



## Lista de exercícios POO

1. Crie uma classe para representar uma **Pessoa** com os atributos privados de nome, data de nascimento e altura. Crie os métodos públicos necessários para **getters** e **setters** e também um método para imprimir todos dados de uma pessoa. Crie um método para calcular a idade da pessoa.
2. Crie uma classe para implementar uma **ContaCorrente**. A classe deve possuir os seguintes atributos: número da conta, nome do correntista e saldo. Os métodos são os seguintes: **alterarNome**, **depósito** e **saque**; No construtor, saldo é opcional, com valor default zero e os demais atributos são obrigatórios.
3. Crie uma classe para representar uma **BombaCombustivel**. A classe **BombaCombustivel** deve conter os seguintes atributos: tipo de combustível, valor por litro e quantidade de combustível. Além desses atributos a classes deve conter os seguintes métodos.
  - a. **abastecerPorValor**; //método onde é informado o valor a ser abastecido e mostra a quantidade de litros que foi colocada no veículo
  - b. **abastecerPorLitro**; // método onde é informado a quantidade em litros de combustível e mostra o valor a ser pago pelo cliente.
  - c. **alterarValor**; //altera o valor do litro do combustível.
  - d. **alterarCombustivel**; //altera o tipo do combustível.
  - e. **alterarQuantidadeCombustivel**; //altera a quantidade de combustível restante na bomba.

Crie uma classe que contenha um método main para testar sua classe BombaCombustível

4. Crie uma classe denominada **Elevador** para armazenar as informações de um elevador dentro de um prédio. A classe deve armazenar o andar atual (térreo = 0), total de andares no prédio (desconsiderando o térreo), capacidade do elevador e quantas pessoas estão presentes nele. A classe deve também disponibilizar os seguintes métodos:
  - a. **Inicializa**: que deve receber como parâmetros a capacidade do elevador e o total de andares no prédio (os elevadores sempre começam no térreo e vazio);
  - b. **Entra**: para acrescentar uma pessoa no elevador (só deve acrescentar se ainda houver espaço);
  - c. **Sai**: para remover uma pessoa do elevador (só deve remover se houver alguém dentro dele);
  - d. **Sobe**: para subir um andar (não deve subir se já estiver no último andar);
  - e. **Desce**: para descer um andar (não deve descer se já estiver no térreo);
5. Escreva uma classe cujos objetos representam alunos matriculados em uma disciplina. Cada objeto dessa classe deve guardar os seguintes dados do aluno: matrícula, nome, 2 notas de prova e 1 nota de trabalho. Escreva os seguintes métodos para esta classe:
  - a. **media**: calcula a média final do aluno (cada prova tem peso 2,5 e o trabalho tem peso 2)
  - b. **final**: calcula quanto o aluno precisa para a prova final (retorna zero se ele não for para a final)
6. Crie uma classe chamada **Invoice** que possa ser utilizado por uma loja de suprimentos de informática para representar uma fatura de um item vendido na loja. Uma fatura deve incluir as seguintes informações como atributos:
  - a. o número do item faturado,
  - b. a descrição do item,
  - c. a quantidade comprada do item e
  - d. o preço unitário do item.



Sua classe deve ter um **construtor** que inicialize os quatro atributos. Se a quantidade não for positiva, ela deve ser configurada como 0. Se o preço por item não for positivo ele deve ser configurado como 0.0. Forneça os métodos **getters** e **setters** para cada variável de instância. Além disso, forneça um método chamado **getInvoiceAmount** que calcula o valor da fatura (isso é, multiplica a quantidade pelo preço por item) e depois retorna o valor como um double. Escreva um aplicativo de teste que demonstra as capacidades da classe **Invoice**.

7. A fim de representar funcionários em uma empresa, crie uma classe chamada **Funcionario** que inclui as três informações a seguir como atributos:

- a. um primeiro nome,
- b. um sobrenome
- c. um salário mensal

Sua classe deve ter um construtor que inicializa os três atributos. Forneça os métodos **getters** e **setters** para cada atributo. Se o salário mensal não for positivo, configure-o como 0.0. Escreva um aplicativo de teste que demonstra as capacidades da classe. Crie duas instâncias da classe e exiba o salário anual de cada instância. Então dê a cada empregado um aumento de 10% e exiba novamente o salário anual de cada empregado.

8. Escreva uma classe **Data** cuja instância (objeto) represente uma data. Esta classe deverá dispor dos seguintes métodos:

- a. **Construtor**: define a data que determinado objeto (através de parâmetro), este método verifica se a data está correta, caso não esteja a data é configurada como 01/01/0001
- b. **Compara**: recebe como parâmetro um outro objeto da Classe data, compare com a data corrente e retorne:
  - i. 0 se as datas forem iguais;
  - ii. 1 se a data corrente for maior que a do parâmetro;
  - iii. -1 se a data do parâmetro for maior que a corrente.
- c. **GetDia**: retorna o dia da data
- d. **GetMes**: retorna o mês da data
- e. **GetMesExtenso**: retorna o mês da data corrente por extenso
- f. **GetAno**: retorna o ano da data
- g. **IsBissexto**: retorna verdadeiro se o ano da data corrente for bissexto e falso caso contrário
- h. **Clone**: o objeto clona a si próprio, para isto, ele cria um novo objeto da classe Data com os mesmos valores de atributos e retorna sua referência pelo método

9. Escreva uma classe em que cada objeto representa um **Voo** que acontece em determinada data e em determinado horário. Cada voo possui no máximo 100 passageiros, e a classe permite controlar a ocupação das vagas. A classe deve ter os seguintes métodos:

- a. **Construtor**: configura os dados do voo (recebidos como parâmetro): número do voo, data (para armazenar a data utilize um objeto da classe Data, criada na questão anterior);
- b. **ProximoLivre**: retorna o número da próxima cadeira livre
- c. **Verifica**: verifica se o número da cadeira recebido como parâmetro está ocupada
- d. **Ocupa**: ocupa determinada cadeira do voo, cujo número é recebido como parâmetro, e retorna verdadeiro se a cadeira ainda não estiver ocupada (operação foi bem sucedida) e falso caso contrário
- e. **Vagas**: retorna o número de cadeiras vagas disponíveis (não ocupadas) no voo
- f. **GetVoo**: retorna o número do voo

10. Crie uma classe para representar um jogador de futebol, com os atributos:

- a. nome;



- b. posição;
- c. data de nascimento;
- d. nacionalidade;
- e. altura;
- f. peso;

Crie os métodos públicos necessários para **getters** e **setters** e também um método para imprimir todos os dados do jogador. Crie um método para calcular a idade do jogador e outro método para mostrar quanto tempo falta para o jogador se aposentar. Para isso, considere que os jogadores da posição de defesa se aposentam em média aos 40 anos, os jogadores de meio-campo aos 38 e os atacantes aos 35.

11. Crie uma classe chamada **Ingresso**, que possui um valor em reais e um método **imprimirValor**. Crie uma classe **IngressoVIP**, que herda de **Ingresso** e possui um valor adicional. Crie um método que retorne o valor do ingresso VIP (com o adicional incluído). Crie um programa para criar as instâncias de **Ingresso** e **IngressoVIP**, mostrando a diferença de preços.
12. Crie uma classe **Agenda** que pode armazenar 10 pessoas e que seja capaz de realizar as seguintes operações:
  - a. void **armazenarPessoa**(String nome, int idade, float altura); //armazena a pessoa em um array
  - b. void **removerPessoa**(String nome); //remove a pessoa do array
  - c. int **buscarPessoa**(String nome); // informa em que posição da agenda está a pessoa
  - d. void **imprimirAgenda**(); // imprime os dados de todas as pessoas da agenda
  - e. void **imprimirPessoa**(int index); // imprime os dados da pessoa que está na posição “i” da agenda.
13. Crie uma classe **Calculadora**. Esta classe deve implementar as operações básicas (soma, subtração, divisão e multiplicação). Utilizando o conceito de herança crie uma classe chamada calculadora científica que implementa os seguintes cálculos: **raizQuadrada** e a **potencia**. Dica utilize a classe Math do pacote java.lang.
14. Crie uma classe em Java chamada **Fatura** para uma loja de suprimentos de informática. A classe deve conter quatro atributos:
  - a. número (String);
  - b. descrição (String);
  - c. quantidade comprada de um item (int);
  - d. preço por item (double).

A classe deve ter um construtor e os métodos **getters** e **setters**. Além disso, forneça um método chamado **getTotalFatura** que calcula o valor da fatura e depois retorna o valor como um double. Se o valor não for positivo, ele deve ser alterado para 0. Se o preço por item não for positivo, ele deve ser alterado para 0. Escreva uma nova classe chamada **FaturaTeste** (que contenha o método main) que demonstre as capacidades da classe Fatura.

15. Crie uma classe chamada **Funcionário** que inclui três atributos:
  - a. nome (String);
  - b. sobrenome (String);
  - c. salário mensal (double).

A classe deve ter um construtor, métodos getters e setters para cada atributo da classe. Escreva uma classe chamada **FuncionarioTeste** (que contenha o método main) que cria dois objetos da classe **Funcionario** e exibe o salário de cada objeto. Então dê a cada **Funcionario** um aumento de 10% e exiba novamente o



salário anual de cada um deles. Introduza na classe **Funcionario** uma variável de classe capaz de contabilizar o numero de funcionarios que passaram pela empresa até a data.

16. Construa uma classe para representar um carro. O tanque de combustível do carro armazena no máximo 50 litros de gasolina. O carro consome 15 km/litro. Deve ser possível:
  - a. Abastecer o carro com certa quantidade de gasolina;
  - b. Mover o carro em uma determinada distância (medida em km);
  - c. Retornar a quantidade de combustível e a distância total percorrida.

No programa principal, crie 2 carros. Abasteça 20 litros no primeiro e 30 litros no segundo. Desloque o primeiro em 200 km e o segundo em 400 km. Exiba na tela a distância percorrida e o total de combustível restante para cada um.

17. O agendamento de compromissos é uma das tarefas mais comuns para profissionais. Um sistema com essa finalidade deve ser capaz de gerenciar compromissos, atribuindo a cada um o seu tipo (reunião, pagamento, entrega de projeto); data; nome do participante ( pessoa, empresa etc) alguém com quem acontecerá é o compromisso) e seu telefone. Desenvolva um sistema que seja capaz de fazer operações básicas como agendar, remover e alterar compromissos e exibir compromissos por participante e por data.