinho virtual que mostre os valores exatos dos atributos do objeto. Consiga isto mostrando o objeto quando uma opção secreta, não listada no menu, for informada na escolha do usuário. Dica: acrescente um método especial str() à classe Bichinho.

4. Crie uma Fazenda de Bichinhos instanciando vários objetos bichinho e mantendo o controle deles através de uma lista. Imite o funcionamento do programa básico, mas ao invés de exigis que o usuário tome conta de um único bichinho, exija que ele tome conta da fazenda inteira. Cada opção do menu deveria permitir que o usuário executasse uma ação para todos os bichinhos (alimentar todos os bichinhos, brincar com todos os bichinhos, ou ouvir a todos os bichinhos). Para tornar o programa mais interessante, dê para cada bichinho um nivel inicial aleatório de fome e tédio.

5. **Classe Macaco:** Desenvolva uma classe Macaco,que possua os atributos nome e bucho (estomago) e pelo menos os métodos comer(), verBucho() e digerir(). Faça um programa ou teste interativamente, criando pelo menos dois macacos, alimentando-os com pelo menos 3 alimentos diferentes e verificando o conteúdo do estomago a cada refeição. Experimente fazer com que um macaco coma o outro. É possível criar um macaco canibal?

6. Exercicio com Abstração, Herança, Encapsulamento e Polimorfismo:
Criar um sistema bancario(extremamente simple) que tem clientes, contas e um banco. A ideia é que o cliente tenha uma conta(poupança ou corrente) e que possa sacar/depositar nessa conta. Contas corrente tem um limite extra.
    1. Conta(ABC)
        i. ContaCorrente
        ii. ContaPoupanca

2. Pessoa(ABC)
    i. Cliente
        1. Cliente -> Conta
3. Banco
    i. Banco -> Cliente
    ii. Banco -> Conta
4. Dicas: Criar classe Cliente que herda da classe Pessoa(Herança)
    i. Pessoa tem nome e idade(com getters)
    ii. Cliente TEM conta(Agregação da classe ContaCorrente ou ContaPoupança)
5. Criar Classes ContaPoupança e ContaCorrente que herdam Conta
    i. ContaCorrente deve ter um limite extra
    ii. Contas tem Agencia, numero da conta e saldo
    iii. Contas devem ter método para deposito
    iv. Conta(Classe Pai) deve ter o metodo sacar abstrato(Abstração e polimorfismo - as classes filhas que implementam o método sacar)
6. Criar classe Banco para AGREGAR classes de clientes e de contas(Agregação)
7. Banco será responsável por autenticar o cliente e as contas da seguinte maneira:
    i. Banco tem contas e cliente(Agregação)
    ii. Checar se a agencia é daquele banco
    iii. Checar se o cliente é daquele banco
    iv. Checar se a conta é daquele banco
8. Só sera possivel sacar se passar na autenticação do banco(descrita acima)
9. Banco autentica por um método.