Lista de Exercícios 02

Implementação de Classes e Objetos: Atividades sobre UML, construtores, métodos, atributos, getters, setters e modificadores de visibilidade

Cada aluno deverá, individualmente, resolver os exercícios abaixo:

Orientações:

- Para cada exercício faça:
  a) Elabore o diagrama de classes utilizando UML;
  b) Implemente a classe em Java;
  c) Implemente um aplicativo específico.

1. A fim de representar funcionários em uma empresa, crie uma classe chamada `Funcionario` que inclua as três informações a seguir como atributos:
   • um primeiro nome,
   • um sobrenome, e
   • um salário mensal.

   a) A classe deve ter um construtor que inicializa os três atributos.
   b) Forneça um método `set` e `get` para cada atributo.
   c) Se o salário mensal não for positivo, configure-o como 0.0.
   d) Escreva um aplicativo de teste que demonstra as capacidades da classe.
   e) Crie duas instâncias da classe e exiba o salário anual de cada instância.
   f) Então dê a cada empregado um aumento de 10% e exiba novamente o salário anual de cada empregado.

2. Implemente uma classe `Proprietário` com os seguintes atributos:
   - Nome
   - CPF
   - RG

- Telefone
- Rua
- Bairro
- Cidade
- Estado
- Cep
- Complemento

a) Faça o encapsulamento dos atributos.
b) Os atributos `nome, cpf e rg` são obrigatórios.
    i. Dica: crie um construtor com esses parâmetros.
c) Implemente um método que verifica se o `cpf` informado é válido ou não.
d) Caso o `cpf` informado seja inválido atribua o valor 0 ao atributo.
e) Implemente um método que exiba o estado da classe.
    i. Dica: O estado é composto por todos os atributos da classe.
f) Implemente um método que calcule e exiba a idade a partir da Data de Nascimento.

3. Implemente uma classe `Carro` com os seguintes atributos na classe:
    - Modelo
    - Cor
    - Ano
    - Marca
    - Chassi
    - Proprietário
    - Velocidade máxima
    - Velocidade atual
    - Número de portas
    - Teto solar
    - Número de marchas
    - Cambio automático
    - Volume de combustível
    - Autonomia

    a) Faça o encapsulamento dos atributos.

b) Os atributos `autonomia`, `número de marchas`, `volume do combustível` são obrigatórios.
c) Implemente o método `acelerar` que aumenta a velocidade de 1 em 1 km/h.
d) Implemente o método `freiar` que para o carro.
    i. Velocidade = 0 km/h.
e) Implemete o método `trocar marcha`.
i. Sobe o número da marcha do carro de 1 em 1 até o número total de marchas do carro.
f) Implemente o método `reduz a marcha`;
    i. Reduz o número da marcha do carro de 1 em 1 até a marcha ré.
g) Implemente um método que exiba o estado da classe.
h) Implemente um método para informar a quantidade de combustível necessária para percorrer uma quantidade em km informada via parâmetro.

4. Escreva um programa completo para jogar o jogo da velha. Para tanto crie uma classe `JogoDaVelha`:
    a) A classe deve conter como dados privados um array bidimensional 3x3 para representar a grade do jogo.
    b) Crie uma enumeração para representar as possibilidades de ocupação de uma casa na grade (vazia, jogador 1 ou jogador 2).
    c) O construtor deve inicializar a grade como vazia.
    d) Forneça um método para exibir a grade.
    e) Permita dois jogadores humanos.
    f) Forneça um método para jogar o jogo; todo movimento deve ocorrer em uma casa vazia; depois de cada movimento, determine se houve uma derrota ou um empate.

5. Crie uma classe para representar `Datas`.
    a) Represente uma data usando três atributos:
        i. `dia`,
        ii. `mês`,
        iii. `ano`.
    b) A classe deve ter um construtor que inicializa os três atributos e verifica a validade dos valores fornecidos.

c) Forneça um construtor sem parâmetros que inicializa a data com a data atual fornecida pelo sistema operacional.
d) Forneça um método `set` um `get` para cada atributo.
e) Forneça o método `toString` para retornar uma representação da data como texto. Considere que a data deve ser formatada mostrando o dia, o mês e o ano separados por barra (/), por exemplo 14/11/2024.
f) Forneça uma operação para avançar uma data para o dia seguinte.
g) Garanta que uma instância desta classe sempre esteja em um estado consistente.

Bom trabalho.