
	<p align="center">SERVIÇO PÚBLICO FEDERAL MINISTÉRIO DA EDUCAÇÃO INSTITUTO FEDERAL DE SÃO PAULO CAMPUS SÃO PAULO</p>		 INSTITUTO FEDERAL São Paulo
Curso: Tecnologia em Análise e Desenvolvimento de Sistemas	Disciplina: LP1A3	Turma:	Série/Período: 3º
Professor(a): Henrique Dias Pastor	Data: 16/05/2022	Semestre: 1º	Nota (%):
Aluno(a):			

1-) (1,5) Você foi contratado para compor uma estrutura de classes para armazenar os alunos do IFSP. Faça uma superclasse com os atributos: número IFSP, nome do aluno e nome do curso. Em seguida, você deve representar duas situações: aluno que faz iniciação científica (atributos: valor da bolsa (double), orientador, órgão financiador da bolsa, e nome do projeto - todos do tipo string) e aluno que faz estágio (atributos nome da empresa - string, valor da bolsa - double, data de início - inteiro e data de finalização - inteiro). Faça uma estrutura proporcionar o máximo possível de reuso de código. Faça métodos para obter, alterar e imprimir todos os atributos.

2-) (1,5) Escreva uma classe MediaAlunoDisciplina que tenha os atributos número do aluno, nome da disciplina, peso de trabalhos, peso de provas, media de trabalhos, media de provas, com as seguintes especificações:

- o número do aluno, o nome da disciplina deve ser fornecido no momento da criação do objeto;
- os pesos de provas e trabalhos são variáveis da classe. A soma desses atributos sempre deverá resultar no valor 10;
- o nome da disciplina também é uma variável da classe;
- nenhum atributo pode ser alterado por meio de acesso direto. Faça métodos para alterar e garantir acesso aos atributos;
- Crie os seguintes métodos (faça as considerações necessárias sobre tipos de atributos, métodos e especificadores de acesso):
 - calculaMediaPonderada() - calcula a média do aluno, multiplicando as notas de trabalhos e provas pelos seus respectivos pesos e altera o atributo que armazena a média ponderada. Este atributo nunca poderá ser maior que 10 ou menor do que 0;
 - imprimeDados() - imprime todos os atributos e a média ponderada;

3-) (1,5) Escreva uma classe para conter 2 atributos do tipo double chamados largura e altura e chame a classe de Retângulo.

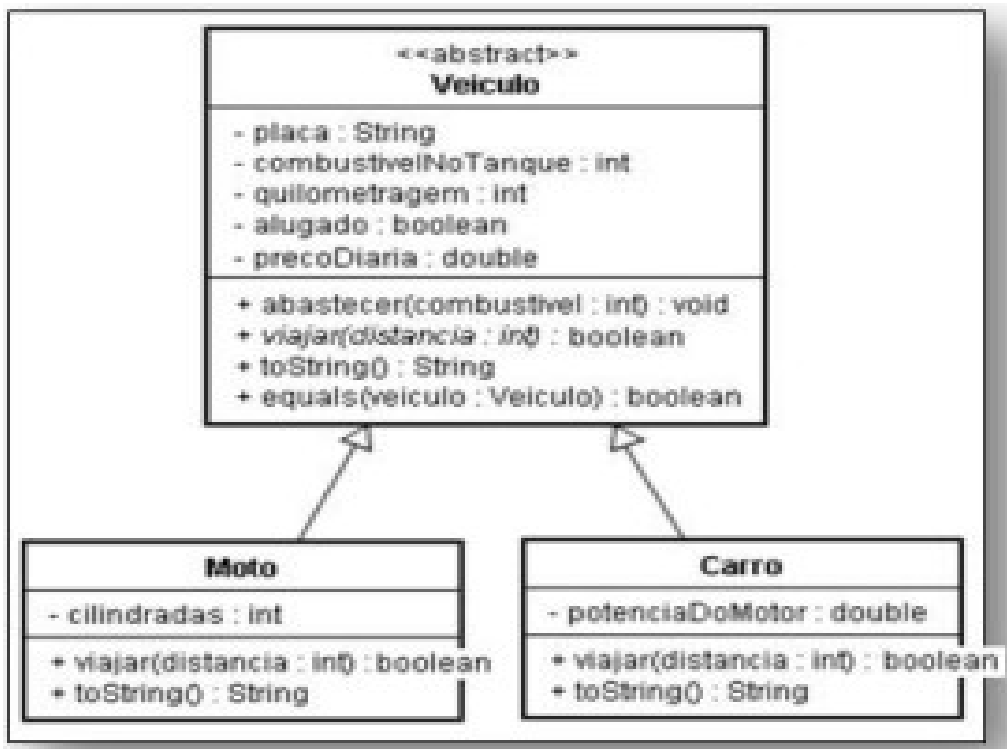
A classe deve possuir os seguintes métodos: um construtor que inicialize os atributos com zero e outro construtor que inicialize os atributos com um valor fixo;

- um método toString();
- um método de acesso para alterar a altura;
- um método de acesso para retornar a altura;
- um método de acesso para alterar a largura;
- um método de acesso para retornar a largura;
- um método para calcular e retornar o perímetro do retângulo;
 $P=2(b+h)$.
- um método para calcular e retornar a área do retângulo. $A=b \cdot h$
- Observação: sempre **valide os atributos** que estão sendo atualizados.
 - Sabendo-se que quadrado é um tipo especial de retângulo (altura e largura **iguais**)
- Crie uma classe Quadrado que estenda de retângulo
- um construtor que inicialize os atributos com zero e outro construtor que inicialize os atributos com um valor fixo;
- Crie todos os métodos de acesso e de cálculo como em Retângulo
- Observação: sempre **valide os atributos** que estão sendo atualizados.
- Crie uma Classe avalia:
- Crie o método: verificaSePossuiUmRetângulo retornando um boolean, recebendo uma lista de Objetos retângulo como entrada;
- Crie o método: **retorneOPrimeiroQuadrado** () – recebe com entrada uma list de retangulo e retornar o primeiro quadrado encontrado;
- Crie um programa para testar as classes anteriores.

4-) (2,5) Utilizando interfaces você pode especificar comportamentos semelhantes para classes possivelmente não relacionadas ou díspares. Há uma preocupação atual com as pegadas de carbono (carbon footprints, emissões anuais de gás carbônico na atmosfera) a partir de instalações que queimam vários tipos de combustíveis para aquecimento, veículos que queimam combustíveis para se mover, e assim por diante. Nesse cenário:

- Crie três pequenas classes **não** relacionadas por herança:
 - classes Building, Car, e Bicycle. De a cada classe alguns atributos e comportamentos (métodos) únicos que ela não tem em comum com as outras classes. Sugestões:
 - Building: número de pessoas (int), uso de energia renovável (boolean), número de lâmpadas (int), uso de ar-condicionado (boolean).

- Car: combustível (string), cilindrada (float).
 - Bicycle: modelo(string)
 - Escreva uma interface CarbonFootprint com um método getCarbonFootprint. Faça cada uma das suas classes implementar essa interface, para que seu método getCarbonFootprint calcule uma pegada de carbono apropriada a cada classe (usando os **atributos sugeridos ou outros**).
 - Escreva um aplicativo que crie 2 objetos de cada uma das três classes. Crie um objeto ArrayList e insira as referências dos objetos instanciados nessa coleção. Finalmente, itere pela coleção, chamando polimorficamente o método getCarbonFootprint de cada objeto.
- 5-) (1,5) Implemente as classes conforme o diagrama:
- O método abastecer deve adicionar o valor passado por parâmetro ao atributo combustívelNoTanque
 - O método equals deve retornar true se o valor do atributo placa for o mesmo para os dois objetos.
 - Método viajar - Moto: o método deve considerar que uma moto faz 30km com 1 litro de combustível. Logo, deve verificar se o combustível no tanque é suficiente para percorrer a distância passada como parâmetro do método. Caso seja, deverá reduzir essa quantidade do atributo combustívelNoTanque e adicionar o valor da distância ao valor do atributo quilometragem. Retorne o valor true caso seja possível realizar a viagem. Caso contrário, retorne false
 - Método viajar - **Carro**: Deve considerar que um carro faz 10km com um litro de combustível. Fazer **as mesmas operações** que as descritas no método viajar da classe **Moto**.
 - Crie a classe TestarVeiculo e instancie objetos das classes implementadas.



6-) (1,5) Implemente uma classe Conta com um atributo saldo. Implemente, além dos construtores, um método depositar e um método sacar. Garanta, utilizando exceções, que não será **depositado** um valor negativo na conta. Crie uma exceção e utilize. Implemente um método main que utilize objetos da classe Conta e realize o tratamento de exceções correspondente.

Implemente uma exceção para tratar a situação em que há a tentativa de fazer um **saque** e deixar o valor negativo. Utilize esta exceção na implementação do método sacar da classe Conta.